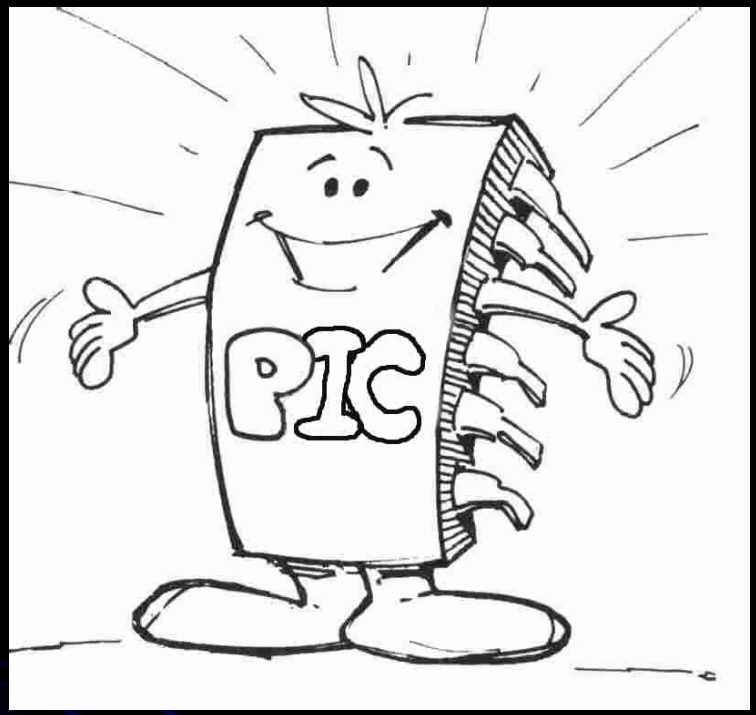


C18

Compilateur C pour PIC 18

www.microchip.com

Première journée

CD-RT Equipe de formation PIC
Académie d'Aix-Marseille

PRISE EN MAIN DE MCC18 DANS MPLAB

```
#include
<stdio.h>

main()
{ puts(" Bonjour
à tous ");
}
```

Fichier source C contenant la fonction main

Bibliothèques en C (texte) *.c

Fichiers header *.h

Bibliothèques pré-compilées (fichiers objet)

Préprocesseur

Remplace les #define et effectue les #include

Compilateur C

Transforme le fichier C en un fichier objet (code machine), les fonctions pré-compilés sont déclarées dans les fichiers *.h

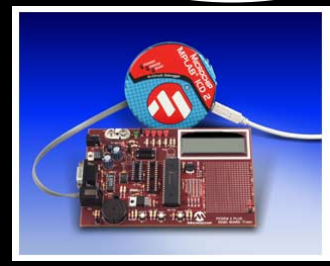
Editeur de lien

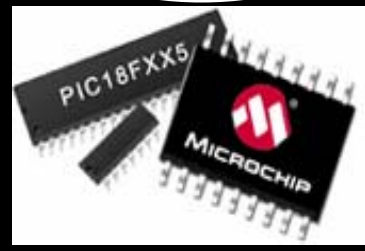
LINKER

Lie (donne des adresses aux fonctions) tous les fichiers objets et crée un fichier exécutable

Fichier de symboles pour debug

Programme exécutable

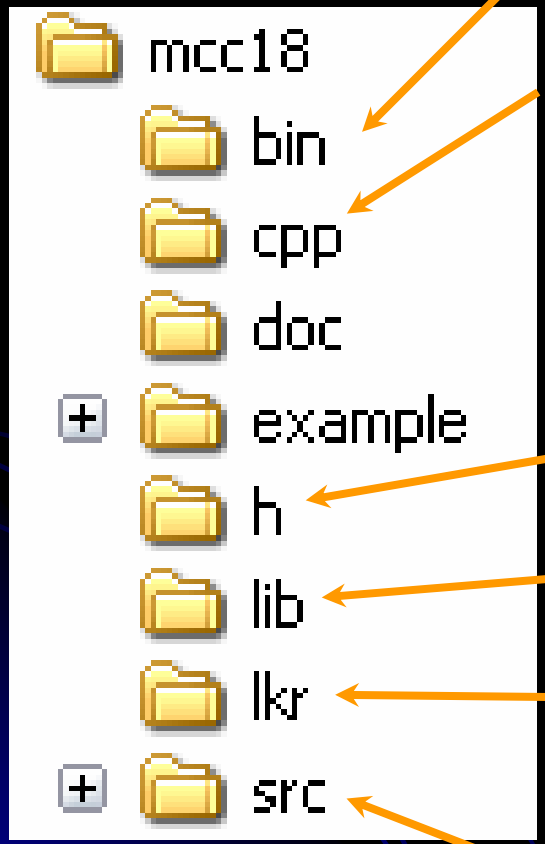




FLUX DE DONNEES D'UN COMPILATEUR C

christian.dupaty@ac-aix-marseille.fr

Arborescence C ANSI



Exécutables : MCC18, MPLINK, MPLIB etc…

Personnalisation du préprocesseur

Headers (*.h)

Librairies (*.lib)

Fichiers de configuration du linker (*.lkr)

Sources ASM et C des librairies

Directives du pré-processeur

#include

Sert à inclure un fichier contenant du code source (.c ou .h) dans un autre fichier.

#include<Nomfichier> → recherche du fichier dans :

Les répertoires mentionnés à l'aide de l'option de compilation /Idirectory

Les répertoires définis à l'aide de la variable d'environnement INCLUDE

#include "Nomfichier" → recherche du fichier dans :

Idem cas précédent **+** Le répertoire courant

p18f4252.h

```
extern volatile near unsigned char PORTA;
extern volatile near union  {
    struct {
        unsigned RA0:1;
        unsigned RA1:1;
        unsigned RA2:1;
        unsigned RA3:1;
        unsigned RA4:1;
        unsigned RA5:1;
        unsigned RA6:1;
    } ;
    struct {
        unsigned AN0:1;
        unsigned AN1:1;
        unsigned AN2:1;
        unsigned AN3:1;
        unsigned :1;
        unsigned AN4:1;
        unsigned OSC2:1;
    } ;
    struct {
        unsigned :2;
        unsigned VREFM:1;
        unsigned VREFP:1;
        unsigned T0CKI:1;
        unsigned SS:1;
        unsigned CLK0:1;
    } ;
    struct {
        unsigned :5;
        unsigned LVDIN:1;
    } ;
} PORTAbits ;
```

```
PORTAbits.RA0 = 1 ; // mise à l'état haut de RA0
```

Directive #pragma config

- MCC18 permet de configurer le microcontrôleur cible sans passer par les menu de MPLAB grâce à la directive #pragma config
- Exemples :
- #pragma config OSC = HS // type d'oscillateur
- #pragma config WDT = OFF // chien de garde
- #pragma config LVP = OFF // mode programmation
- #pragma config DEBUG = ON // mode debug (ICD)

AND : &

Bit1	Bit2	AND
0	0	0
0	1	0
1	0	0
1	1	1

Masquage

PORTA	x	x	x	x	x	x	x	x
&	0	0	0	1	0	0	0	0
=	0	0	0	x	0	0	0	0

Forçage à 0

PORTA	x	x	x	x	x	x	x	x
&	1	1	1	0	1	1	1	1
=	x	x	x	0	x	x	x	x

OR : |

Bit1	Bit2	OR
0	0	0
0	1	1
1	0	1
1	1	1

Forçage à 1

PORTA	x	x	x	x	x	x	x	x
OU	0	0	0	1	0	0	0	0
=	x	x	x	1	x	x	x	x

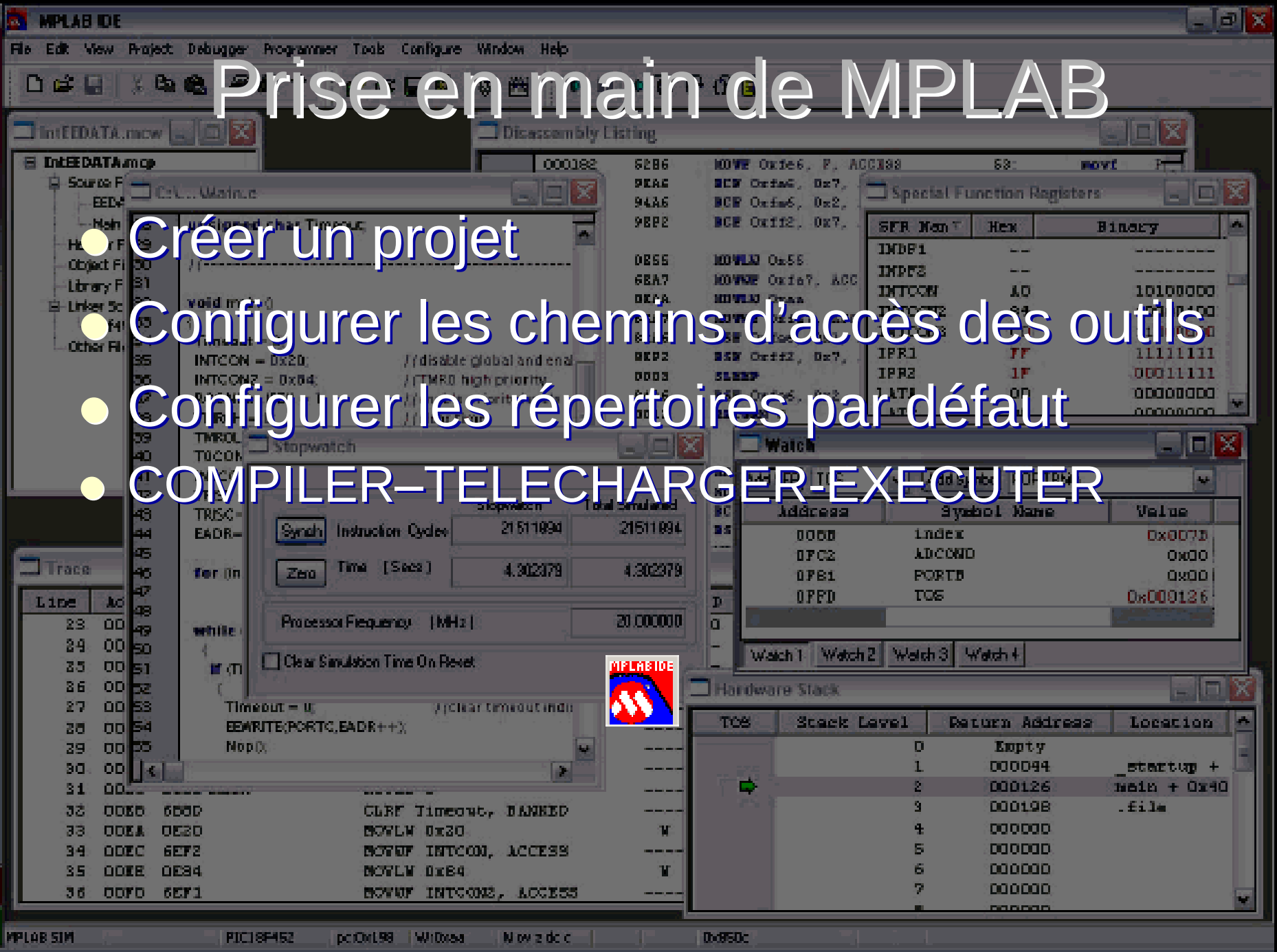
XOR : ^

Bit1	Bit2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

BASCULEMENT

PORTA	x	x	x	x	x	x	x	x
XOR	0	0	0	1	0	0	0	0
=	x	x	x	/x	x	x	x	x

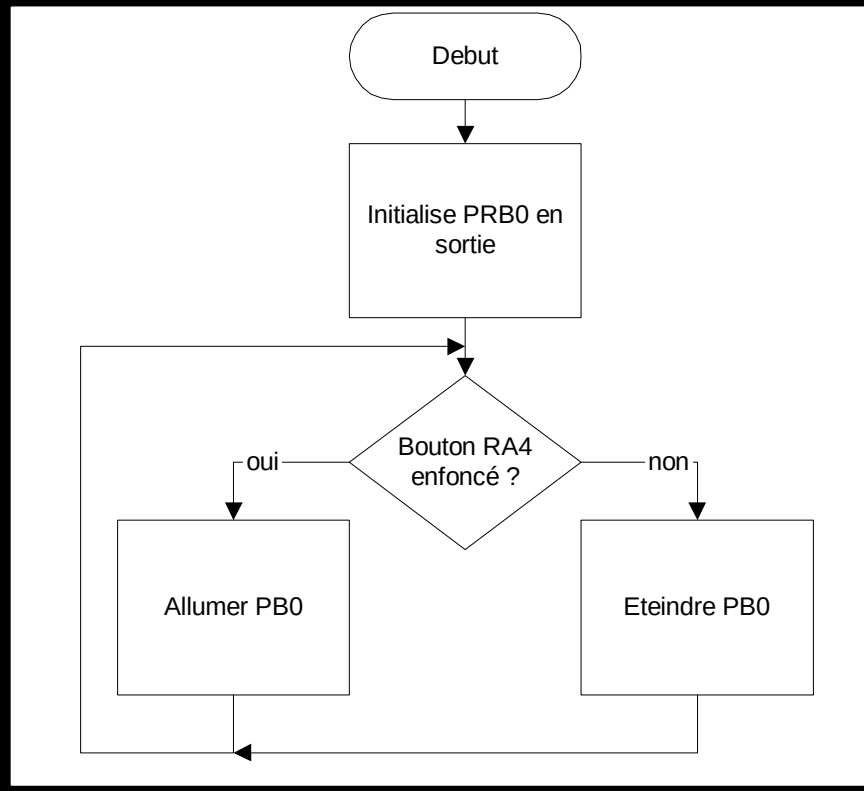
Prise en main de MPLAB



- Créer un projet
- Configurer les chemins d'accès des outils
- Configurer les répertoires par défaut
- COMPILER–TELECHARGER-EXECUTER

Gérer les ports parallèles


```
#include <p18f452.h>
void main(void)
{   TRISB = 0;
    while(1)
   {
       if (PORTA & 0x10) PORTB=1;
           else PORTB=0;
   }
}
```



PORTAbits.RA4

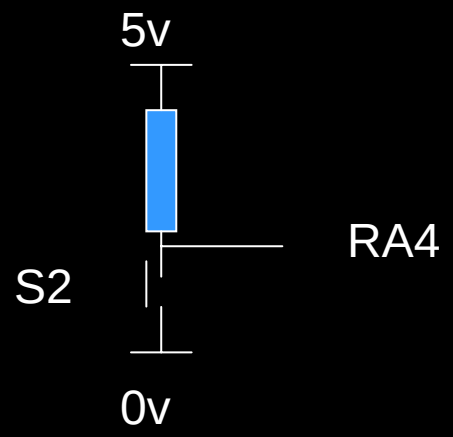
Gérer les ports parallèles

Test de S2 (RA4) (actif à « 0 », pull up)

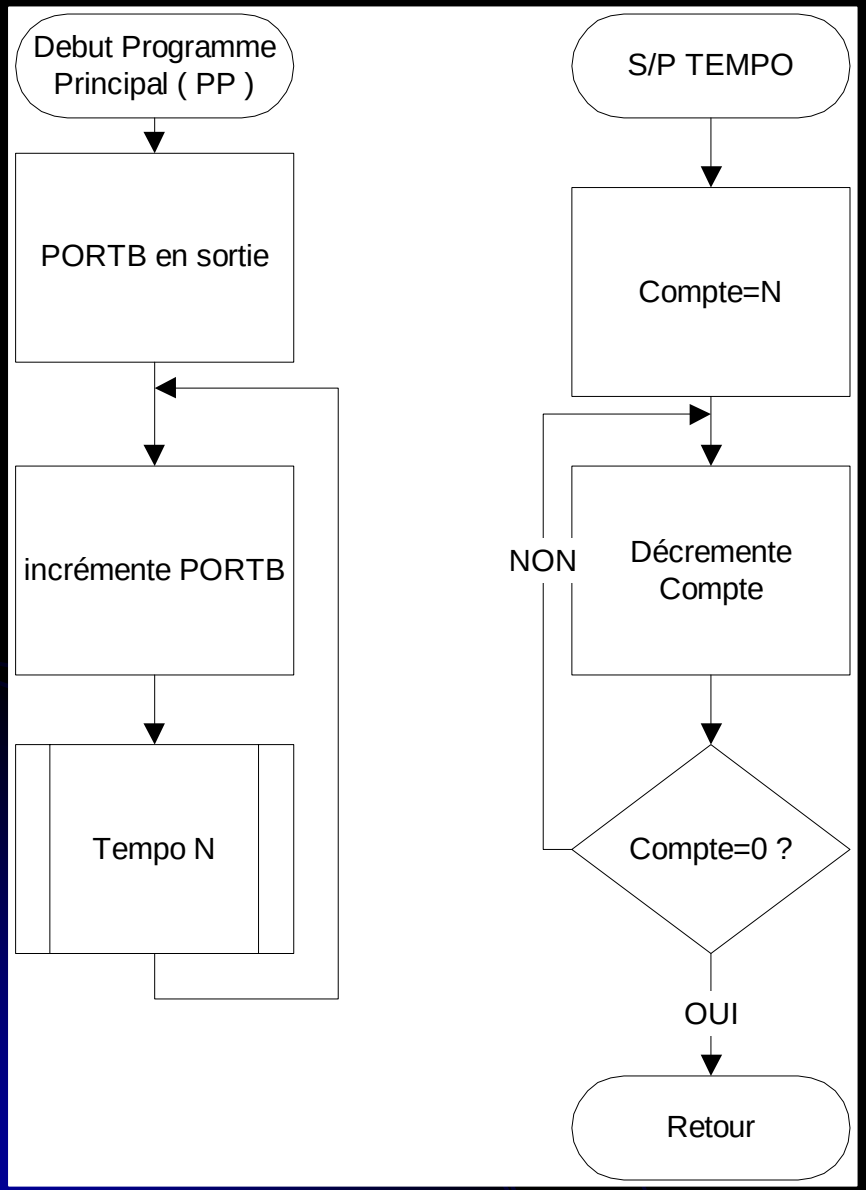
Attendre TANT QUE RA4=1 puis

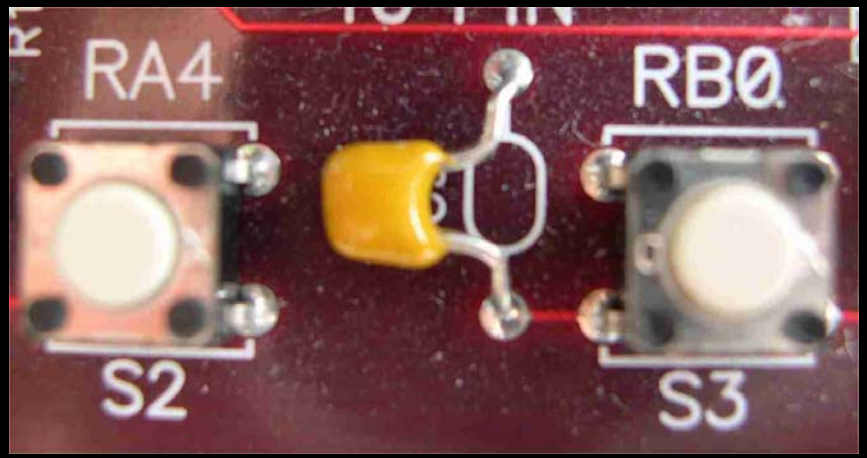
Attendre TANT QUE RA4=0

Afin de limiter les rebonds on peut introduire une temporisation entre les tests.

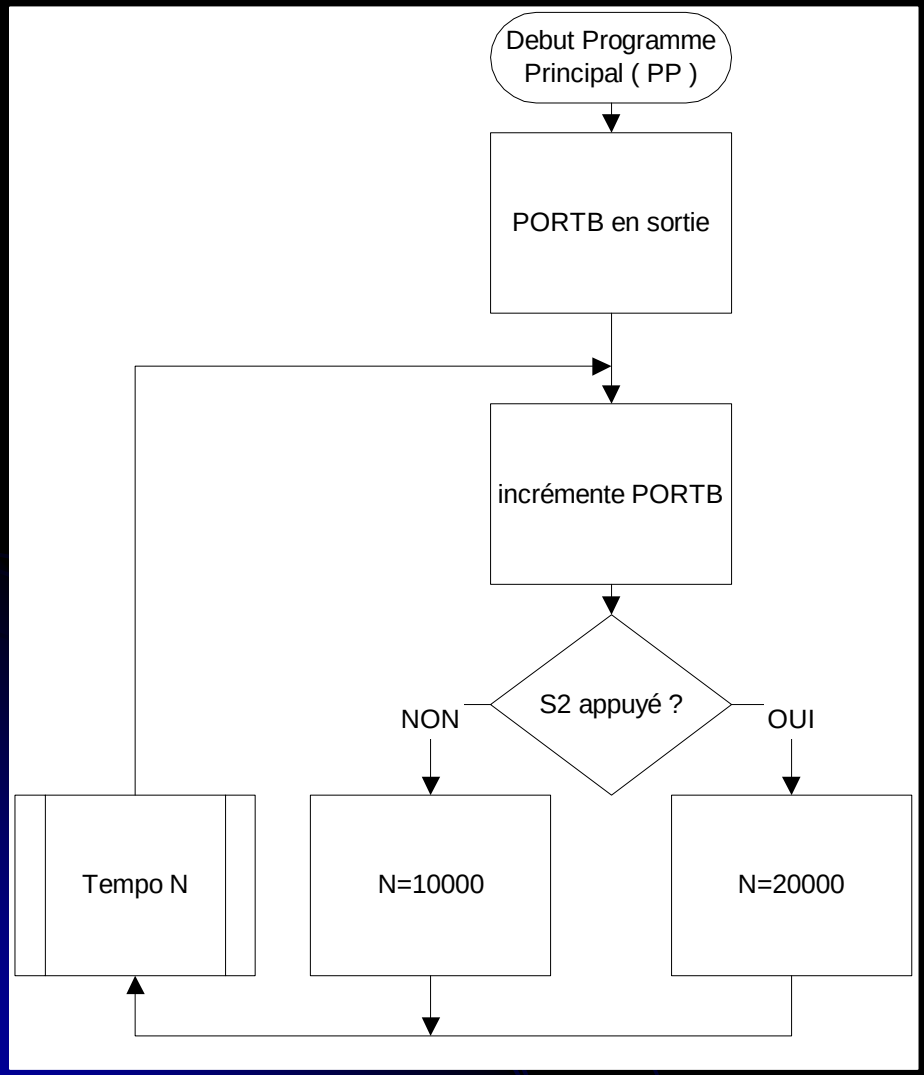


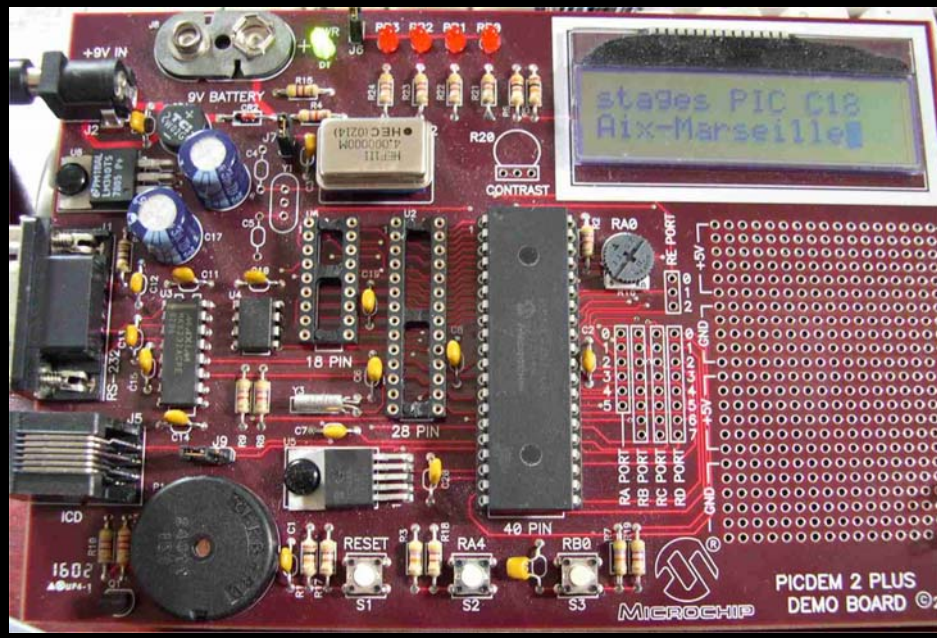
Exercice


Debut Programme Principal (PP)
PORTB en sortie
incrémente PORTB
Tempo N
S/P TEMPO
Compte=N
Décremente Compte
NON
Compte=0 ?
OUI
Retour

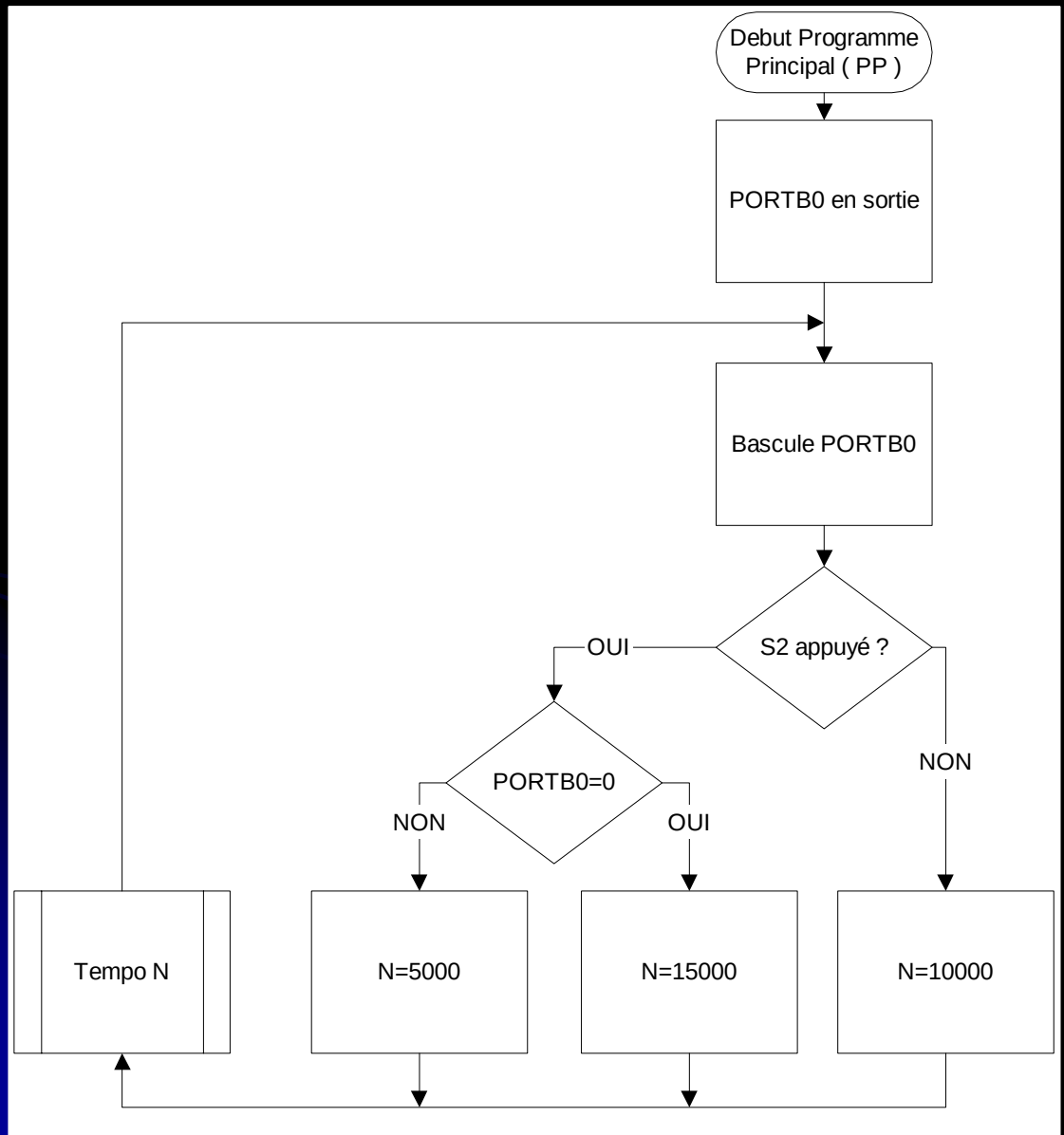

RA4
RB0
S2
S3

Exercice


Debut Programme Principal (PP)
PORTB en sortie
incrémente PORTB
S2 appuyé ?
NON
OUI
Tempo N
N=10000
N=20000


stages PIC C18
Aix-Marseille
+9V IN
9V BATTERY
CONTRAST
18 PIN
28 PIN
40 PIN
RESET
RA4
RB0
S1
S2
S3
MICROCHIP
PICDEM 2 PLUS
DEMO BOARD

Exercice


Debut Programme Principal (PP)
PORTB0 en sortie
Bascule PORTB0
S2 appuyé ?
OUI
NON
PORTB0=0
NON
OUI
Tempo N
N=5000
N=15000
N=10000


Microchip

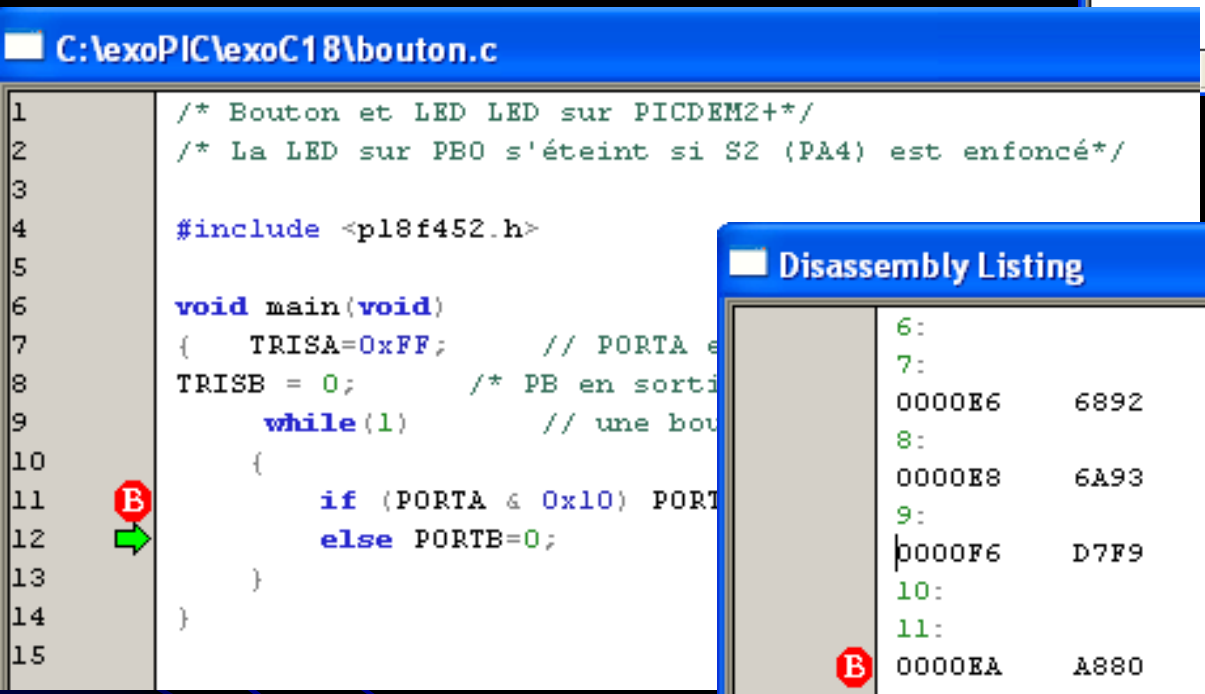
Décalages

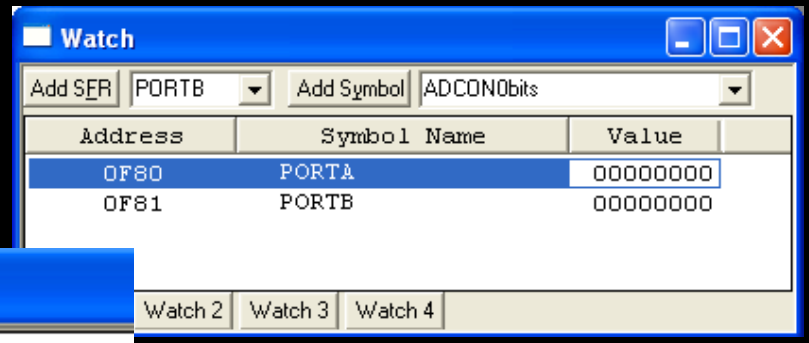
```
#include <p18f452.h>
void wait(int cnt)
{
for (;cnt>0; cnt--);
}
void main(void)
{
int x;
char c=0;
 TRISB = 0;
 PORTB=0b00000001;
while(1)
        {
          if (PORTB==8) c++;
          if (PORTB==1) c--;
          if (!c) PORTB>>=1;
                    else PORTB<<=1;
          if (PORTA&0x10) x= 20000;
                    else x=5000;
          wait(x);
        }
}
```

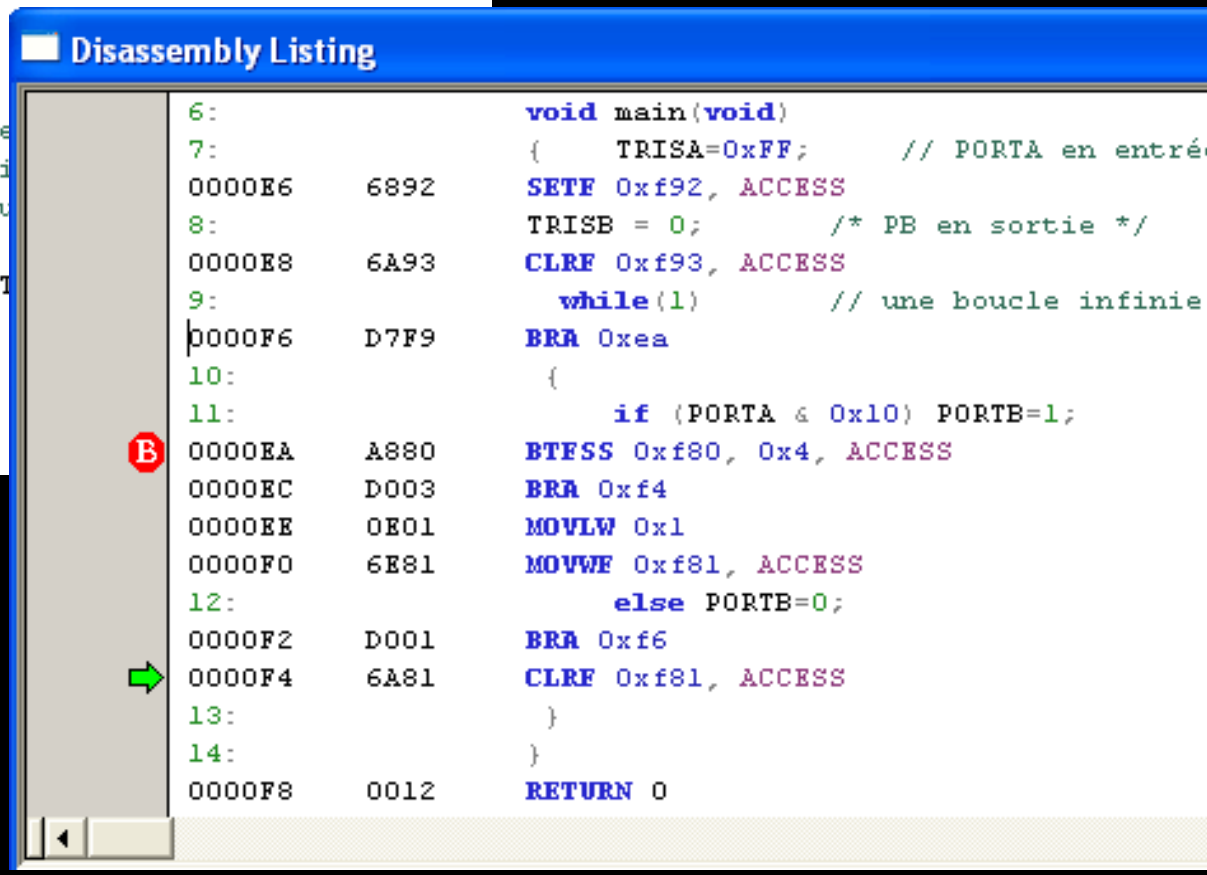
Déclare un entier signé (16bits)

Déclare un caractère signé (8bits)

Mise au point

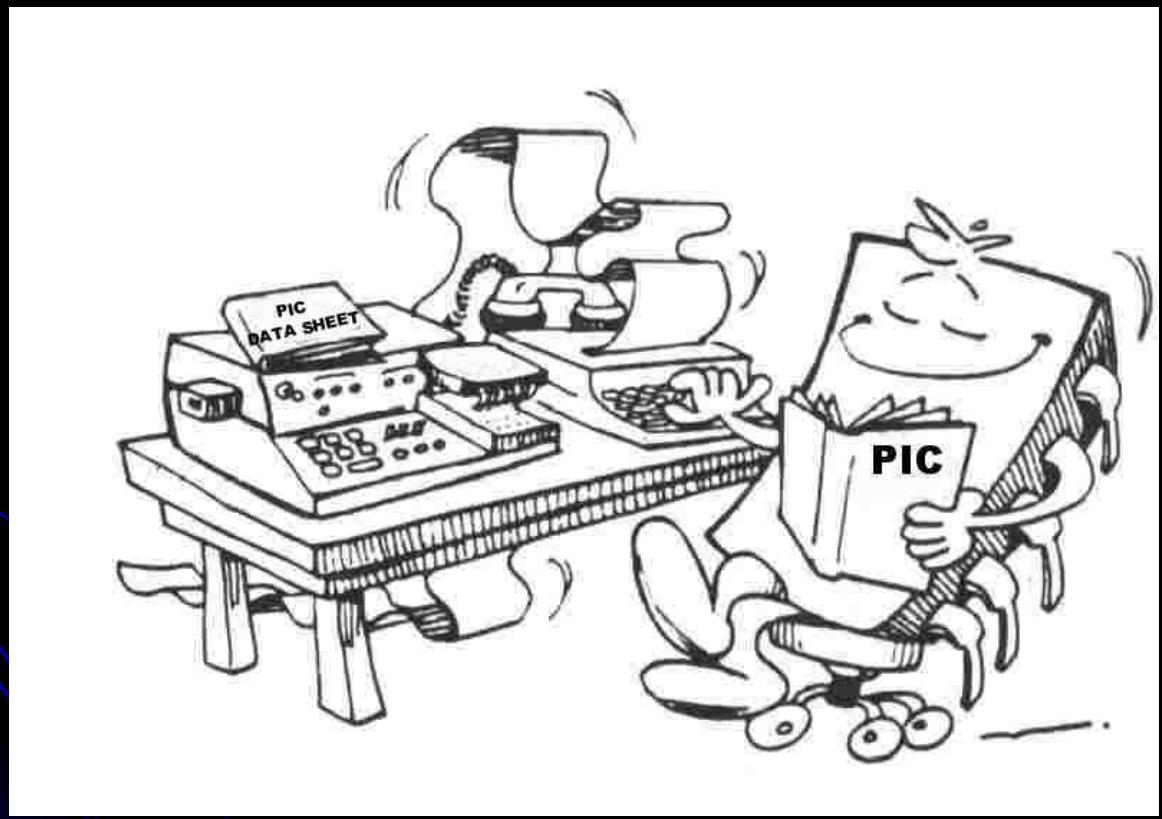






Travaux pratiques

- Faire les exercices précédents



Les bibliothèques de MCC18
Standard CANSI
&
propriétaires

L’édition des liens

Editeur de liens MPLINK

Permet :

- D’indiquer des chemins d’accès à de nouveaux répertoires
- D’inclure des bibliothèques pré-compilées ou des fichiers objet
- De définir l’organisation mémoire du processeur cible
- D’allouer des sections sur le processeur cible
- D’initialiser la pile (taille et emplacement)

18f452i.lkr

Chemins d'accés de bibliothèques ou fichiers objet.

Fichiers objets et bibliothèques précompilées à lier.

Définition de la mémoire programme

Définition de la mémoire Données

Définition de la pile logicielle

```
// Sample linker command file for 18F452i used with MPLAB ICD 2
// $Id: 18f452i.lkr,v 1.2 2002/07/29 19:09:08 sealep Exp $
LIBPATH  .
FILES c018i.o
FILES clib.lib
FILES p18f452.lib
CODEPAGE   NAME=vectors  START=0x0       END=0x29      PROTECTED
CODEPAGE   NAME=page     START=0x2A      END=0x7DBF
CODEPAGE   NAME=debug    START=0x7DC0     END=0X7FFF     PROTECTED
CODEPAGE   NAME=idlocs   START=0x200000   END=0x200007   PROTECTED
CODEPAGE   NAME=config   START=0x300000   END=0x30000D   PROTECTED
CODEPAGE   NAME=devid    START=0x3FFFFE   END=0x3FFFFF   PROTECTED
CODEPAGE   NAME=eedata   START=0xF00000   END=0xF000FF   PROTECTED

ACCESSBANK  NAME=accessram  START=0x0     END=0x7F
DATABANK    NAME=gpr0       START=0x80    END=0xFF
DATABANK    NAME=gpr1       START=0x100   END=0x1FF
DATABANK    NAME=gpr2       START=0x200   END=0x2FF
DATABANK    NAME=gpr3       START=0x300   END=0x3FF
DATABANK    NAME=gpr4       START=0x400   END=0x4FF
DATABANK    NAME=gpr5       START=0x500   END=0x5F3
DATABANK    NAME=dbgspr     START=0x5F4   END=0x5FF   PROTECTED
ACCESSBANK  NAME=accesssfr  START=0xF80   END=0xFFF   PROTECTED
SECTION     NAME=CONFIG     ROM=config
STACK SIZE=0x100 RAM=gpr4
```

C Run Time

Co18.o Initialise la pile logicielle et se branche au début du programme utilisateur (fonction **main**) → minimum de code .

Co18i.o Idem + initialisation des données avant l'appel du programme utilisateur

Co18iz.o Idem co18i.o + initialisation à zéro des variables statiques non initialisées par le programme (compatibilité C ANSI).

Bibliothèques spécifiques

Elles sont contenues dans les bibliothèques **" p*processeur*.lib "** → **P18F452.lib**

Hardware Peripheral Library
Fonctions de gestion des périphériques matériels:

- oADC
- oCapture
- oI2C
- oPorts d'E/S //
- oPWM
- oSPI
- oTimer
- oUSART

Software Peripheral Library
Fonctions de gestion des périphériques externes Afficheur lcd

- oCAN2510
- oI2C logiciel
- oSPI logiciel
- oUART logiciel

delays → Temporisations

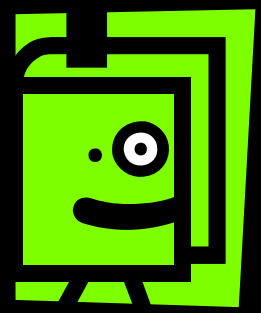
reset → Origine d'un RESET

Bibliothèques « lycée » – Aix-Marseille:

lcdpd2 : gestion de l'afficheur LCD sur PICDEM2+

libpd2 : gestion périphériques avancée (USART, I2C etc…)

Bibliothèques C ANSI



Elles sont contenues dans la bibliothèque " clib.lib ".

math → fonctions mathématiques
stdlib → Fonctions de conversion de données standard C ANSI(atof, itoa etc.)
strings → Fonctions de mémorisation et de manipulation de chaînes de caractères
stdio → Fonctions de gestion des entrées/sorties standards, USART, LCD, clavier
…

Utilisation des bibliothèques exemple : la gestion d'un afficheur LCD

- Une bibliothèque « **bas niveau** » fournissant des fonctions:
  - d'initialisation
  - de contrôle
  - d'affichage **d'un** caractère
- Un bibliothèque « **haut niveau** » au standard CANSI disposant de fonctions de traitement de chaînes de caractères

Gestion bas niveau d'un Afficheur LCD (xlcd.h) (partiel)

Fonctions	Descriptions
void **OpenXLCD**(unsigned char lcdtype);	Initialise l'afficheur LCD
void **putsXLCD**(char *buffer); void **putrsXLCD**(const rom char *buffer);	Affiche une chaîne présente en **RAM** Affiche une chaîne présente en **ROM**
void **SetDDRamAddr**(unsigned char addr);	Fixe l'adresse d'affichage des caractères dans l'afficheur
void **WriteCmdXLCD**(unsigned char ***cmd***);	Envoie une commande à l'afficheur
void **putcXLCD** (unsigned char ***data***);	Envoie un caractère sur l'afficheur.

Bibliotheque « Haut niveau » de gestion des chaines : stdio.h

- Permet le formatage de données ASCII à destination de l'USART (par défaut) ou d'un périphérique utilisateur (ex : LCD)

- Exemples :

  putc('A ');

  puts (« bonjour »);

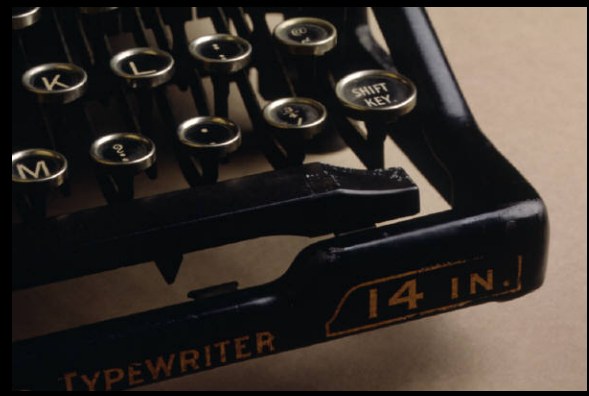
  printf, sprintf, fprintf …

Fontions avancées de stdio.h
printf – sprintf - fprintf

- **printf** envoie une texte formaté sur la sortie standard (l'USART par défaut)
  ex : `printf (« Bonjour »);`
- **sprintf** envoie une texte formaté vers un pointeur
  ex :
```
unsigned char monpointeur[20];
sprintf (monpointeur,« Bonjour »);
```
- **fprintf** envoie une texte formaté sur la sortie désignée
  ex : `fprintf (_H_USER,« Bonjour »);`

Le C18 définit deux sorties : _H_USART et _H_USER

printf




```
int a=10, b=27;
printf(" Formats %d, %x, %X, %#X " ,a,a,a,b);
```

Formats 10, a, A, 0x1B

Formats sur printf (C ANSI)

- %c (char)
- %s ou S (chaîne de caractères)
- %d (entier)
- %u (entier non signé)
- %x ou X (entier affiché en hexadécimal)
- %b ou B (entier affiché en binaire)
- %f (réel en virgule fixe)
- %p ou P (pointeur)
- % (pour afficher le signe %)

\ n nouvelle ligne
\ t tabulation
\ b backspace
\ r retour chariot
\ f form feed
\ ' apostrophe
\ \ antislash
\ " double quote
\ 0 nul

christian.dupaty@ac-aix-marseille.fr

Exemples sur printf – sprintf -fprintf

`printf("Dec : %d %u",a,a);`	`Dec : -27 65509`
`printf("Hex: %#06X %x ",b,b);`	`Hex: 0X00B5 b5`
`printf("Bin: %16b",b);`	`Bin: 0000000010110101`
`printf("Bin: %#010B",b);`	`Bin: 0B10110101`
`printf("%c %c %d",'b',c,(int)c);`	`b A 65`
`printf("J habite %S",chrom);`	`J habite en ROM`
`printf("J habite %s",chram);`	`J habite en RAM`
`printf("pointeur RAM:%p %04P",pram,pram);`	`pointeur RAM:1cd  01CD`
`printf("pointeur ROM:%p %P",prom,prom);`	`pointeur ROM:12Ab 12AB`

L’affichage des réels

Le C18 ne possède pas la fonction ftoa (float to ACSII), printf ne peut donc pas formater les nombre réels

Ajouter le fichier ftoa.c au projet et utiliser ftoa :

unsigned char *ftoa (float x, unsigned char *str,char prec,char format);

x : nombre réel à convertir

str : pointeur sur un octet

prec indique la précision, 0 pour avoir le maximum

si **format** ='s' affichage scientifique 1.6666666E3

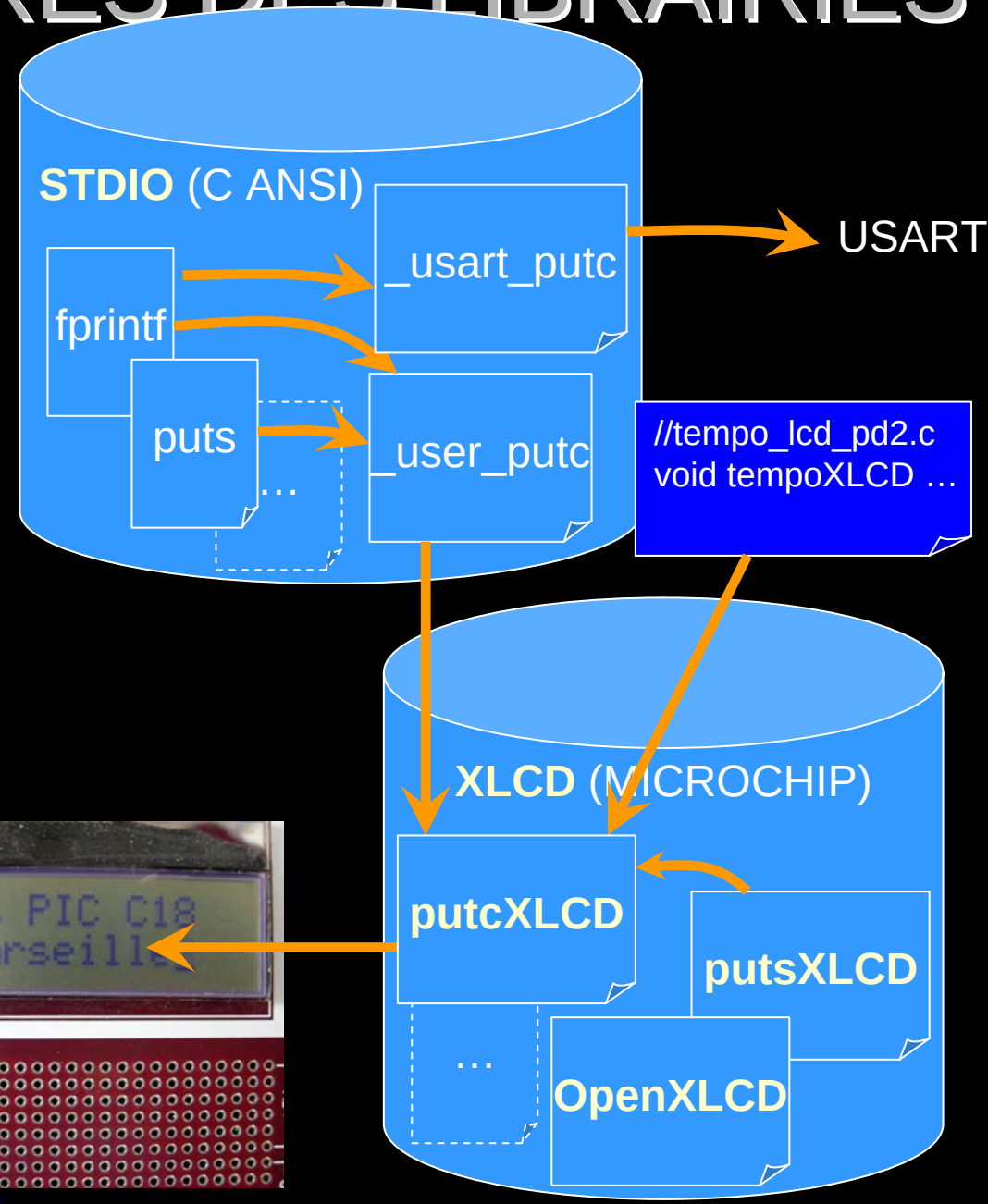
si **format** ='f' affichage classique 1666.6666

```
unsigned char chaine[10];
ftoa(314.152,chaine,2,’s’)
printf (« %s »,chaine);
```

Affichera : 3.14E2

STRUCTURES DES LIBRAIRIES

```
#include <stdio.h>
#include <xlcd.h>
#include <tempo_lcd_pd2.c>

Void main (void)
{
   openxlcd….
   SetDDRamAddr(0);
   fprintf (_H_USER,"stages PIC");
…
}
```



L’afficheur de PICDEM2+

- Les fonctions de gestion bas niveau de la librairie **xlcd.h** (gestion d’un afficheur LCD) de **ne fonctionnent PAS avec l’afficheur LCD du KIT PICDEM2+** . Les temporisations étant trop courtes.
- Il faut recompiler **xlcd** avec le fichier modifié **openxlcd.c**
- Le fichier installTout.bat effectue cette modification et installe la librairie xcldpd2.h

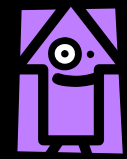
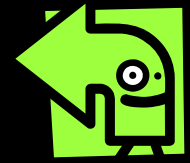
La librairie XLCDPD2.h

- Reconfigure xlcd pour l'afficheur du PICDEM2+
- Contient les fonctions :

int _user_putc(char c); /* redirection de putc sur LCD*/
void initLCDPD2(void); // initialisation afficheur LCD
void gotoxy(unsigned char x, unsigned char y);
void efface(void);
void tempo(unsigned int t);
void decaleLCD(unsigned char c);
void initNouveauxCharacters(void);
ftoa // conversion reel -> ASCII

Utilisation : #include <lcd_pd2.h>


Redirections

- Les fonctions de stdio.h envoient les caractères vers le « flux » stdout.
- Par défaut stdout=_H_USART;
- Pour sortir sur l'afficheur LCD il faut définir la fonction int _user_putc(char c);

```
int _user_putc(char c)
{
        putcXLCD(c)
}
```

Exemple : fprintf.c

```
#include <p18f452.h>
#include <stdio.h>  // pour fprintf
#include <xlcd.h>   // pour OpenXLCD et putcXLCD

// dirige user_putc vers l'afficheur LCD du PD2+
int _user_putc (char c)
{
        putcXLCD(c);
}

void main(void)
{
        SPBRG = 25;       /* configure la vitesse (BAUD) 9600 N 8 1*/
        TXSTA = 0x24;
        RCSTA = 0x90;                   /* active l'USART*/
        OpenXLCD(FOUR_BIT & LINES_5X7 );        //initialise LCD sur PD2
        SetDDRamAddr(0);                                //ligne 0 de l'afficheur
        fprintf (_H_USART, "fprintf USART\n");          // vers USART
        fprintf (_H_USER, "fprintf USER\n" );           // vers LCD
        while(1);
}
```

Structuration et lisibilité

init.c

```
#include <p18f452.h>
#include <stdio.h>
#include <xlcd.h>
int _user_putc (char c)
{
        putcXLCD(c);
}

void init_comm_pd2(void)
{
SPBRG = 25;
TXSTA = 0x24;
RCSTA = 0x90;
OpenXLCD(FOUR_BIT & LINES_5X7 );

}
```

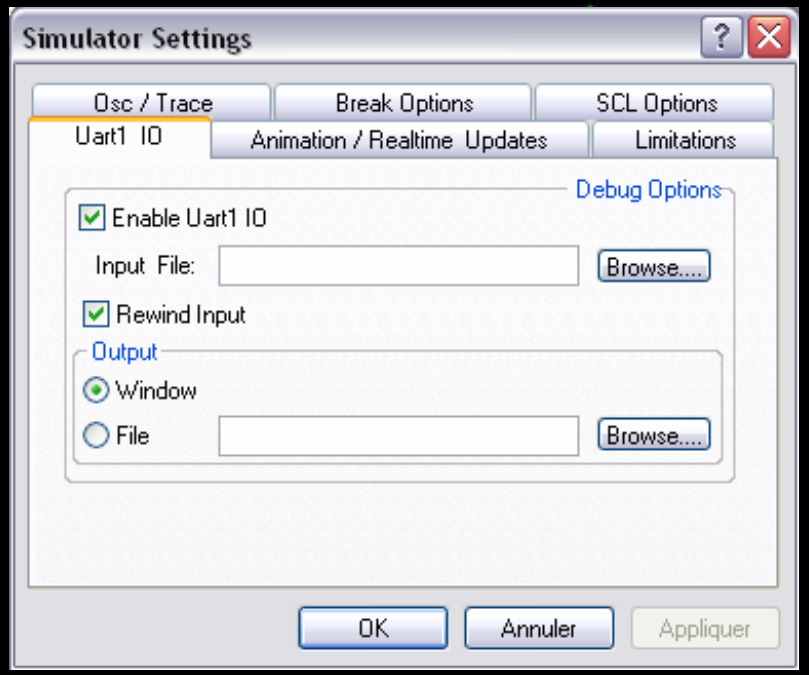
```
#include <p18f452.h>
#include « init.c »
void main(void)
{
Init_comm_pd2();
SetDDRamAddr(0);
fprintf (_H_USART, "fprintf USART\n");
fprintf (_H_USER, "fprintf USER" );
while(1);
}
```

Utiliser le simulateur USART de MPLAB

Activer le simulateur

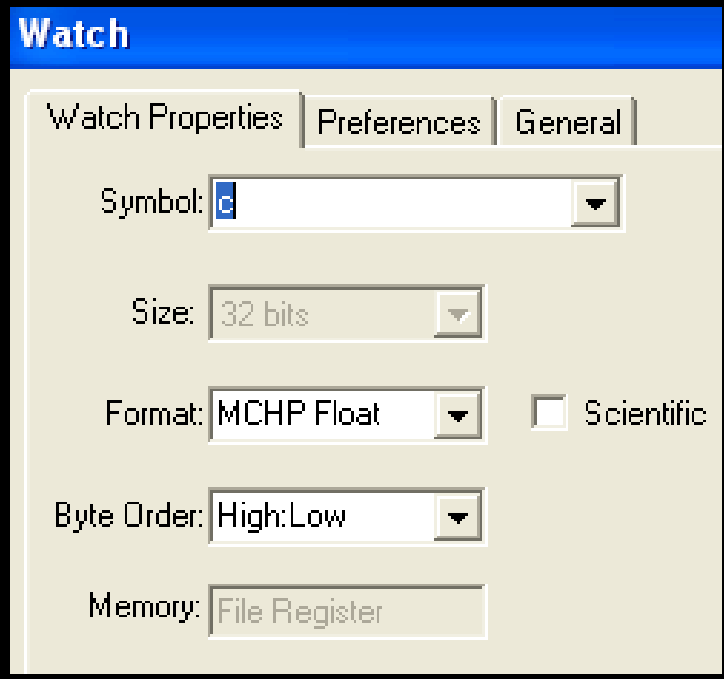
Dans debugger-settings cocher enable UART IO et output windows

Un nouvel onglet UART apparaît dans la fenêtre OUTPUT

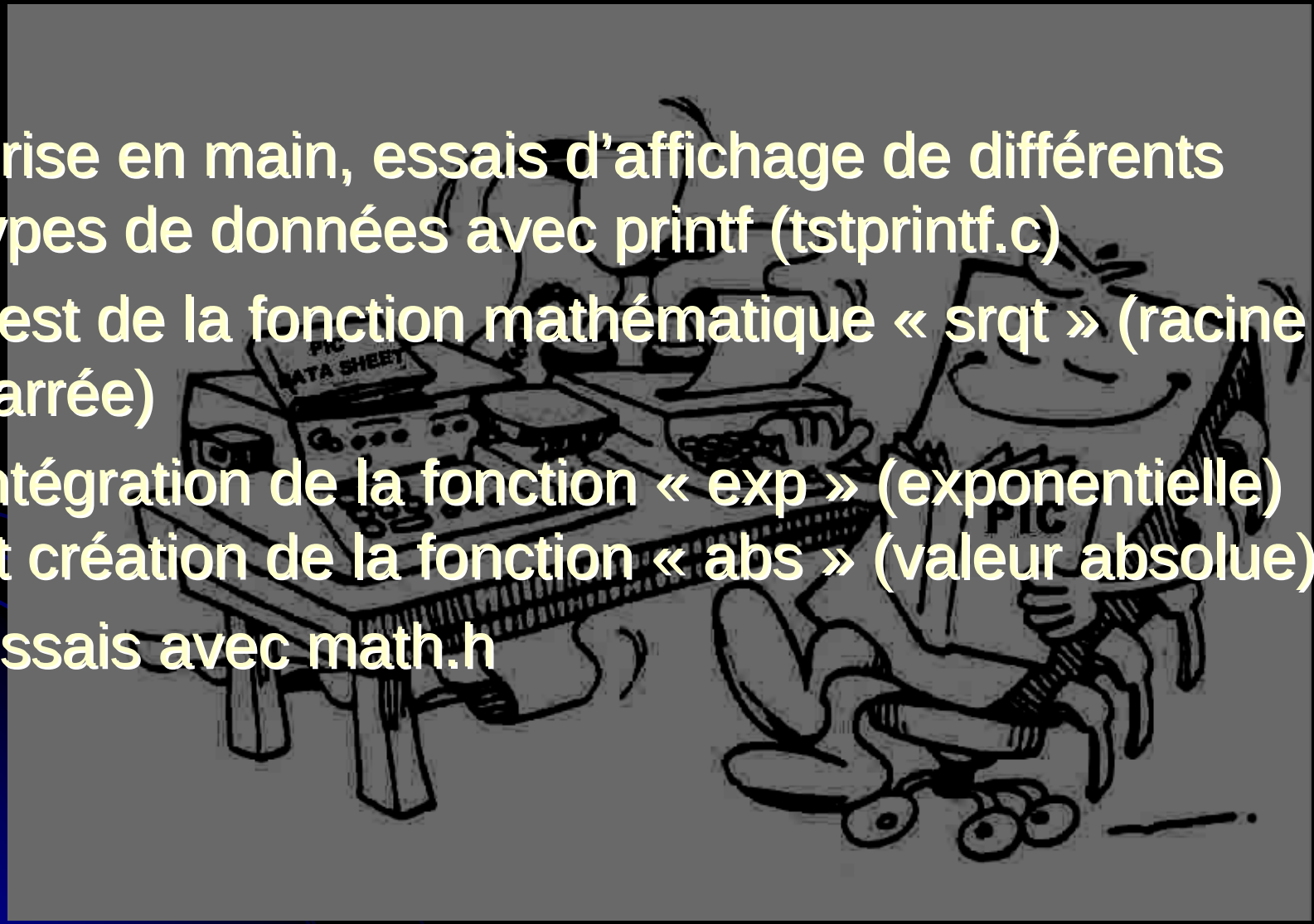


Librairie mathématique math.h

```
#include <p18f452.h>
#include <math.h>
float calcul, angle;
void main(void)
{
angle=45.0; // angle en degrees
Convertir angle en radians
calcul=sin( angle);
…
}
```

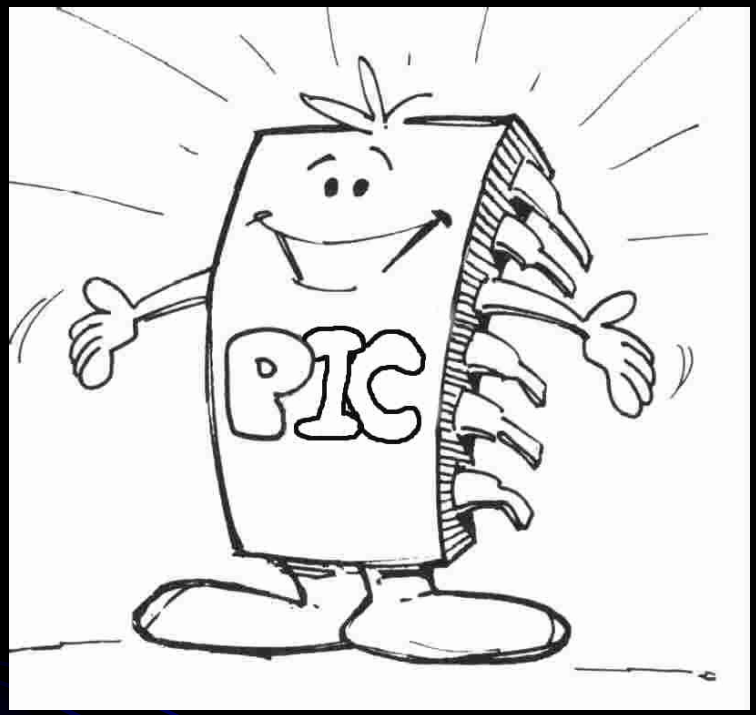


Exercices sur librairies



- Prise en main, essais d'affichage de différents types de données avec printf (tstprintf.c)
- Test de la fonction mathématique « srqt » (racine carrée)
- Intégration de la fonction « exp » (exponentielle) et création de la fonction « abs » (valeur absolue)
- Essais avec math.h

FIN DE LA PREMIERE JOURNEE



C18

Compilateur C pour PIC 18

www.microchip.com

Deuxième journée

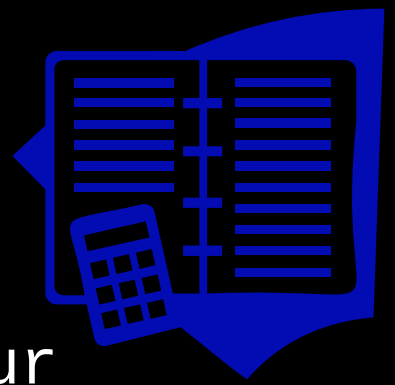
CD-RT Equipe de formation PIC
Académie d'Aix-Marseille

Spécificités du compilateur

MACROS

Instruction Macro	Action
Nop()	Executes a no operation (NOP)
ClrWdt()	Clears the watchdog timer (CLRWDT)
Sleep()	Executes a SLEEP instruction
Reset()	Executes a device reset (RESET)
Rlcf(*var, dest, access*)2,3	Rotates *var* to the left through the carry bit.
Rlncf(*var, dest, access*)2,3	Rotates *var* to the left without going through thecarry bit
Rrcf(*var, dest, access*)2,3	Rotates *var* to the right through the carry bit
Rrncf(*var, dest, access*)2,3	Rotates *var* to the right without going through thecarry bit
Swapf(*var, dest, access*)2,3	Swaps the upper and lower nibble of *var*

Assembleur en ligne



```
_asm
//Code assembleur utilisateur
// Place 10 dans la variable
compte
MOVLW 10
MOVWF compte, 0
//boucle jusqu’à 0
debut:
DECFSZ compte, 1, 0
GOTO fin
BRA debut
fin:
_endasm
```

Directives de gestion de la mémoire

#PRAGMA « SECTIONTYPE »

code ***: #pragma code toto=0x1000***

Contient des instructions exécutables

romdata ***: #pragma romdata titi=0x2000***

Contient des constantes et des variables

udata : ***#pragma udata mes_data=0x100***

Contient des variables statiques non initialisées

idata ***: #pragma idata pas_zero=0x200***

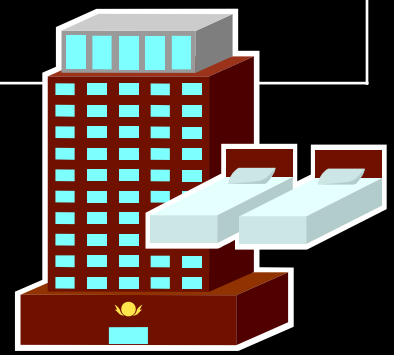
Contient des variables statiques non initialisées

Access : Zone access ram 00h à 7Fh

Overlay : Permet de localiser plusieurs sections à la même adresse physique.

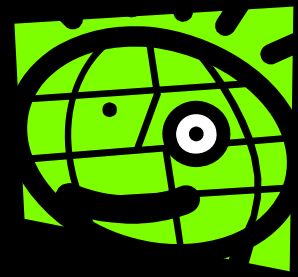
Emplacements mémoire

	rom	**ram**
far	N'importe ou dans la mémoire programme	N'importe ou dans la mémoire donnée (par défaut)
near	Dans la mémoire programme avec des adresses inférieures à 64KO	Dans access memory



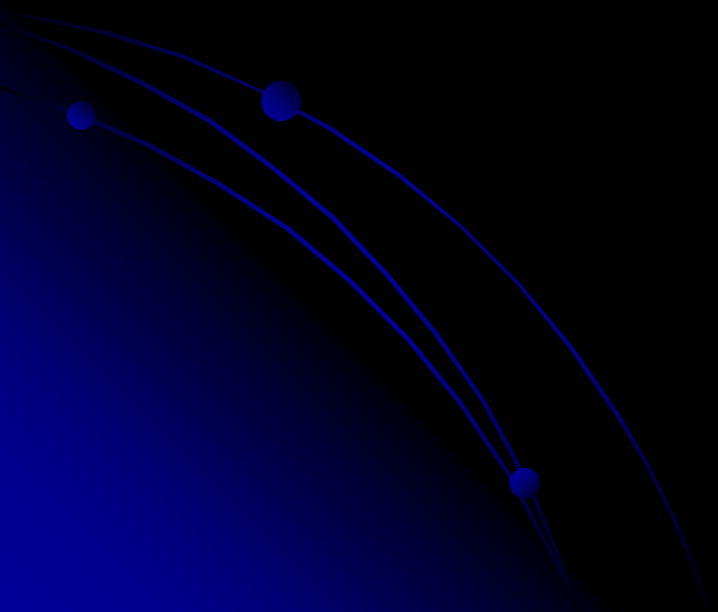
Fichier *.map

Name	Address	Location	Storage	File
c	0x000128	program	extern	
d	0x000129	program	extern	
main	0x0000f6	program	extern	
r	0x00012b	program	extern	
s	0x00012d	program	extern	
a	0x00008f	data	extern	
b	0x000090	data	extern	
f	0x00008e	data	static	
p	0x00008a	data	extern	
q	0x00008c	data	extern	



Project ⇨Build Option ⇨Project → MPLINK linker et activer « Generate map file »

Gestion mémoire

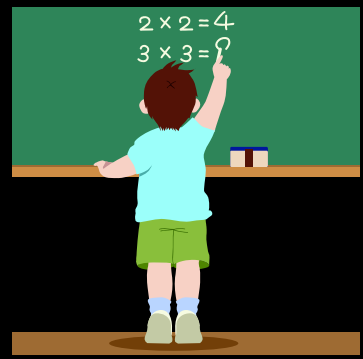


Exemple 1 : Qualificatifs de mémorisation

```
ram char a=0;                // a est en ram (le mot ram est facultatif
const ram char b=5;          // b est en ram mais ne peut être modifiée
rom char c=6;                // c est en rom et est modifialble si rom flash
const rom d=7;               // d est en rom et non modifialble

char *p;                     // p est en ram et pointe en ram
rom char *q;                 // q pointe an rom
char *rom r;                 // r est en rom et pointe en ram
rom char *rom s;             // s est en rom et pointe en rom
void fonction (void)
{
auto ram char e;   //e est dans la pile (ram et auto sont facultatifs)
static ram char f; // f est locale à la fonction mais à une adresse fixe
}
void main(void)
{
a=4;
c=0xAA;
}
```

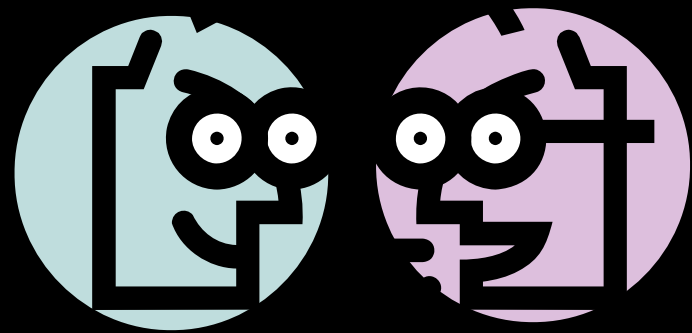
Gestion d'un tableau

```
#include <p18f452.h>
#define tbltaille 16          // taille de la table
rom const unsigned char sortie[]=
{0b00000000,0b00000001,0b00000010,0b00000100,0b00001000,0b00000100,0b00000010,0b00000001,0b00000000,0b00000001,0b00000011,0b00000111,0b00001111,0b00001110,0b00001100,0b00001000 };
void wait(int cnt)
{       for (;cnt>0; cnt--);
}

void main(void)
{       char c=0;
        TRISB = 0;                                  // PB = sortie
        while(1)                                    // boucle infinie
        { for(c=0;c<tbltaille;c++)
                        {       PORTB=sortie[c];    // c indexe la table
                                wait(5000);
                        }
        }
}
```



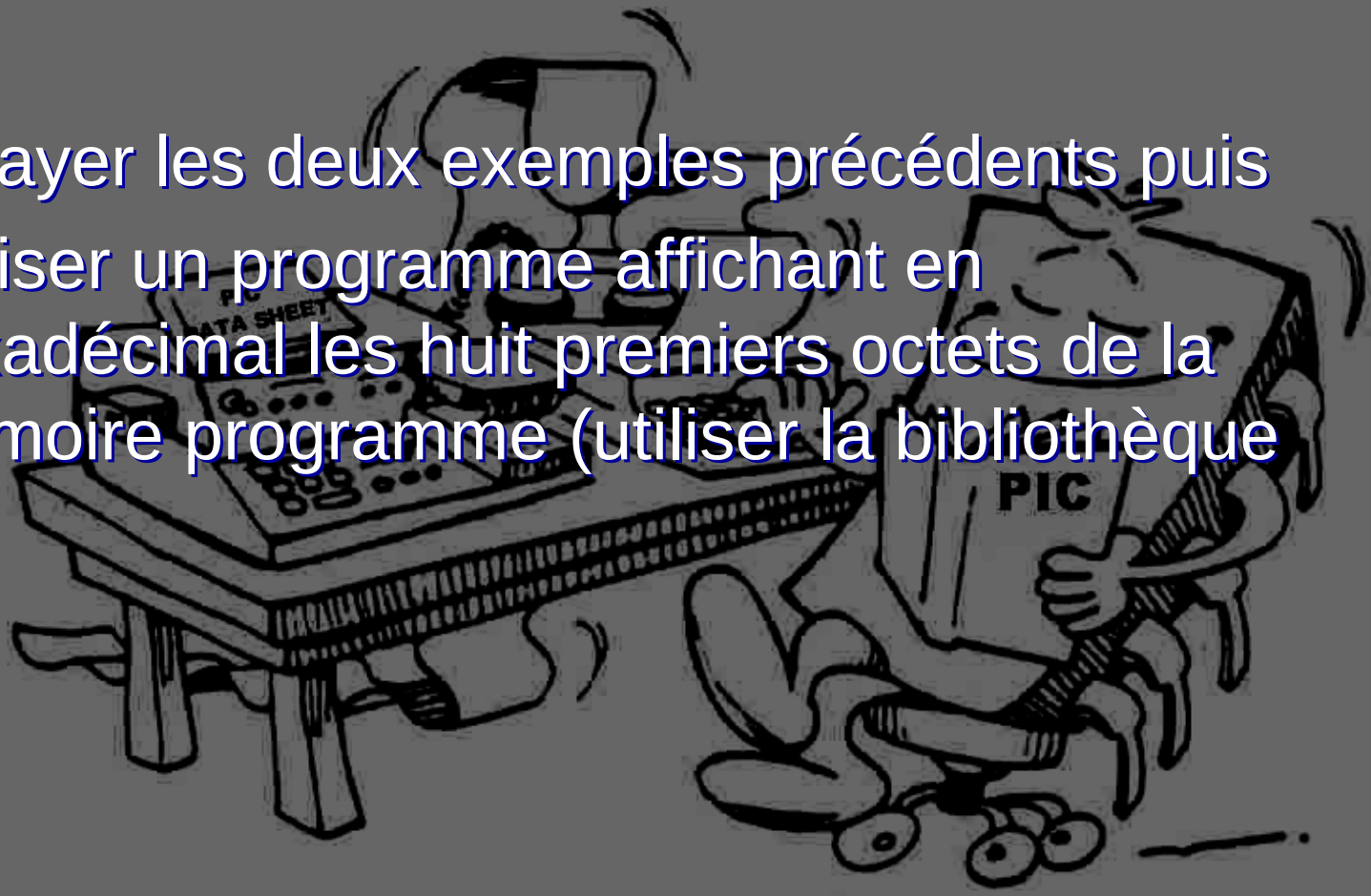
Copie ROM/RAM

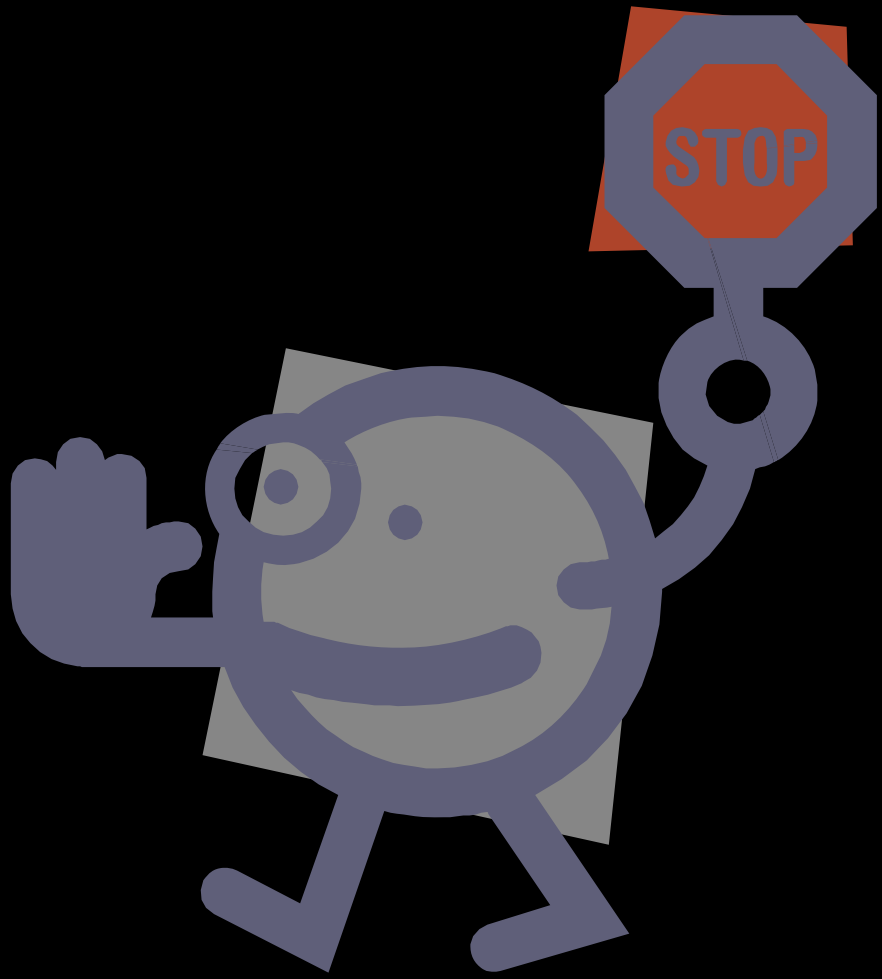
```
#include <p18f452.h>
#pragma romdata mm=0x1000
rom unsigned char chaine1[]="bonjour",*ptr1;
#pragma udata my_data=0x300
unsigned char *ptr2,chaine2[20];

void copyRom2Ram(rom unsigned char *Romptr,unsigned char *Ramptr)
{
while(*Romptr)
  {
        *Ramptr++=*Romptr++;
  };
}
void main(void)
{
        ptr1=chaine1;
        ptr2=chaine2;
        copyRom2Ram(ptr1,ptr2);
}
```



Gestion mémoire : Exercice

Essayer les deux exemples précédents puis réaliser un programme affichant en hexadécimal les huit premiers octets de la mémoire programme (utiliser la bibliothèque lcd)





INTERRUPTIONS

Le problème des INTERRUPTIONS

- **Le C ne sait pas traiter les sous programmes d'interruption. Ceux-ci se terminent par l'instruction assembleur RETFIE et non par RETURN**

- **Le C ne laisse pas le contrôle des adresses au programmeur. Les interruptions renvoient à une adresse fixée par MICROCHIP (0x08 ou 0x18)**

#PRAGMA et interruptions

#pragma interruptlow *it_lente*
Déclaration d'une fonction en tant que programme de traitement d'interruption **non prioritaire (vect=0x18)**

#pragma interrupt *it_rapide*
Déclaration d'une fonction en tant que programme de traitement d'interruption **prioritaire(vect=0x08)**

#pragma code monprog=0x08 force le compilateur à placer le code en 0x08

#pragma code redonne toute liberté au compilateur

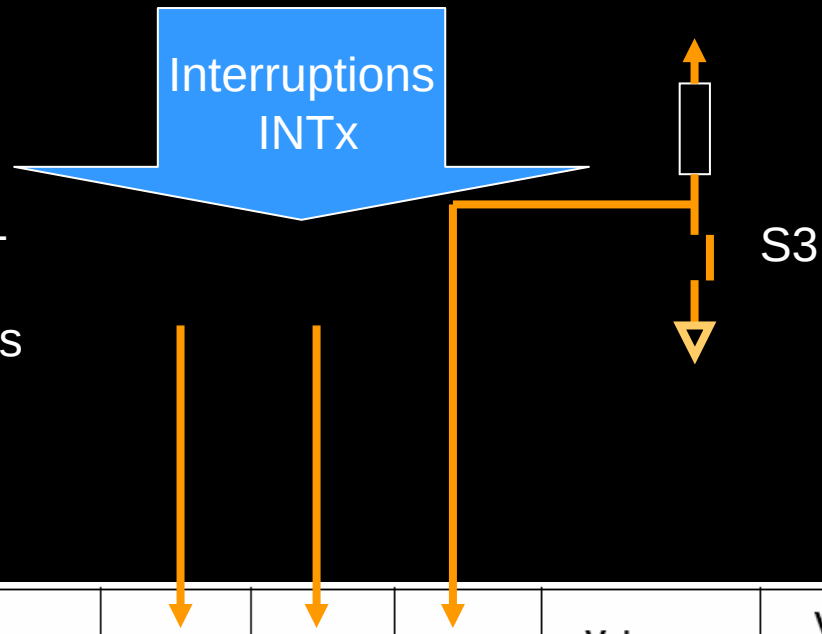
Les priorités d'IT sur PIC18

- Si le bit IPEN(RCON)=1 (0 par défaut) les priorités d'interruptions sont activées.

- Si IPEN=0; GIE=1 active toutes les interruptions autorisés
- Si IPEN=1; GIEH=1 active les interruptions prioritaires et GIEL=1 les interruptions non prioritaires. Les registres PIR permettent de définir les priorités (prioritaires pas défaut).

Interruptions sur le PORTB

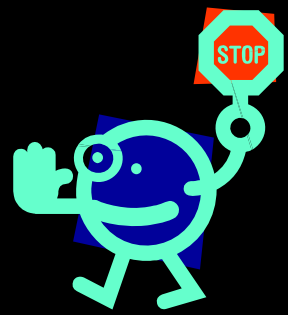
Activer les interruptions sur PRB0:
Autoriser les IT sur PRB0: INTOIE=1
Autoriser toutes les interruptions: GIE=1
Lors de l'interruption :
Placer le code à exécuter dans une sous programme d'interruption en 0x08
Avant le retour d'IT effacer le drapeau : INT0IF=0



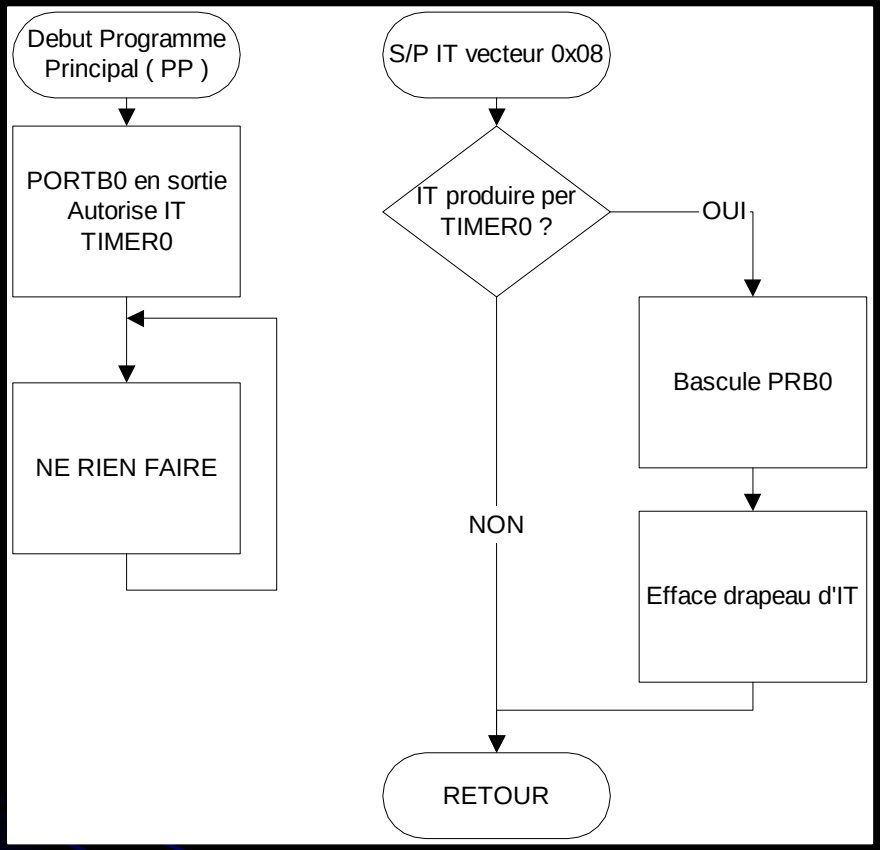
Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other RESETS
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
LATB	LATB Data Output Register								xxxx xxxx	uuuu uuuu
TRISB	PORTB Data Direction Register								1111 1111	1111 1111
INTCON	GIE/ GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
INTCON2	RBPU	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP	1111 -1-1	1111 -1-1
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	11-0 0-00	11-0 0-00

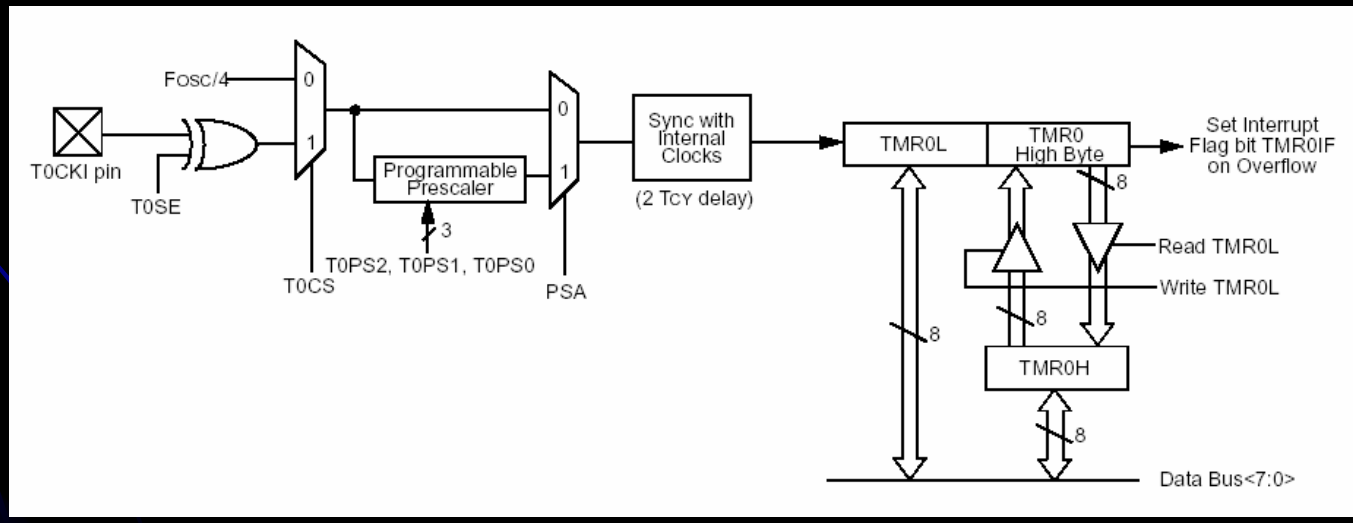
Exemple : demo_it_prb0.c

```
#include <p18f452.h>
unsigned int cpt=0;       // compteur d'interruption

#pragma interrupt it_sur_rb0
void it_sur_rb0(void)
{
                if (INTCONbits.INT0IF)
                                {
                                cpt++;
                                INTCONbits.INT0IF=0;
                                }
}

#pragma code vecteur_d_IT=0x08
void une_fonction(void)
{
_asm
   goto it_sur_rb0
_endasm
}
#pragma code

void main (void)
{
                TRISBbits.TRISB0=1;
                INTCONbits.INT0IE=1;
                INTCONbits.GIE=1;
                while(1);
}
```



Interruptions TIMER 0

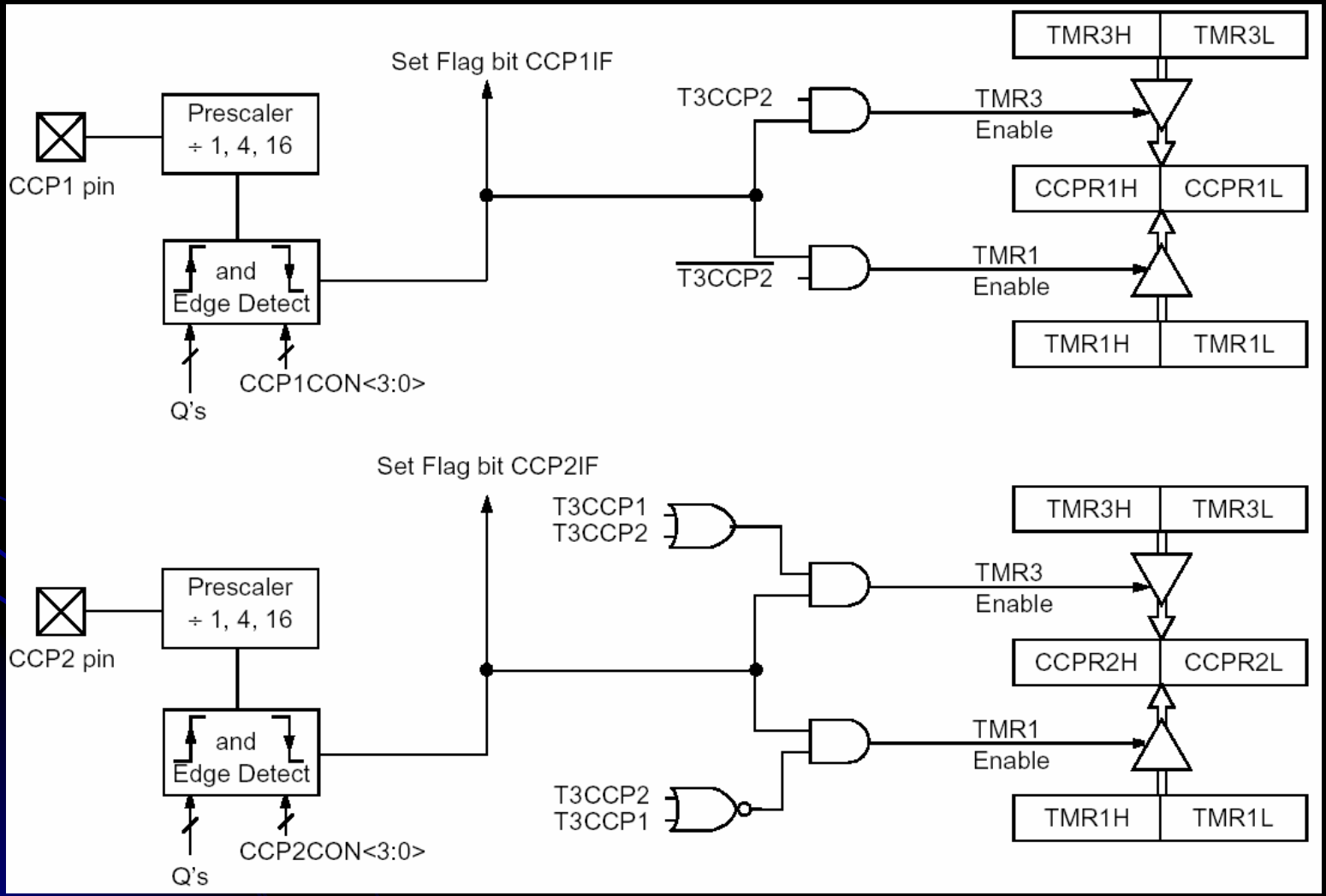




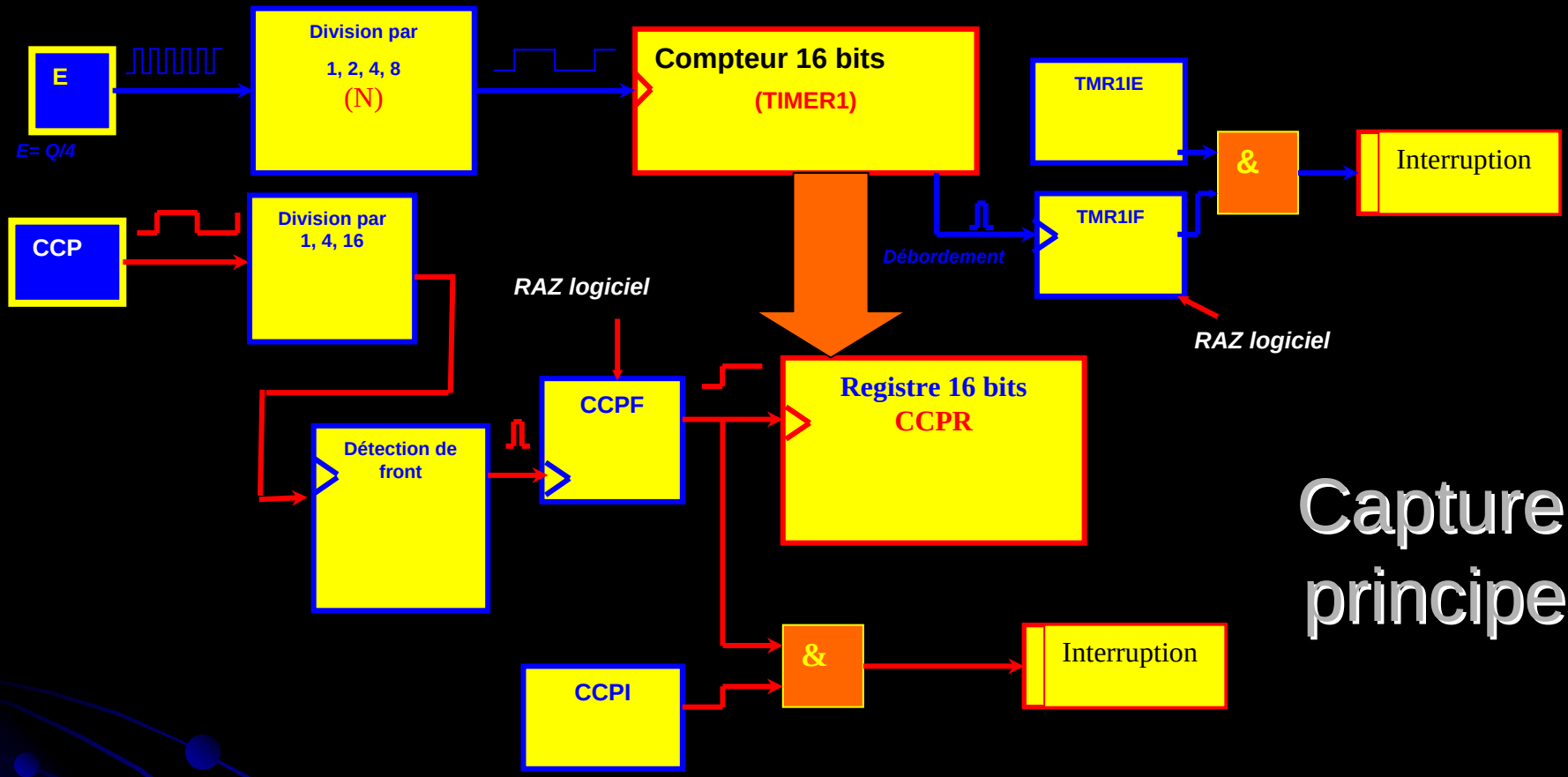
Interruptions

```
#include <p18f452.h>
void traiteIT(void);    // prototype, traiteIT est définit après son appel
#pragma code it=0x08
void saut_sur_spIT(void)
{    _asm
            goto traiteIT
     _endasm
}
#pragma code

#pragma interrupt traiteIT
void traiteIT(void)
{    if(INTCONbits.TMR0IF)     //vérifie un débordement sur TMR0
                        {    INTCONbits.TMR0IF = 0;   //efface le drapeau d'IT
                             PORTBbits.RB0 = !PORTBbits.RB0;  //bascule LED sur RB0
                        }
}
void main()
{
     PORTBbits.RB0 = 0;          // port B0 en sortie
     TRISBbits.TRISB0 = 0;       // et à 0
     T0CON = 0x82;                              //Active le TIMER0 - prescaler 1:8
     INTCONbits.TMR0IE = 1;                     //Autorise interruption sur TMR0
     INTCONbits.GIEH = 1;                       //autorise toutes les IT démasquées
     while(1);                                  // une boucle infinie, tout fonctionne en IT
}
```

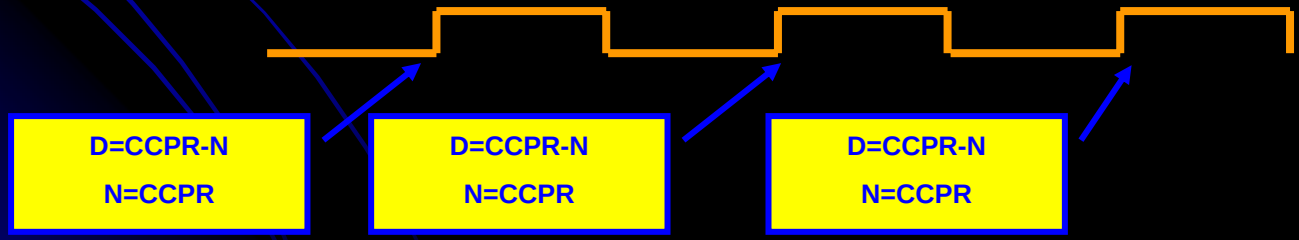
CAPTURE


Set Flag bit CCP1IF
CCP1 pin
Prescaler
÷ 1, 4, 16
and
Edge Detect
Q's
CCP1CON<3:0>
T3CCP2
T3CCP2
TMR3H
TMR3L
TMR3
Enable
CCPR1H
CCPR1L
TMR1
Enable
TMR1H
TMR1L
Set Flag bit CCP2IF
CCP2 pin
Prescaler
÷ 1, 4, 16
and
Edge Detect
Q's
CCP2CON<3:0>
T3CCP1
T3CCP2
T3CCP2
T3CCP1
TMR3H
TMR3L
TMR3
Enable
CCPR2H
CCPR2L
TMR1
Enable
TMR1H
TMR1L

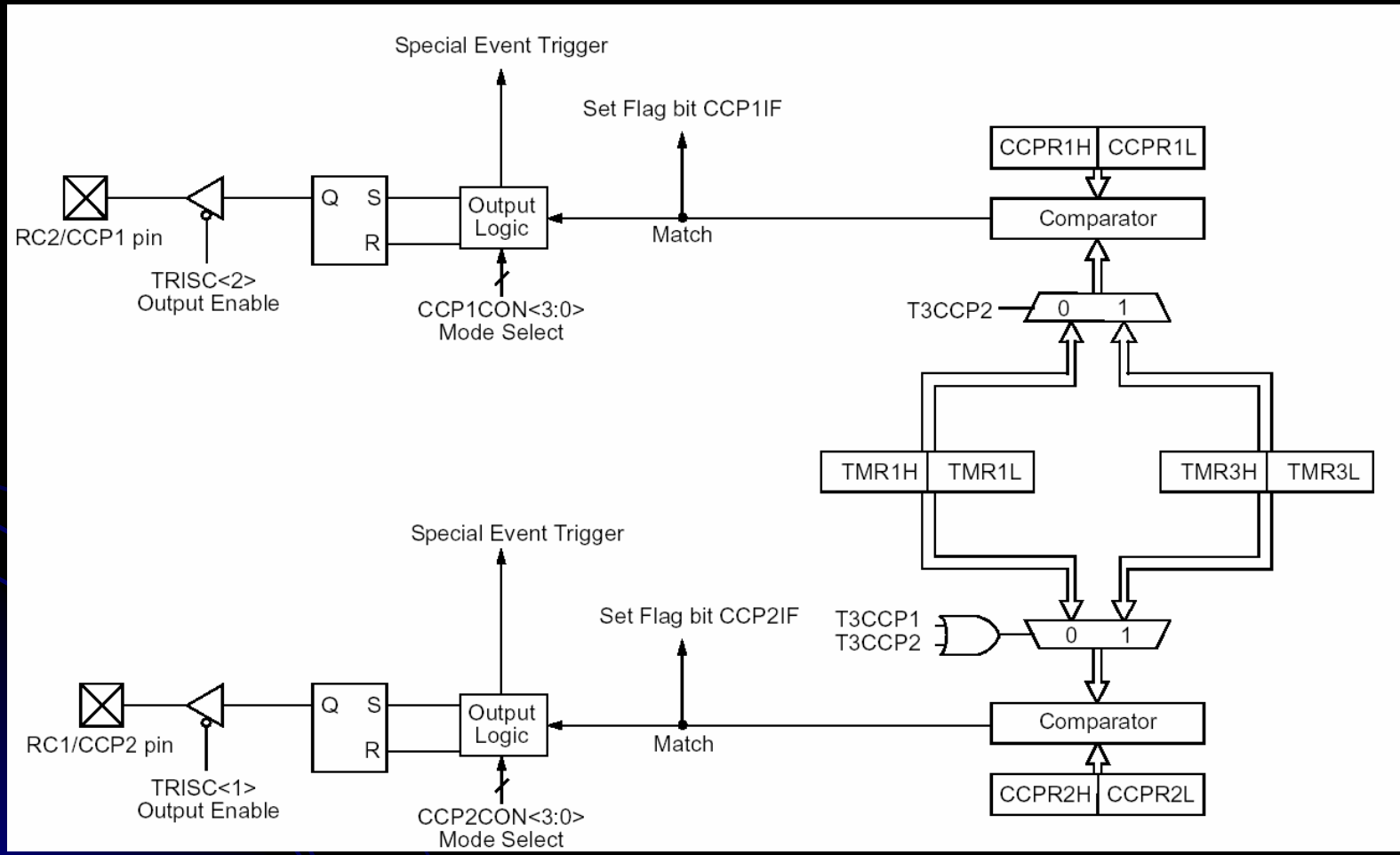
Capture - principes



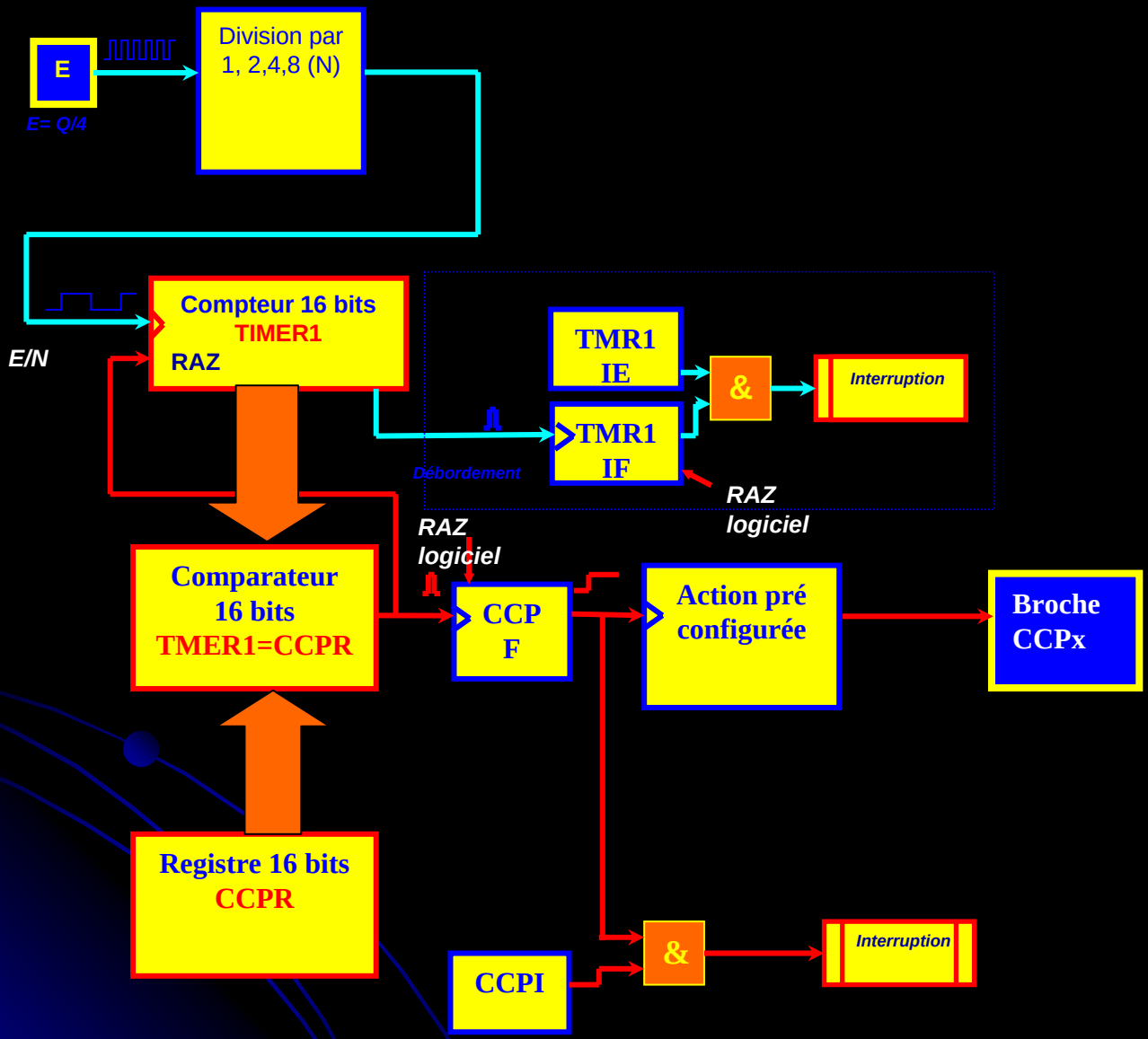
T=D*E/N



COMPARE

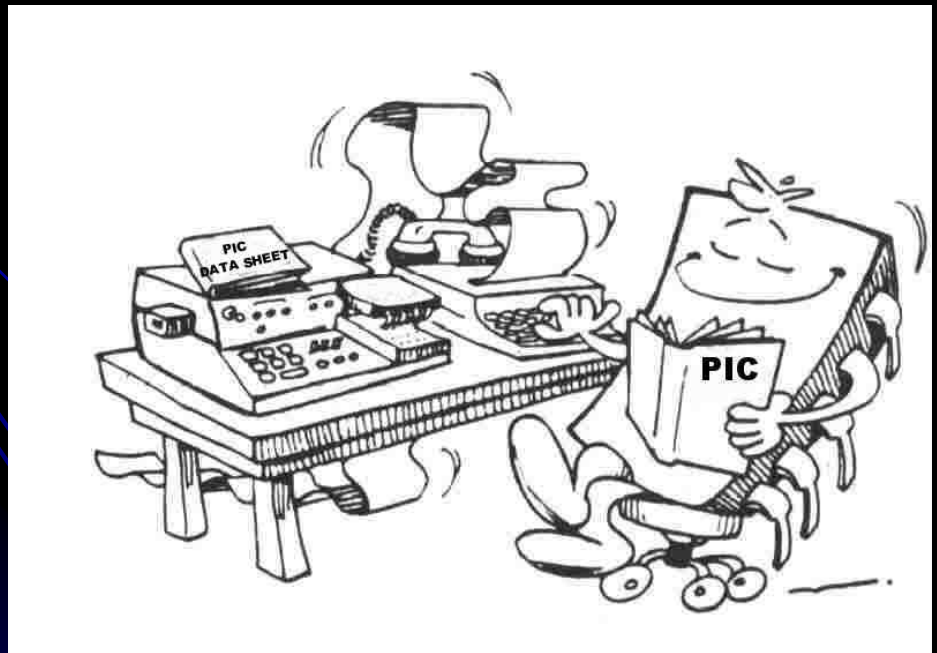

Special Event Trigger
Set Flag bit CCP1IF
CCPR1H
CCPR1L
Q
S
R
Output Logic
Match
Comparator
RC2/CCP1 pin
TRISC<2>
Output Enable
CCP1CON<3:0>
Mode Select
T3CCP2
0
1
TMR1H
TMR1L
TMR3H
TMR3L
Special Event Trigger
Set Flag bit CCP2IF
T3CCP1
T3CCP2
0
1
Q
S
R
Output Logic
Match
Comparator
RC1/CCP2 pin
TRISC<1>
Output Enable
CCP2CON<3:0>
Mode Select
CCPR2H
CCPR2L

COMPARE - Principes



Exercices avec interruptions

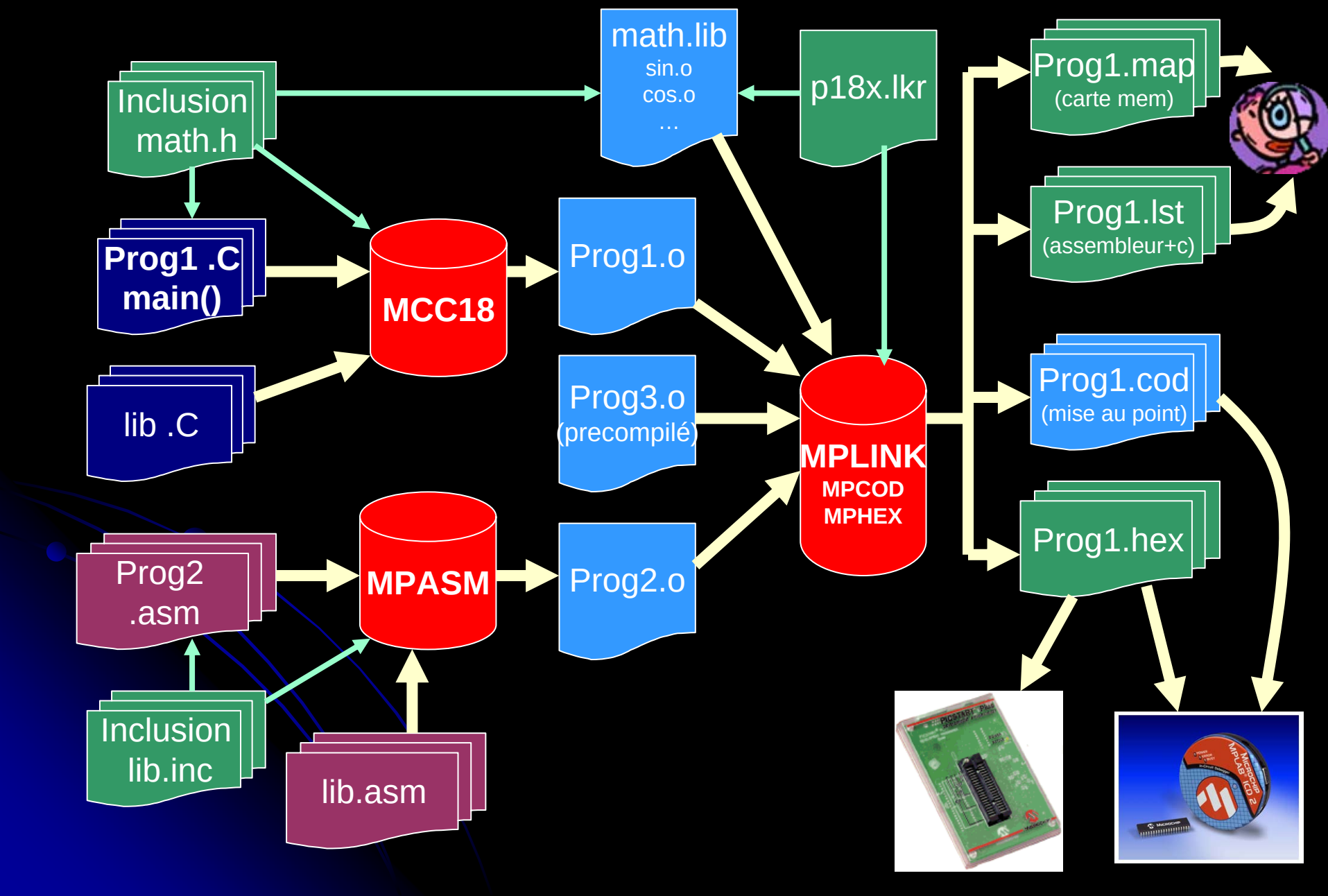
- **Production de temporisation**
- **Modification de rapport**
- **Mesure de durée (fréquencemètre)**
- **Réalisation d'une horloge temps réel**



Création et gestion des

BIBLIOTHEQUES

Flux des données



Librairie libpd2.h

LIBRAIRIE USART HARD (sur PIC équipés de cette fonction)	
void initsci(void)	Configuration BAUD et interruption (9800,n,8,1)
char getsci(void)	Retourne le premier caractère reçu sur SCI
void putsci(char c)	Emet c sur SCI
char *getstsci(char *s, char finst)	lit une chaîne de caractère sur SCI se terminant par finst la variable finst contient le caractère attendu en fin de chaîne
int putstsci(char *s)	émet la chaîne s (finit par 0)
char carUSARTdispo(void)	Cette fonction retourne 1 (vrai) si un caractère est disponible dans le buffer de réception et 0 si non, si les deux pointeurs sont identiques, il n'y a rien dans le buffer

I2C fonction HARD interne PIC18 (testée sur TC74)	
void init_i2c(void);	initialise port i2c en mode maitre
signed char lit_i2c(unsigned char adresse,unsigned char registre);	retourne l'octet de l'adresse i2c

MPLIB

Format : MPLIB [/q] /{ctdrx} NOM_LIBRAIRIE [liste de fichiers objets]

/c	Création d'une nouvelle librairie contenant les fichiers objets suivants
/t	Liste le contenu de la librairie
/d	Efface un objet de la librairie
/r	Remplace un objet existant dans la librairie et la place à la fin de la librairie
/x	Extrait un membre de la librairie
/q	Pas d'affichage du résultat

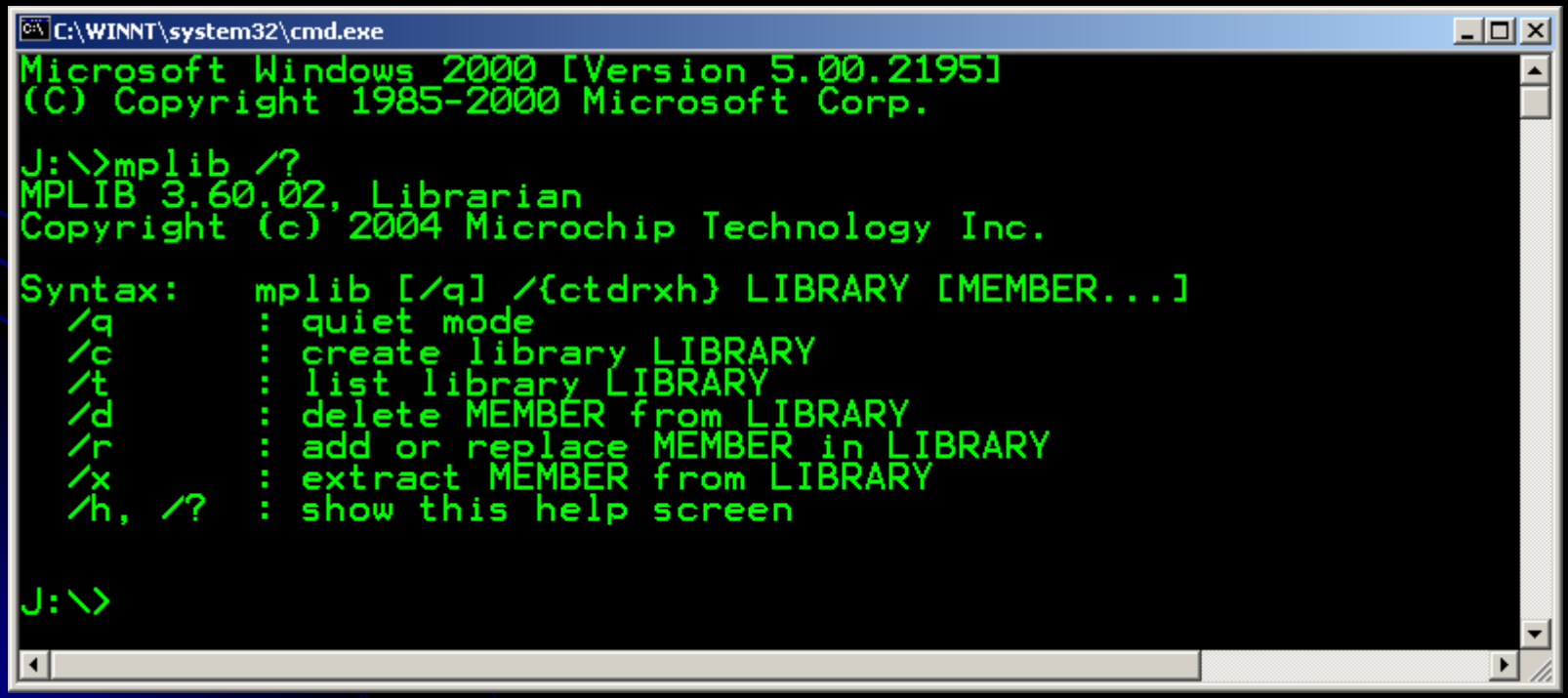
MPLIB - LANCEMENT

MPLIB est un programme « console »

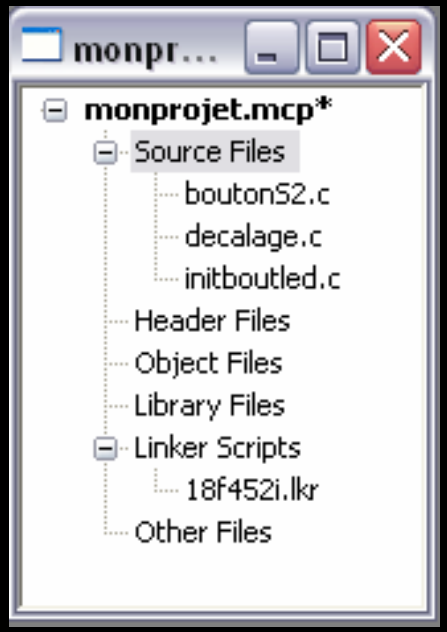
Sous WINDOWS :

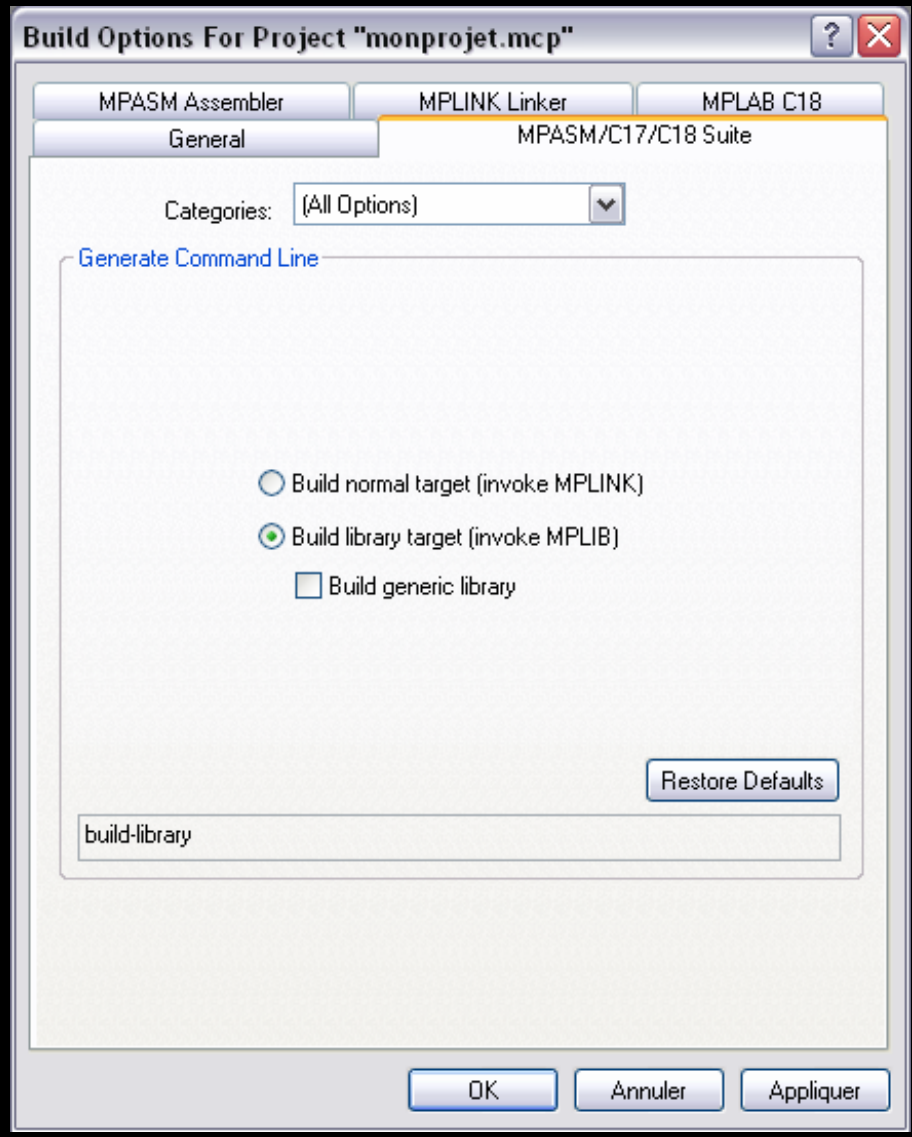
Démarrer – exécuter – cmd

Dans la fenêtre DOS tapez : MPLIB suivi des options ou MPLIB /?



Compiler directement une librairie





La librairie sera créée dans le dossier par défaut et portera le nom du projet (projet.lib)

Il restera à créer le fichier header projet.h

Utilisation d'une librairie

Les bibliothèques personnelles doivent être déclarées au compilateur

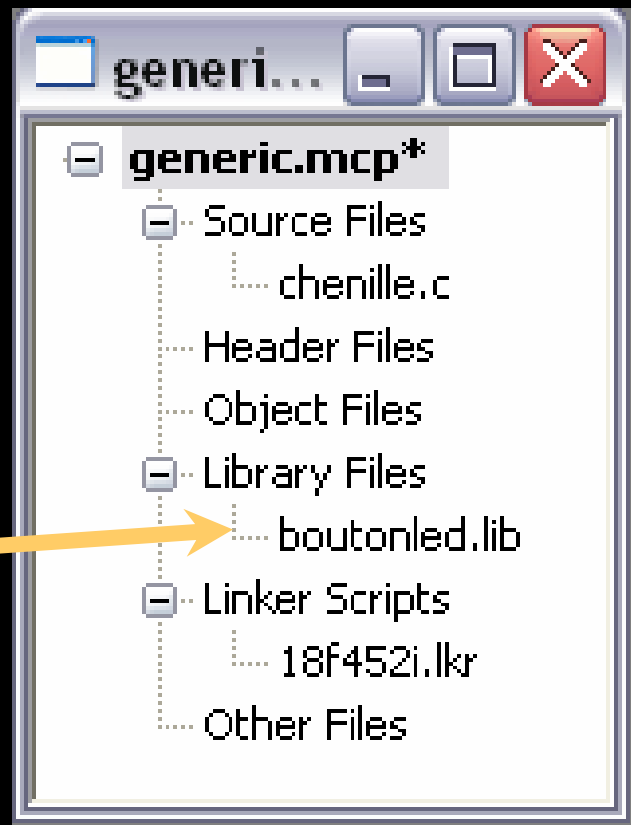
Ajouter les lignes suivantes dans p18fxx.lkr

LIBPATH .;..\persolib

FILES boutonled.lib

Ou

Inscérer la librairie dans le gestionnaire de projet MPLAB



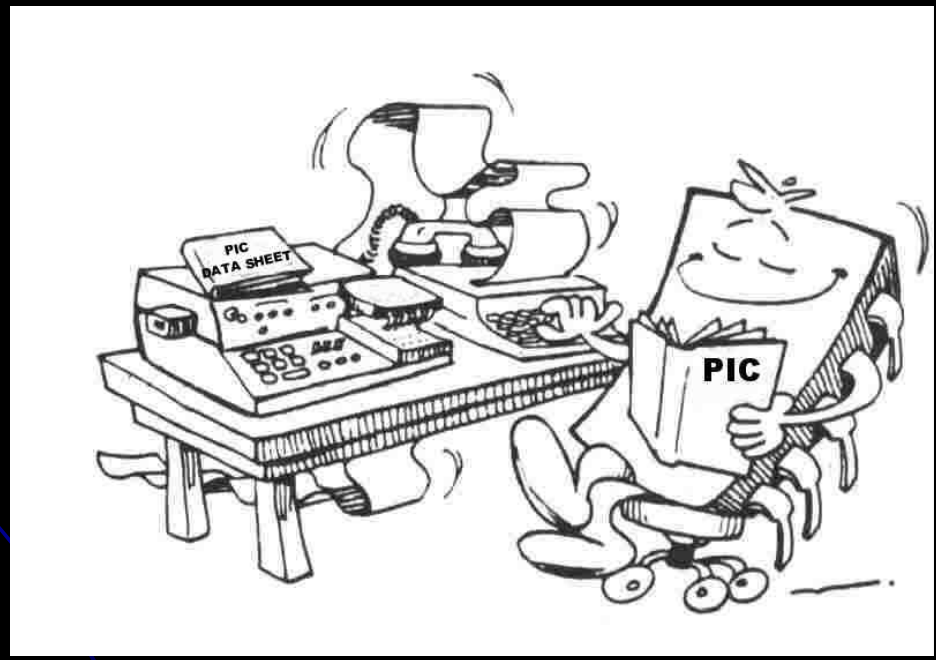
Créer une bibliothèque

- Créer la librairie boutonled.lib, la tester dans le projet, puis par insertion dans p18f452.lkr
- initboutled.c
  - Initialise les ports du PIC18
- boutons2.c
  - Attend l'enfoncement de s2

- decalage.c
  - Décale le portb à gauche ou à droite
- chenille.c
  - Programme de test de la bibliothèque

Exercices avec MPLIB

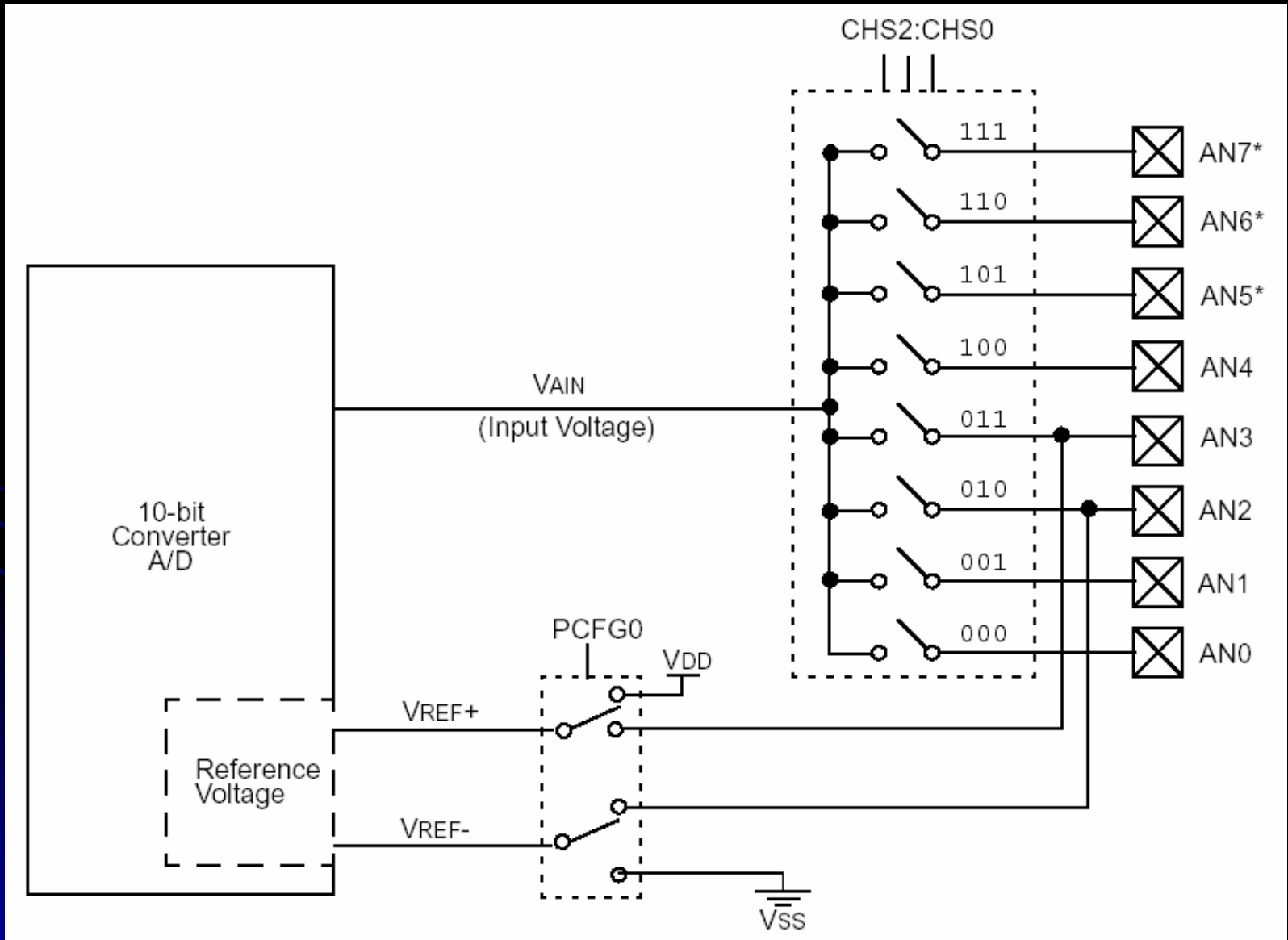
- Créer et tester la librairie « **boutonled** »



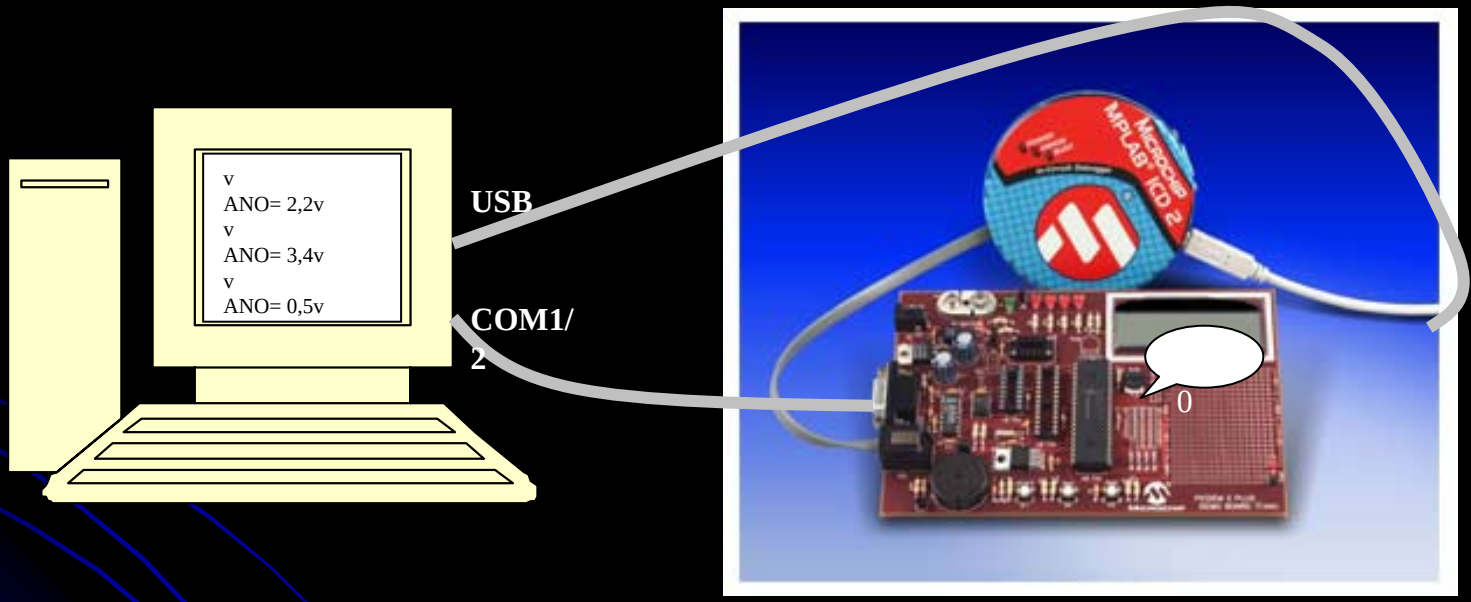
FIN DE LA DEUXIEME JOURNEE

Mise en œuvre des périphériques intégrés sur PIC18

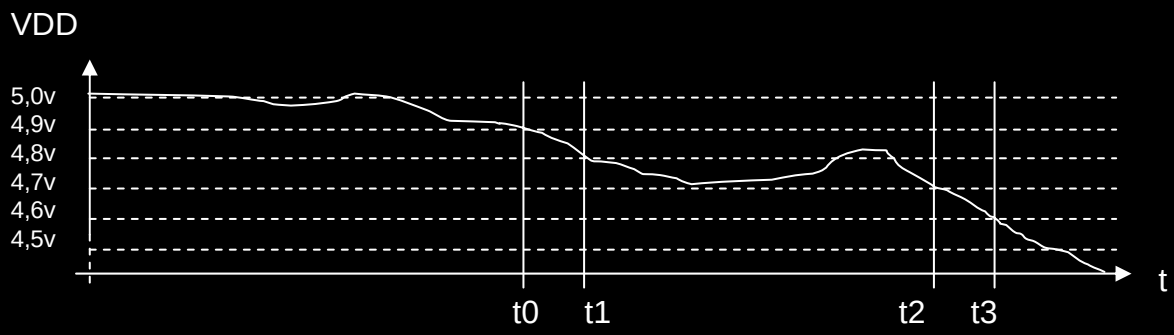
ADC 10 bits


CHS2:CHS0
111
110
101
100
011
010
001
000
AN7*
AN6*
AN5*
AN4
AN3
AN2
AN1
AN0
VAIN
(Input Voltage)
10-bit
Converter
A/D
PCFG0
VDD
VREF+
Reference
Voltage
VREF-
VSS

USART



Exercice : contrôle de batterie



VDD	temps	RB0	RB1	RB2	RB3
4,9v	t0	1	0	0	0
4,8v	t1	1	1	0	0
4,7v	t2	1	1	1	0
4,6v	t3	1	1	1	1

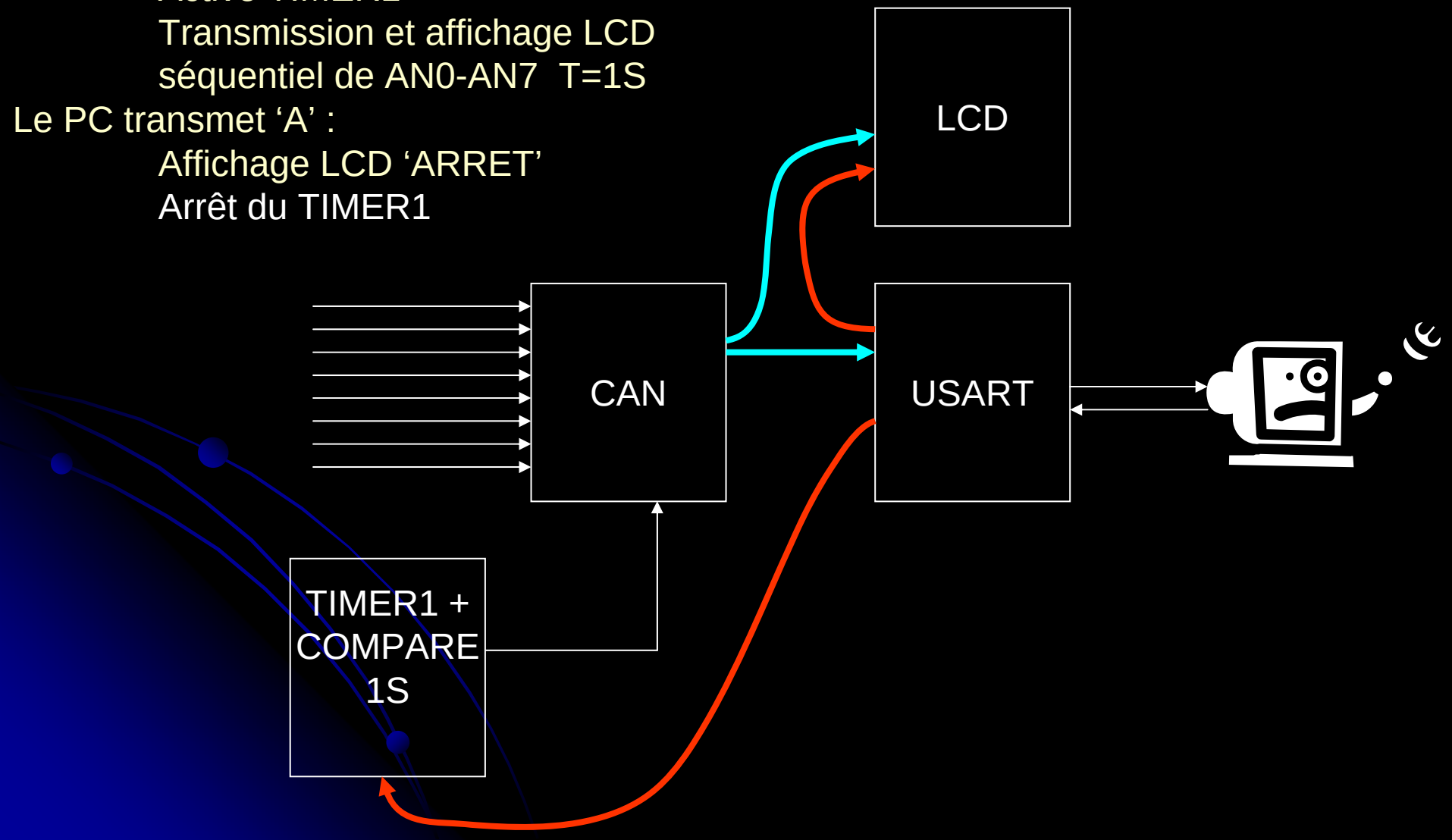
Exercice : centrale d'acquisition

Le PC transmet 'M' :

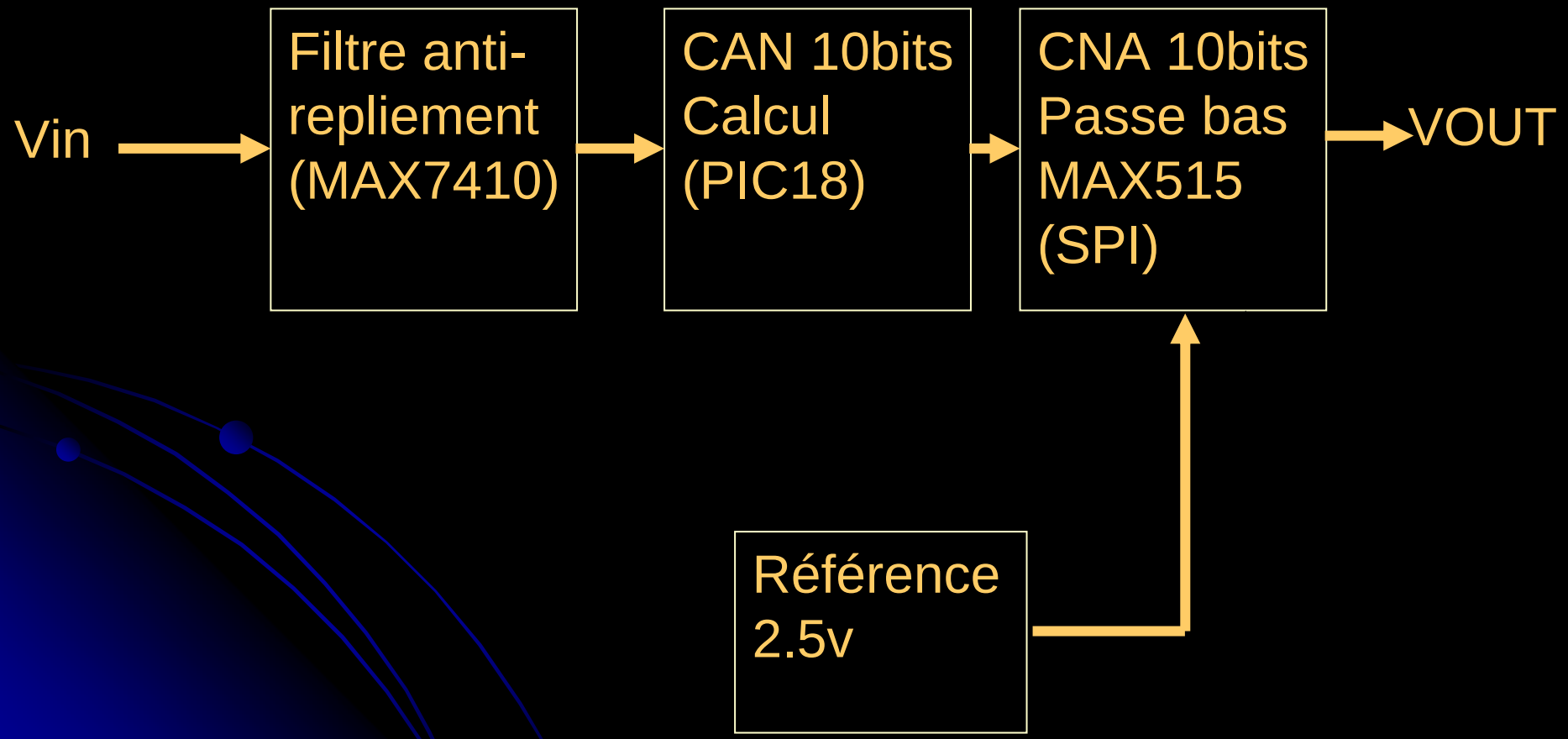
- Affichage LCD 'MESURE'
- Active TIMER1
- Transmission et affichage LCD séquentiel de AN0-AN7 T=1S

Le PC transmet 'A' :

- Affichage LCD 'ARRET'
- Arrêt du TIMER1



SPI ex: filtre numérique



Exercices sur …

- Convertisseur analogique numérique
- EEPROM interne
- Communications asynchrones
- BUS I2C
- BUS SPI

Merci pour votre attention