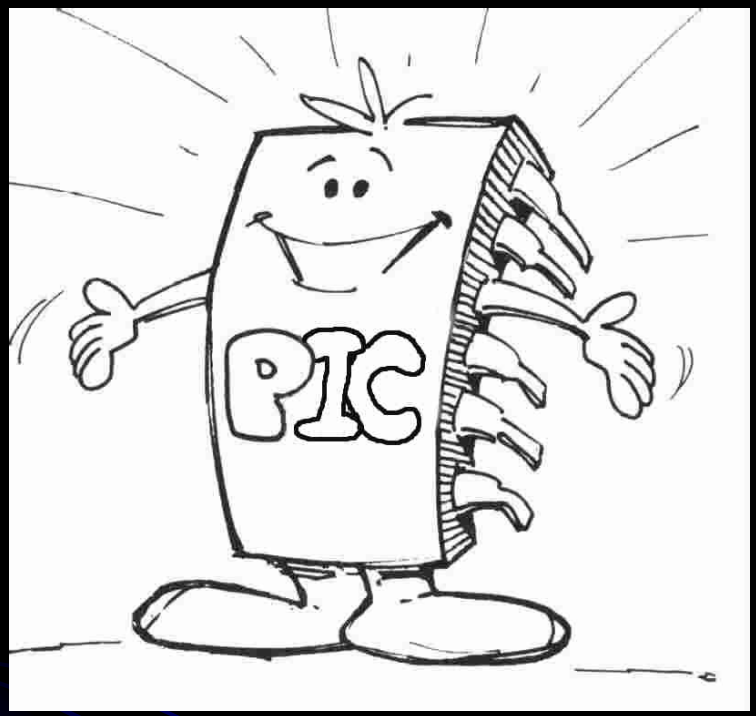


C18

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 ");
}
```

Bibliothèques en C (texte) *.c

Fichiers header *.h

Bibliothèques pré-compilées (fichiers objet)

Fichier source C contenant la fonction main

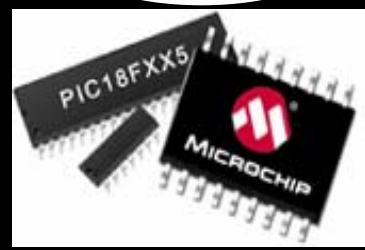
Préprocesseur
Remplace les #define et effectue les #include

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

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

- La version 2.40 de 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
- #pragma config WDT = OFF
- #pragma config LVP = OFF
- #pragma config DEBUG = ON

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 : ^

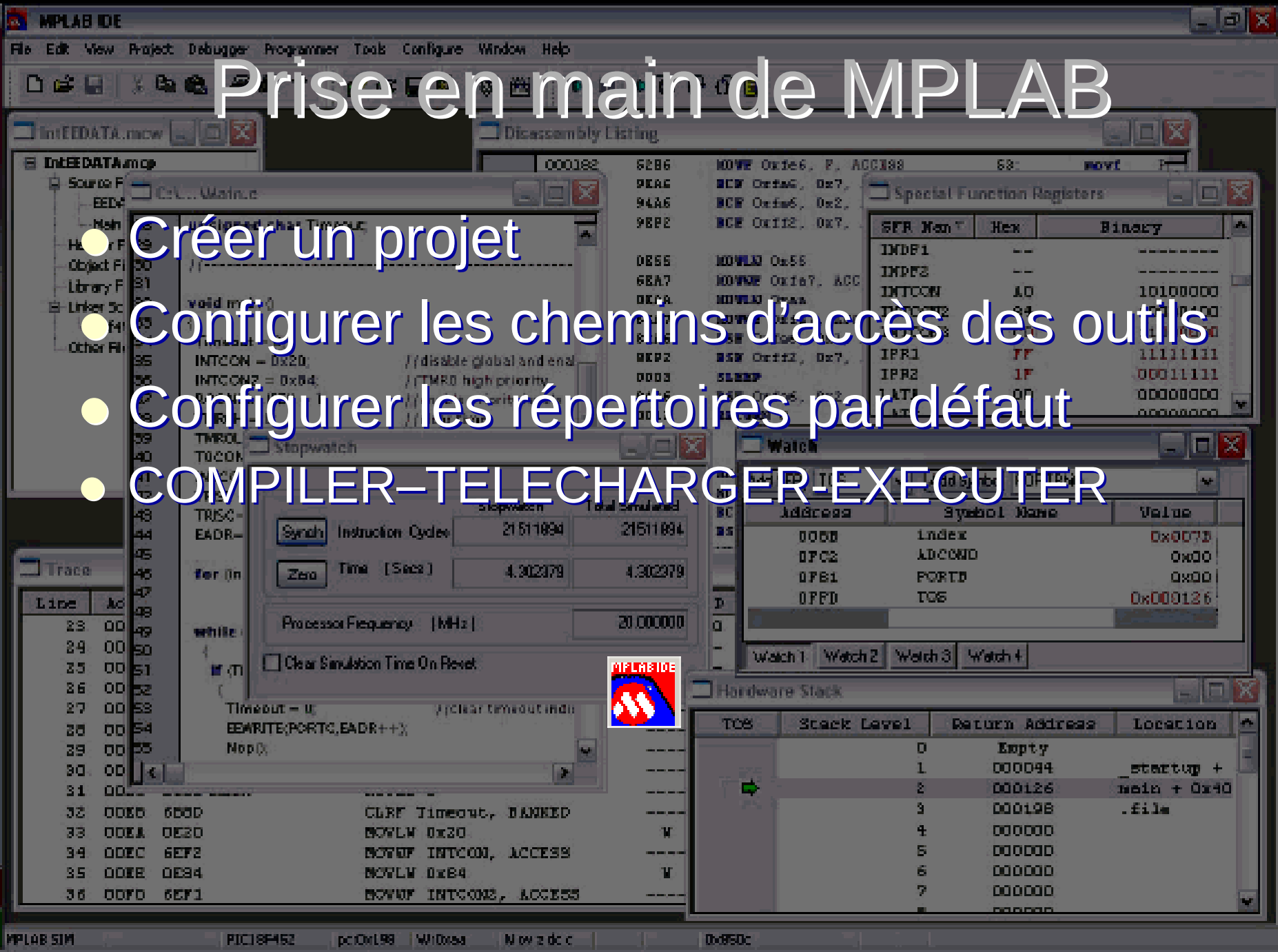
Bit1	Bit2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

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

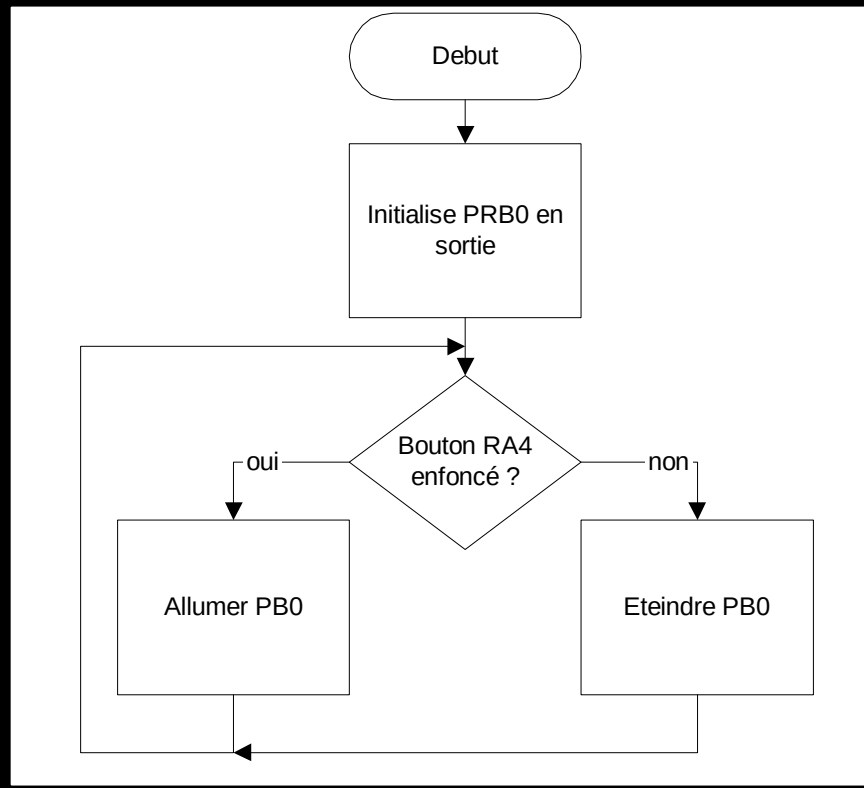
Prise en main de MPLAB

- 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

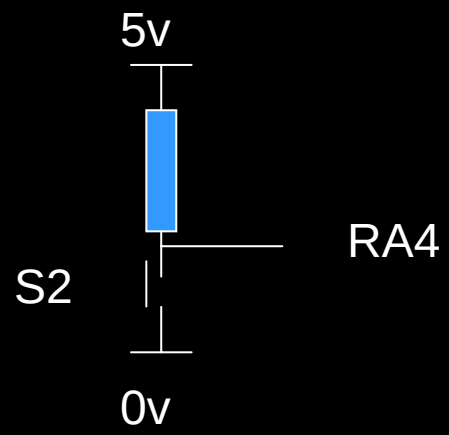
Gérer les ports parallèles

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

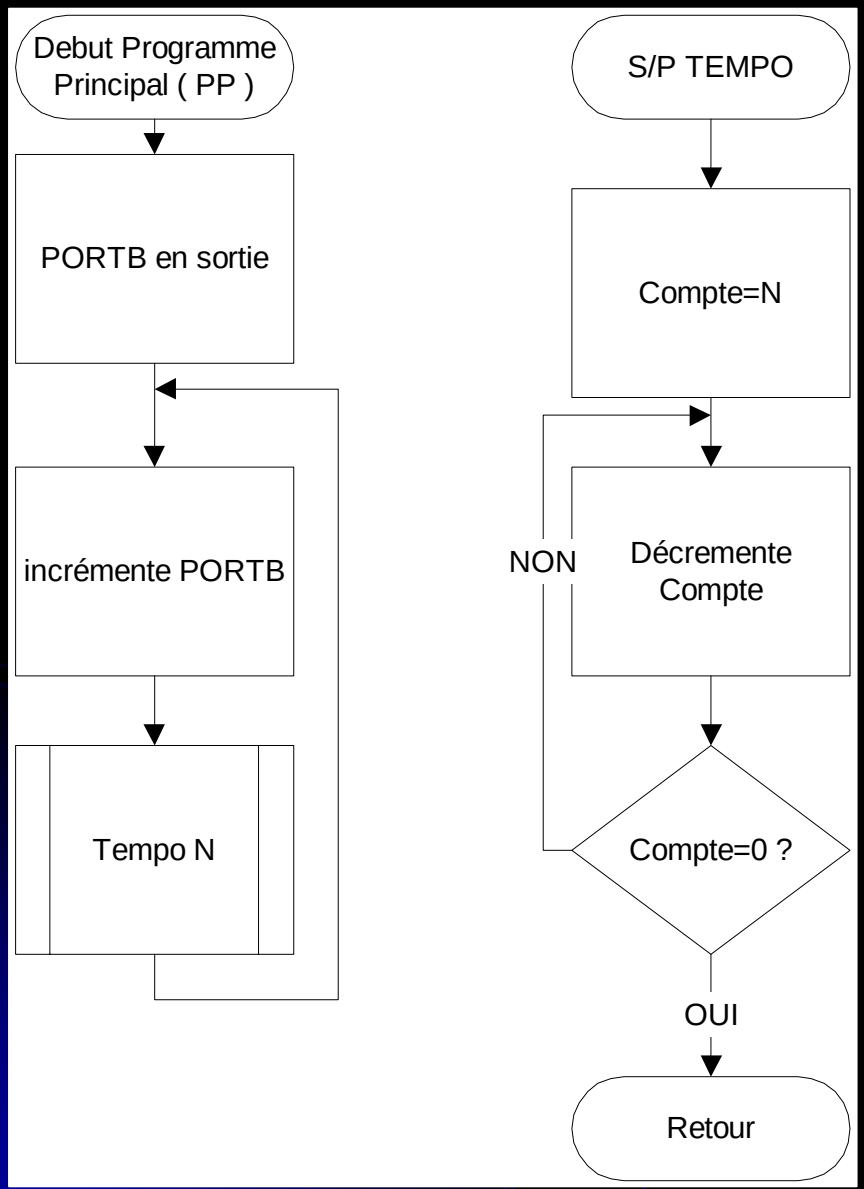
Attendre TANT QUE RA4=1 puis

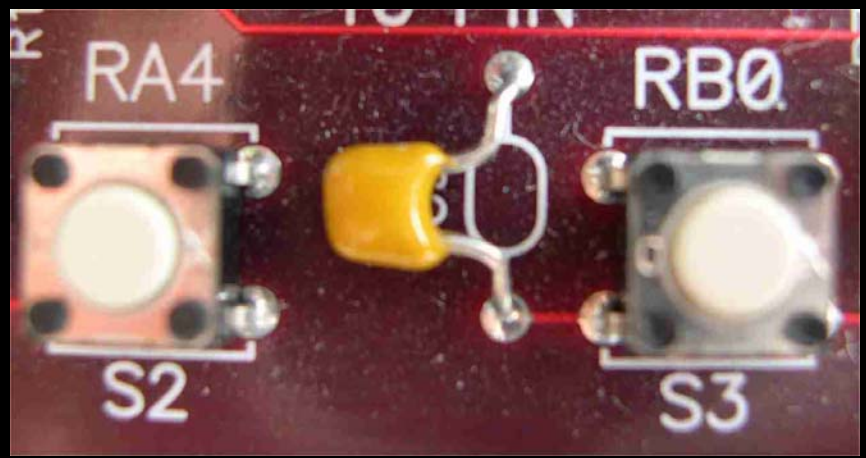
Attendre TANT QUE RA4=0

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

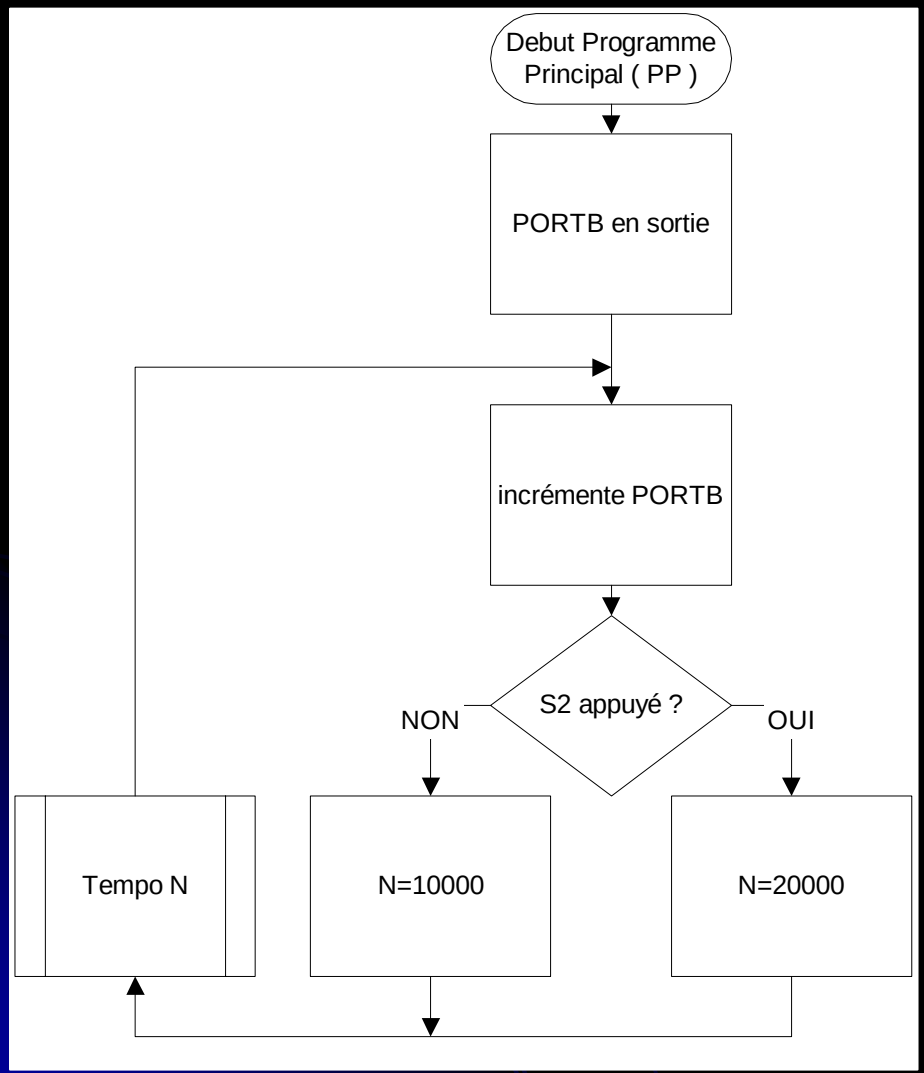


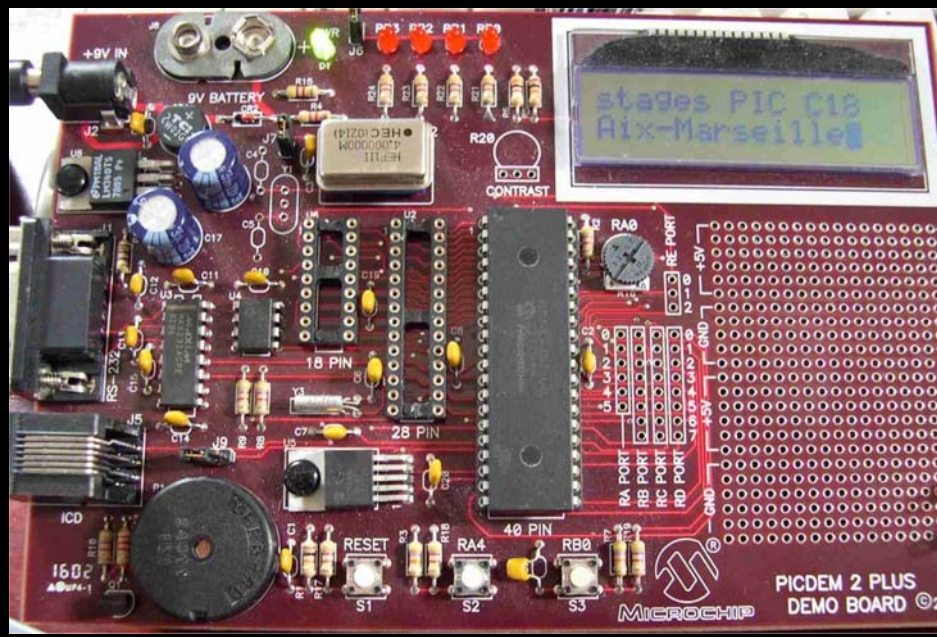
Exercice


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

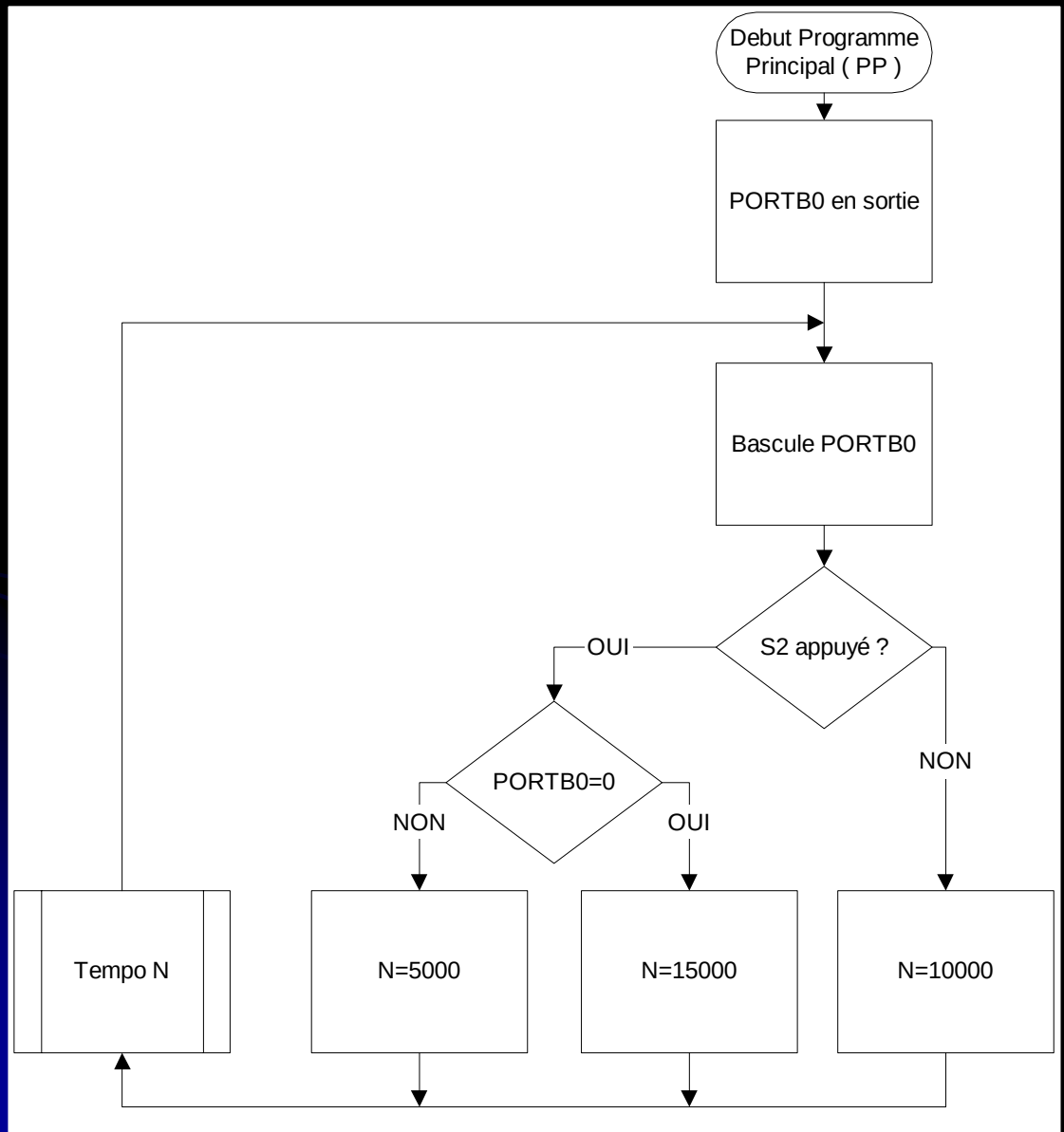

RA4
RB0
S2
S3

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
PICDEM 2 PLUS
DEMO BOARD
MICROCHIP

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

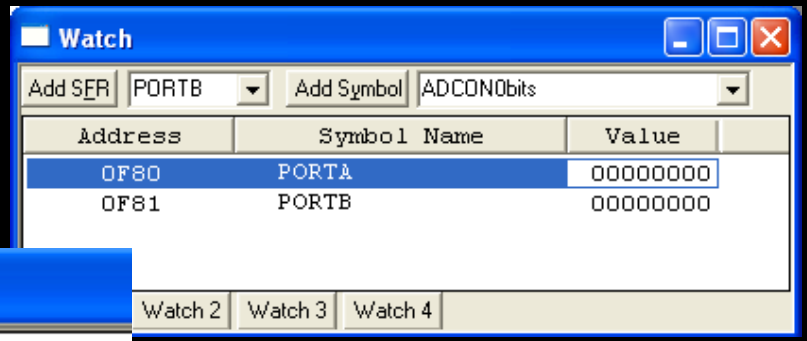
Décalages

```
#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



C:\exoPIC\exoC18\bouton.c

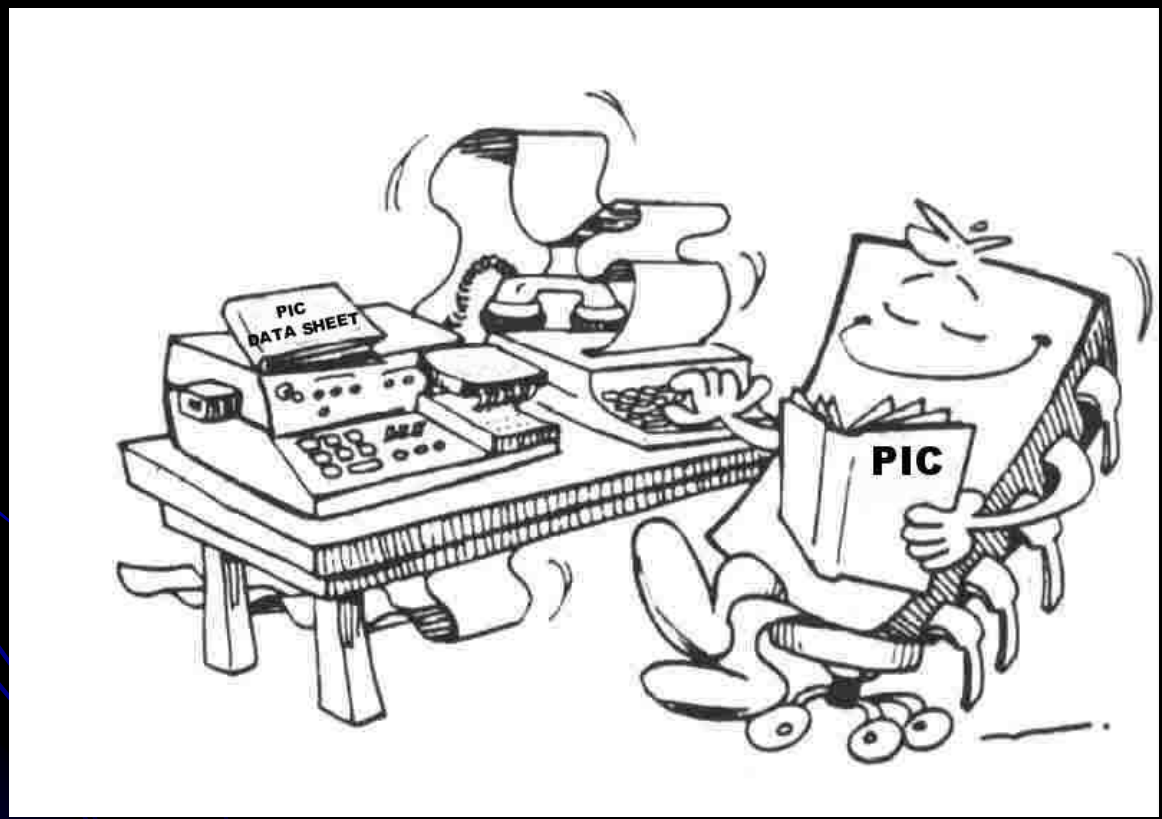
```
1   /* Bouton et LED LED sur PICDEM2+*/
2   /* La LED sur PB0 s'éteint si S2 (PA4) est enfoncé*/
3
4   #include <p18f452.h>
5
6   void main(void)
7   {    TRISA=0xFF;      // PORTA e
8   TRISB = 0;       /* PB en sorti
9        while(1)        // une bou
10      {
11  B       if (PORTA & 0x10) PORT
12          else PORTB=0;
13      }
14  }
15
```

Disassembly Listing

```
6:                  void main(void)
7:                  {     TRISA=0xFF;      // PORTA en entré
0000E6    6892      SETF 0xf92, ACCESS
8:                  TRISB = 0;        /* PB en sortie */
0000E8    6A93      CLRF 0xf93, ACCESS
9:                    while(1)        // une boucle infinie
0000F6    D7F9      BRA 0xea
10:                  {
11:                      if (PORTA & 0x10) PORTB=1;
B 0000EA  A880      BTFSS 0xf80, 0x4, ACCESS
0000EC    D003      BRA 0xf4
0000EE    0E01      MOVLW 0x1
0000F0    6E81      MOVWF 0xf81, ACCESS
12:                      else PORTB=0;
0000F2    D001      BRA 0xf6
0000F4    6A81      CLRF 0xf81, ACCESS
13:                  }
14:                 }
0000F8    0012      RETURN 0
```

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

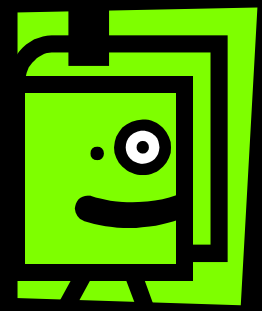
reset → Origine d'un RESET

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
ctype → Classification des caractères
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

Afficheur LCD (xlcd.h)

Fonctions	Descriptions
void **OpenXLCD**(unsigned char lcdtype);	Initialise l'afficheur LCD
unsigned char **BusyXLCD**(void);	La fonction teste la disponibilité du contrôleur LCD.
void **putsXLCD**(char *buffer); void **putrsXLCD**(const rom char *buffer);	Affiche une chaîne présente en **RAM** Affiche une chaîne présente en **ROM**
unsigned char **ReadAddrXLCD**(void);	Cette fonction lit l'adresse courante du LCD .
char **ReadDataXLCD**(void);	Cette fonction lit l'octet à l'adresse spécifiée du LCD .
void **SetCGRamAddr**(unsigned char addr);	Fixe l'adresse en ram du générateur de caractères du LCD
void **SetDDRamAddr**(unsigned char addr);	Fixe l'adresse d'affichage des caractères
void **WriteCmdXLCD**(unsigned char ***cmd***);	**Envoie une commande à l'afficheur**
void **WriteDataXLCD**(char ***data***); ou void **putcXLCD** (char ***data***); ou void **putchar**(char ***data***);	Les deux fonctions envoient un caractère sur l'afficheur.

btoa, itoa, ftoa

- byte to ASCII (btoa) : char vers ASCII
- integer to ASCII (itoa) : int vers ASCII
- float to ascii (ftoa) : float vers ASCII
- ftoa n'est pas incluse dans MCC18
- La fonction doit être déclarée dans le projet ou intégrée à une bibliothèque

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

prec indique la précision, 0 pour avoir le maximum

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

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

stdio.h

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

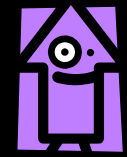
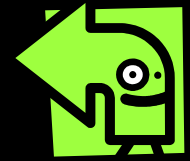
- Exemples :

  putc(unsigned char,_H_USART) vers l'USART

  puts (« bonjour »);

  printf, sprintf, vprintf …


Redirections

- Les fonctions de stdio.h envoient les caractères vers le « flux » stdout.
- Par défaut stdout=_H_USART;
- Pour choisir une autre sortie il faut :
  - Redéfinir stdout, **stdout=_H_USER;**
  - Definir la fonction int _user_putc(char c);
    - **Ex : int _user_putc(char c)**

```
{
        putcXLCD(c)
}
```

L’afficheur de PICDEM2+

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

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

Installation : installXLCDPD2.bat
Utilisation : #include <lcd_pd2.h>

Installation de stdio pour PICDEM2+

- Le fichier **installXLCDPD2.bat**

  installe les modifications permettant à la librairie **xlcd.h** de gérer l'afficheur LCD du KIT PICDEM2+ (**xlcd.h**)

- Le fichier **installLCDLIB.bat**

  installe **_user_putc** pour **printf** ainsi que quelques utilitaires de gestion de l'afficheur LCD **initLCDPD2, gotoXY, ftoa, etc...** (**lcd_pd2.h**)

Le fichier installTOUT.bat lance les deux précédents

Utiliser la librairie stdio.h et la fonction printf

printf envoie les caractères vers « stdout »
Il faut définir stdout vers _H_USART ou _H_USER
La sortie _H_USART est automatiquement définie sur l'USART
La sortie _H_USER doit être définie par l'utilisateur

```
void init_printf_LCDPD2(void)
{
stdout=_H_USER;
OpenXLCD(FOUR_BIT &
LINES_5X7 );
gotoxy(0,0);
}
```

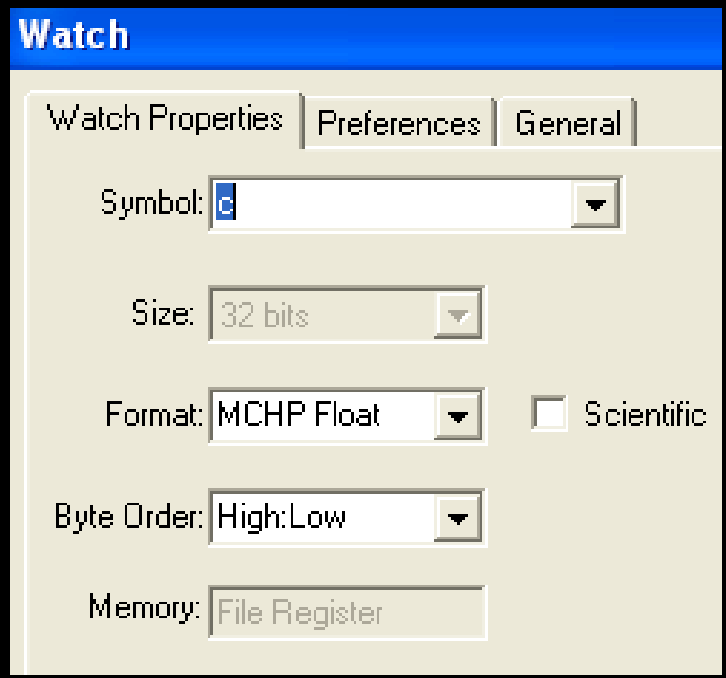
```
int _user_putc(char c)
{
    putcXLCD(c);
    return (c);
}
```

Exemples sur printf

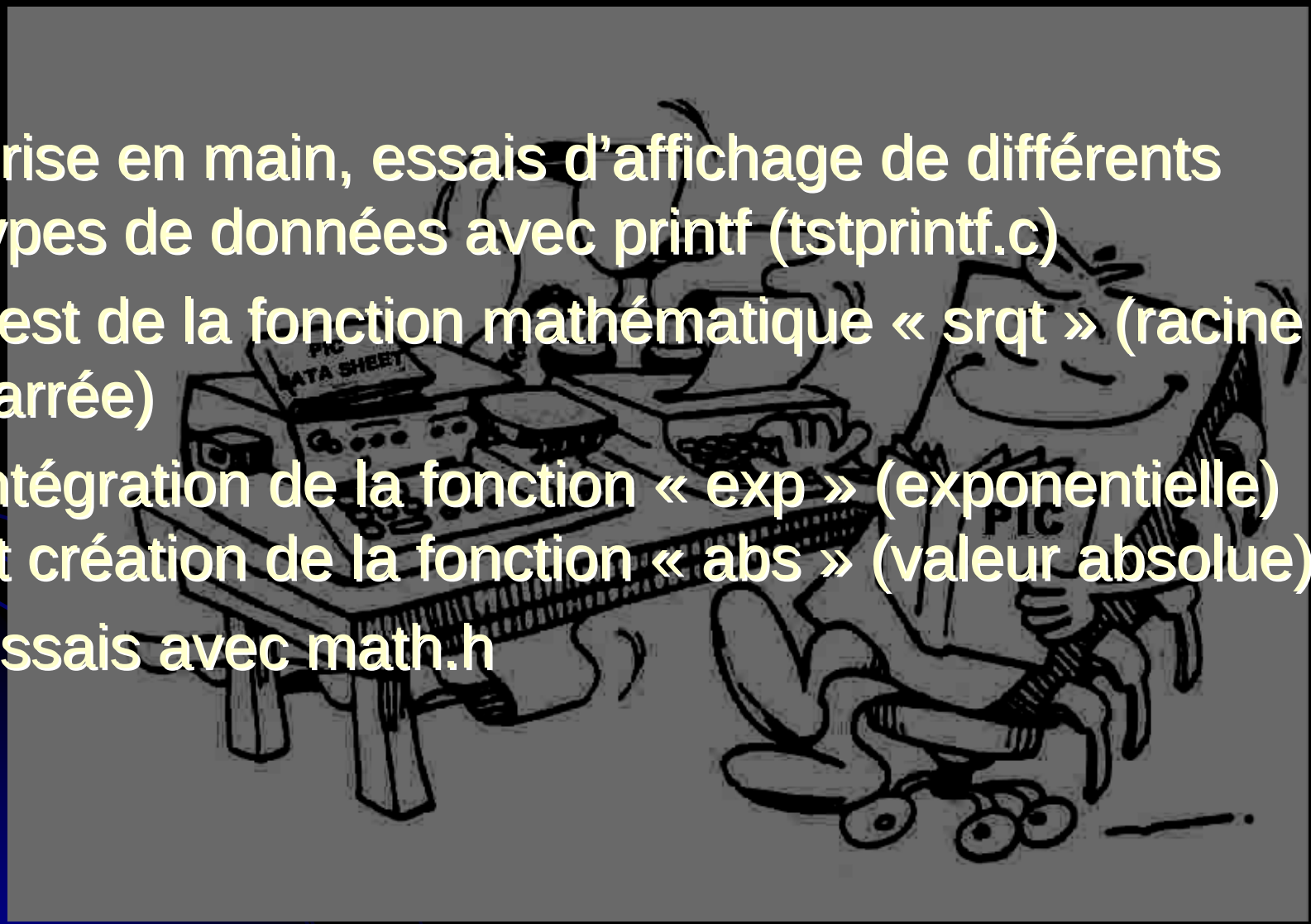
`printf("Dec : %d %u",a,a);`	`Dec : -27 65509`
`printf("Hex: %#06X %x ",b,b);`	`Hex: 0X00B5 b5`
`printf("Bin: %16b",b);`	`Bin: 0000000010110101`
`printf("Bin: %#010B",b);`	`Bin: 0B10110101`
`printf("%c %c %d",'b',c,(int)c);`	`b A 65`
`printf("J habite %S",chrom);`	`J habite en ROM`
`printf("J habite %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`

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);
…
}
```

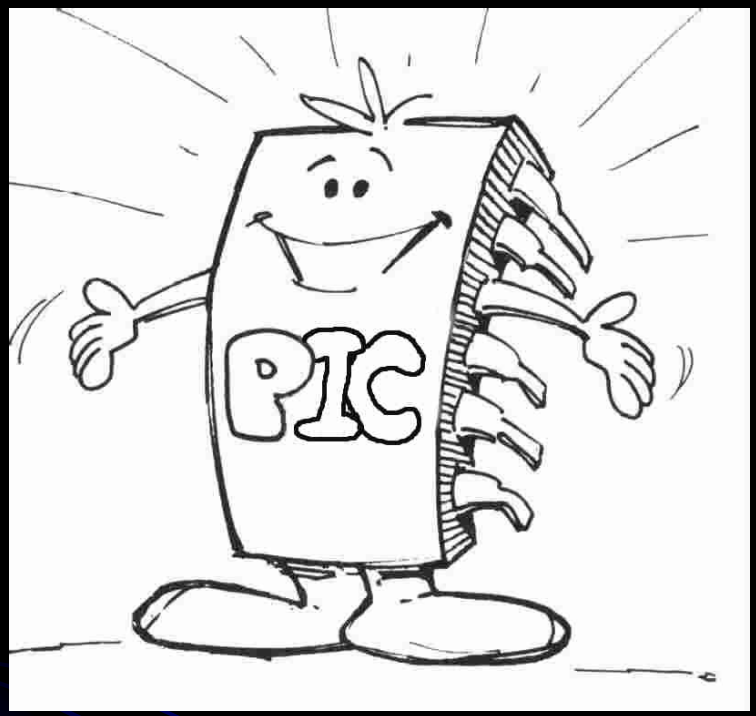


Exercices sur librairies



- 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

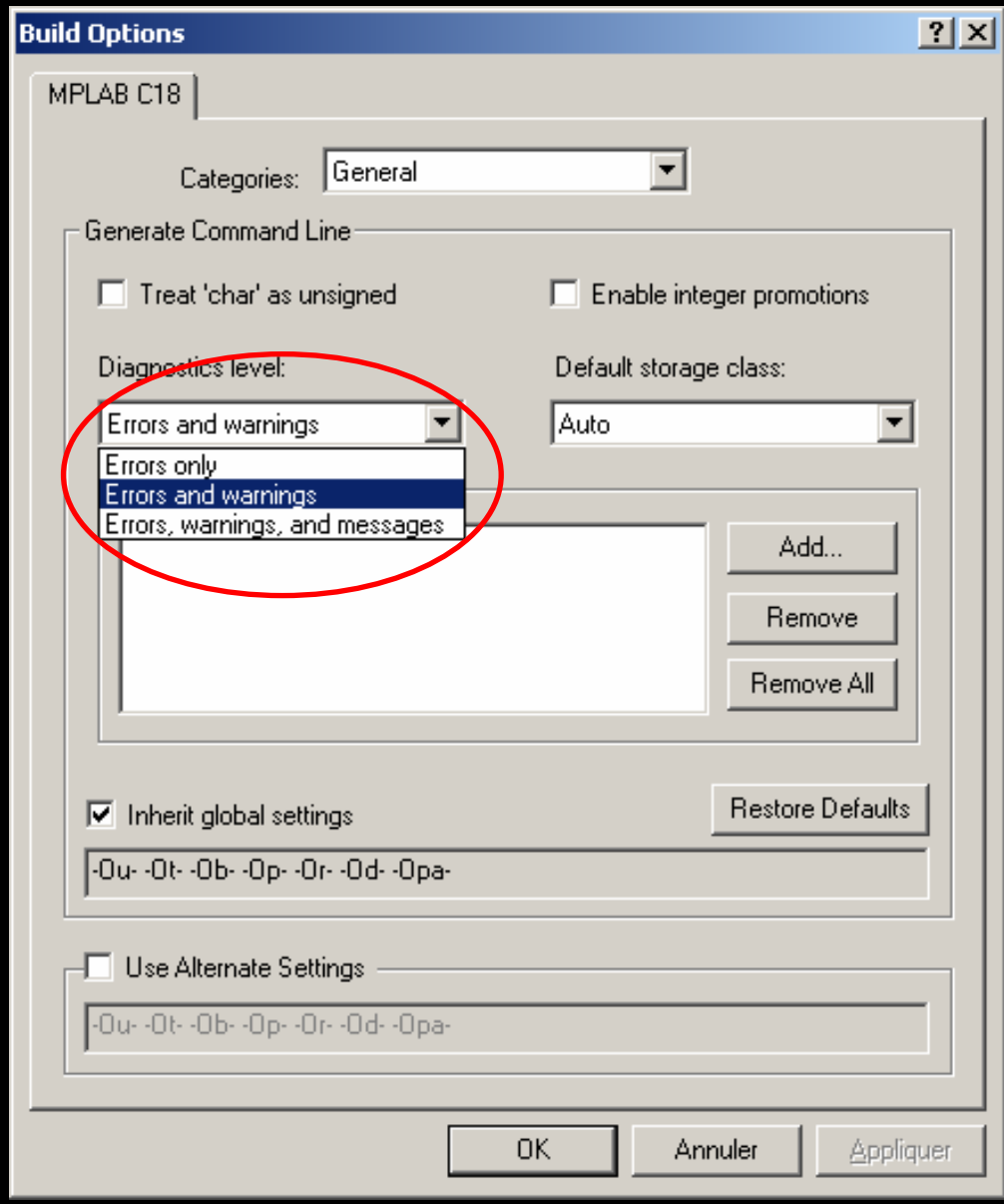
Deuxième journée

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

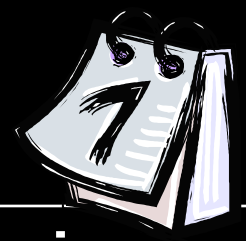
Spécificités du compilateur

Diagnostic des erreurs

Niveau	Diagnostic affiché
1	Codes Erreurs
2	Idem 1 + avertissements
3	Idem 2 + messages



Types entiers



Type	Size	Minimum	Maximum
char	8 bits	-128	127
signed char	8 bits	-128	127
unsigned char	8 bits	0	255
int	16 bits	-32768	32767
unsigned int	16 bits	0	65535
short	16 bits	-32768	32767
unsigned short	16 bits	0	65535
short long	24 bits	-8,388,608	8,388,607
unsigned short long	24 bits	0	16,777,215
long	32 bits	-2,147,483,648	2,147,483,647
unsigned long	32 bits	0	4,294,967,295

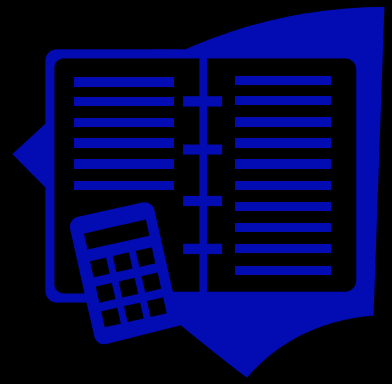
Types réels

Type	BITS	Minimum	Maximum
float	32	$2^{-126} = 1.17549435e^{-38}$	$2^{128} = 6.80564693e^{+38}$
double	32	$2^{-126} = 1.17549435e^{-38}$	$2^{128} = 6.80564693e^{+38}$

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 [overlay] [nom[=adresse]]***

Contient des instructions exécutables

romdata ***: #pragma romdata [overlay] [nom[=adresse]]***

Contient des constantes et des variables

udata : #pragma udata [overlay/access] [nom[=adresse]]

Contient des variables statiques non initialisées

idata ***: #pragma idata [overlay/access] [nom[=adresse]]***

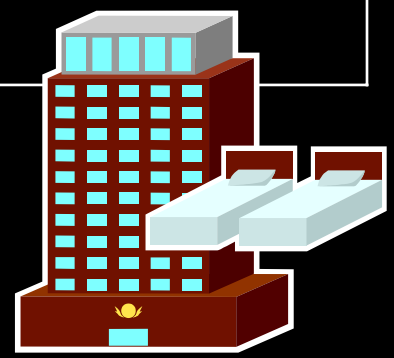
Contient des variables statiques non initialisées

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



Taille des pointeurs

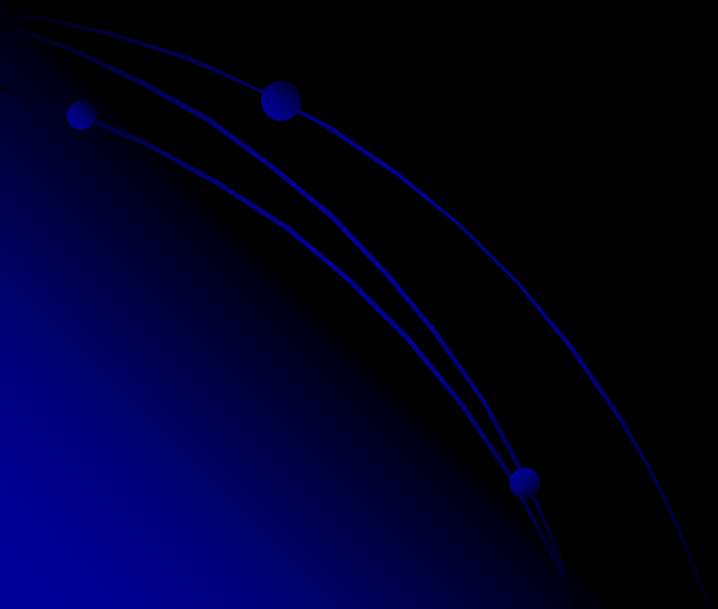
Type	Exemple	Taille
Pointeur de données en RAM	char * dmp;	16 bits
Pointeur de donnée en ROM (<64KO)	rom near * npmp;	16 bits
Pointeur de donnée en ROM (>64KO)	rom far * fpmp;	24 bits

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

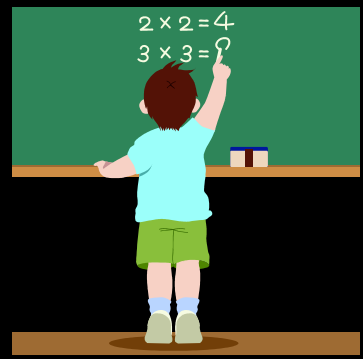


Exemple 1 : Qualificatifs de mémorisation

```
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;
}
```

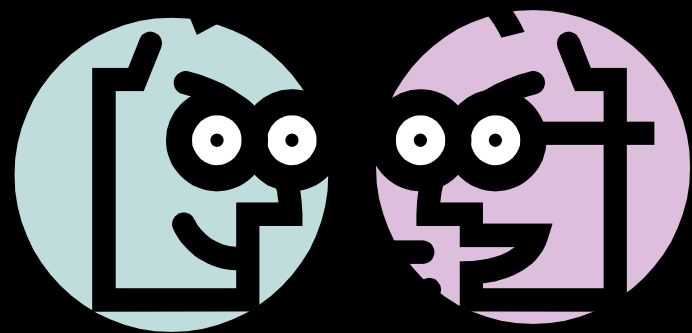
Gestion d'un tableau

```
#include <p18f452.h>
#define tbltaille 16          // taille de la table
const rom 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);
                        }
        }
}
```



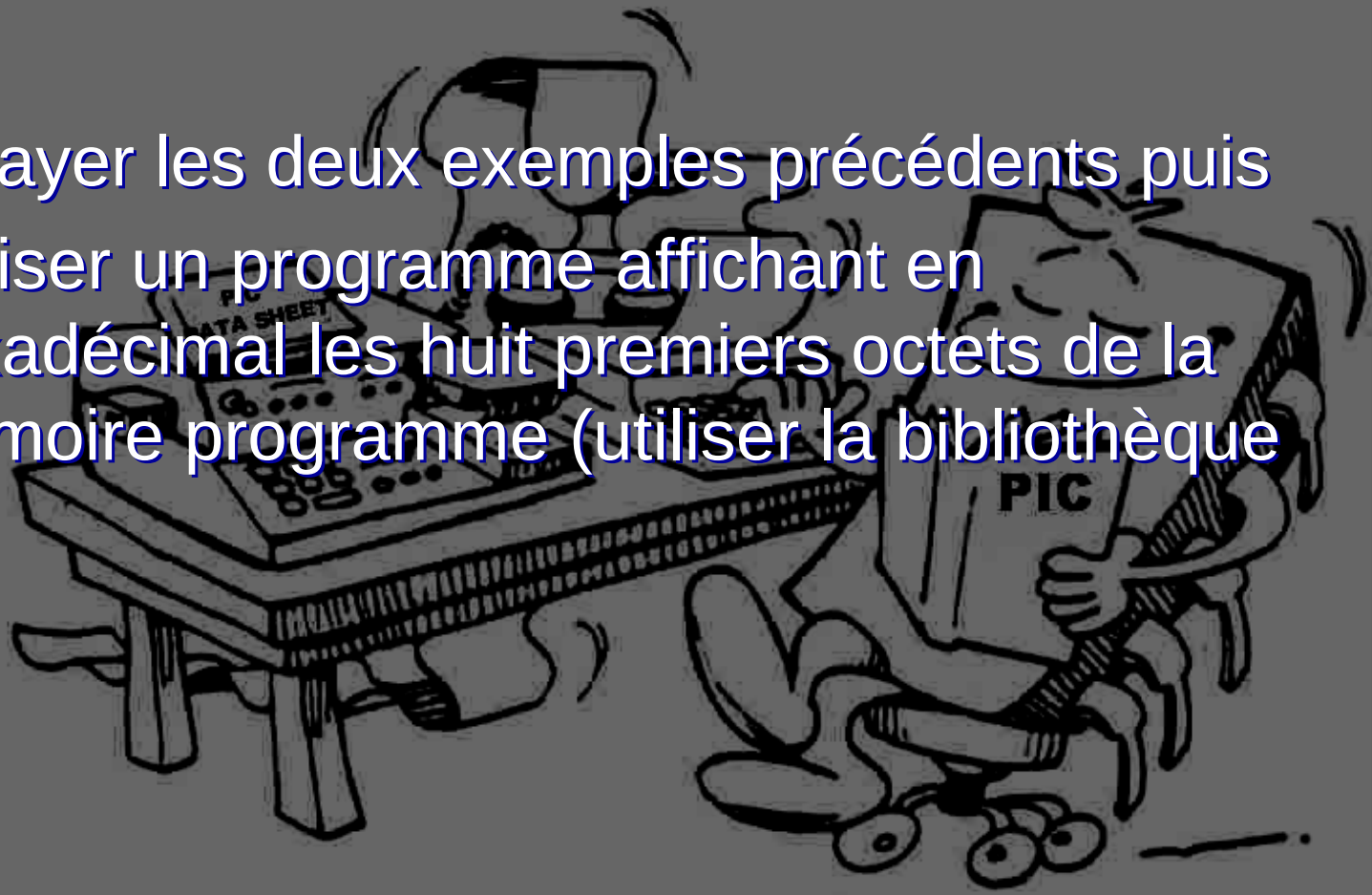
Copie ROM/RAM

```
#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);
}
```

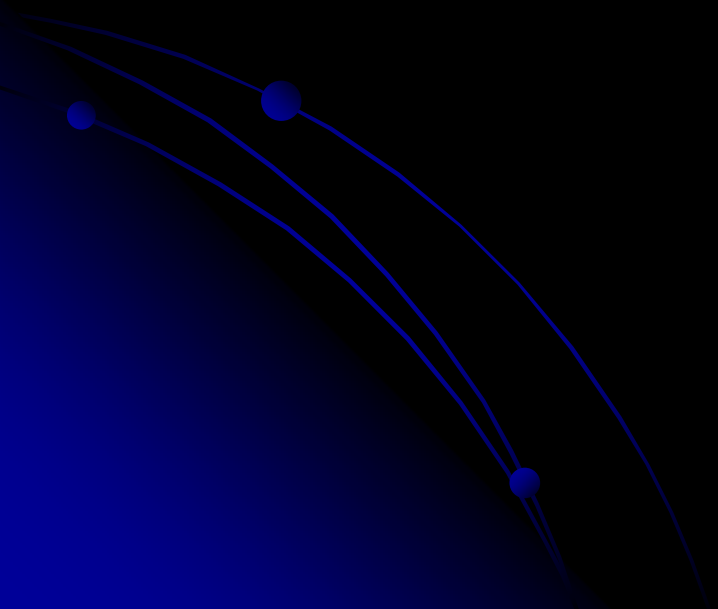


Gestion mémoire : Exercice

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

#pragma *interrupt*
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

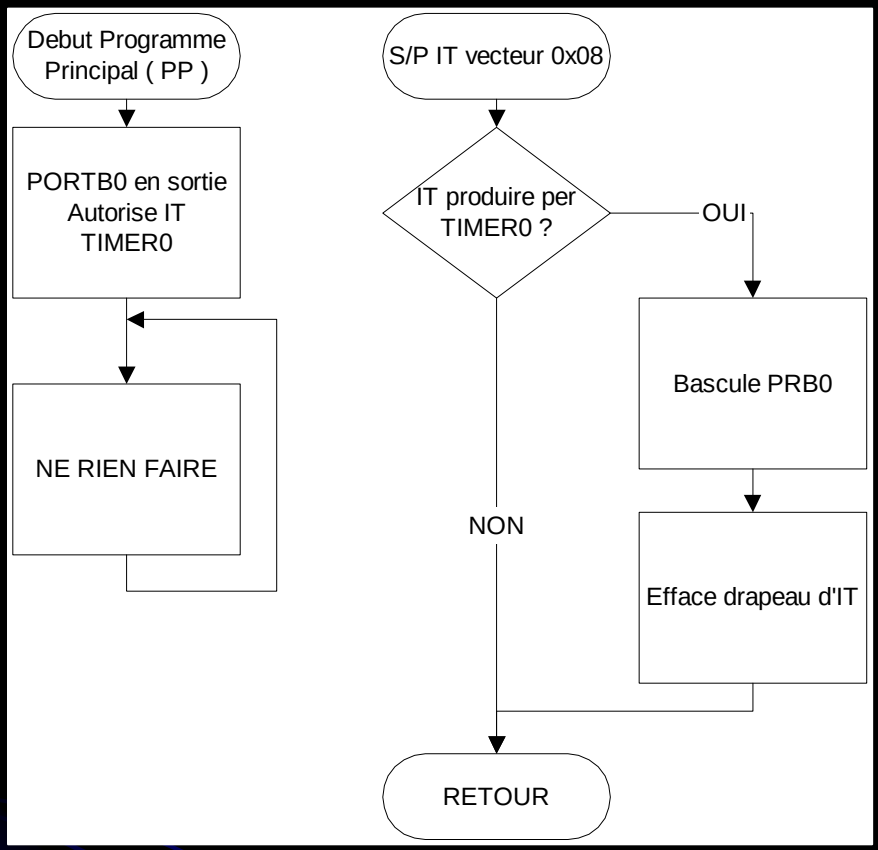
#pragma code redonne toute liberté au compilateur

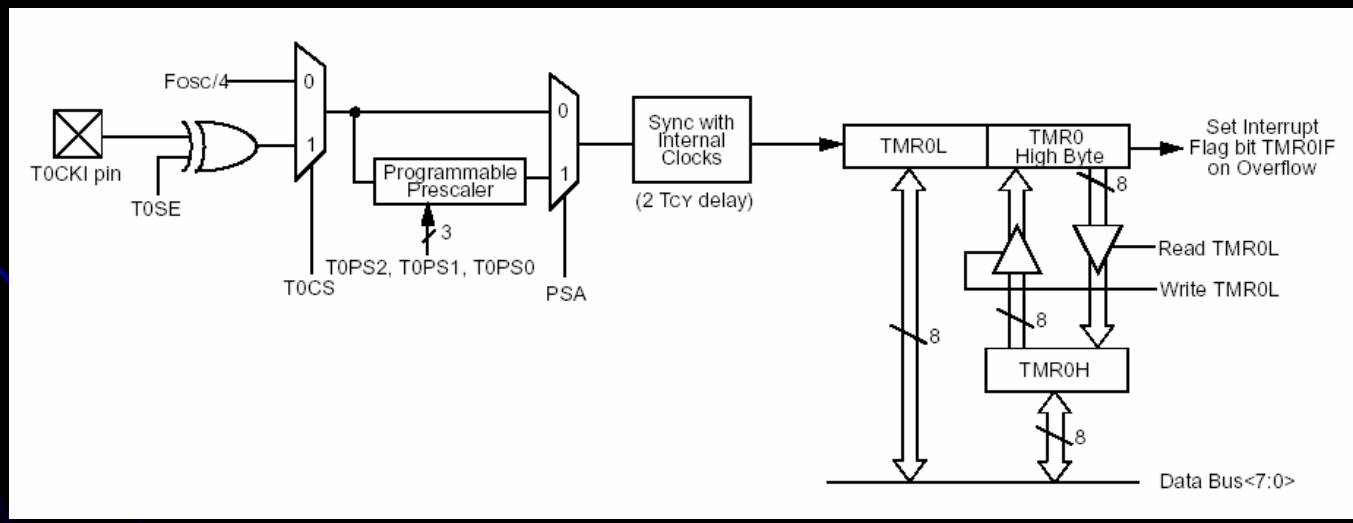
Les priorités d'IT sur PIC18

- 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 TIMER 0

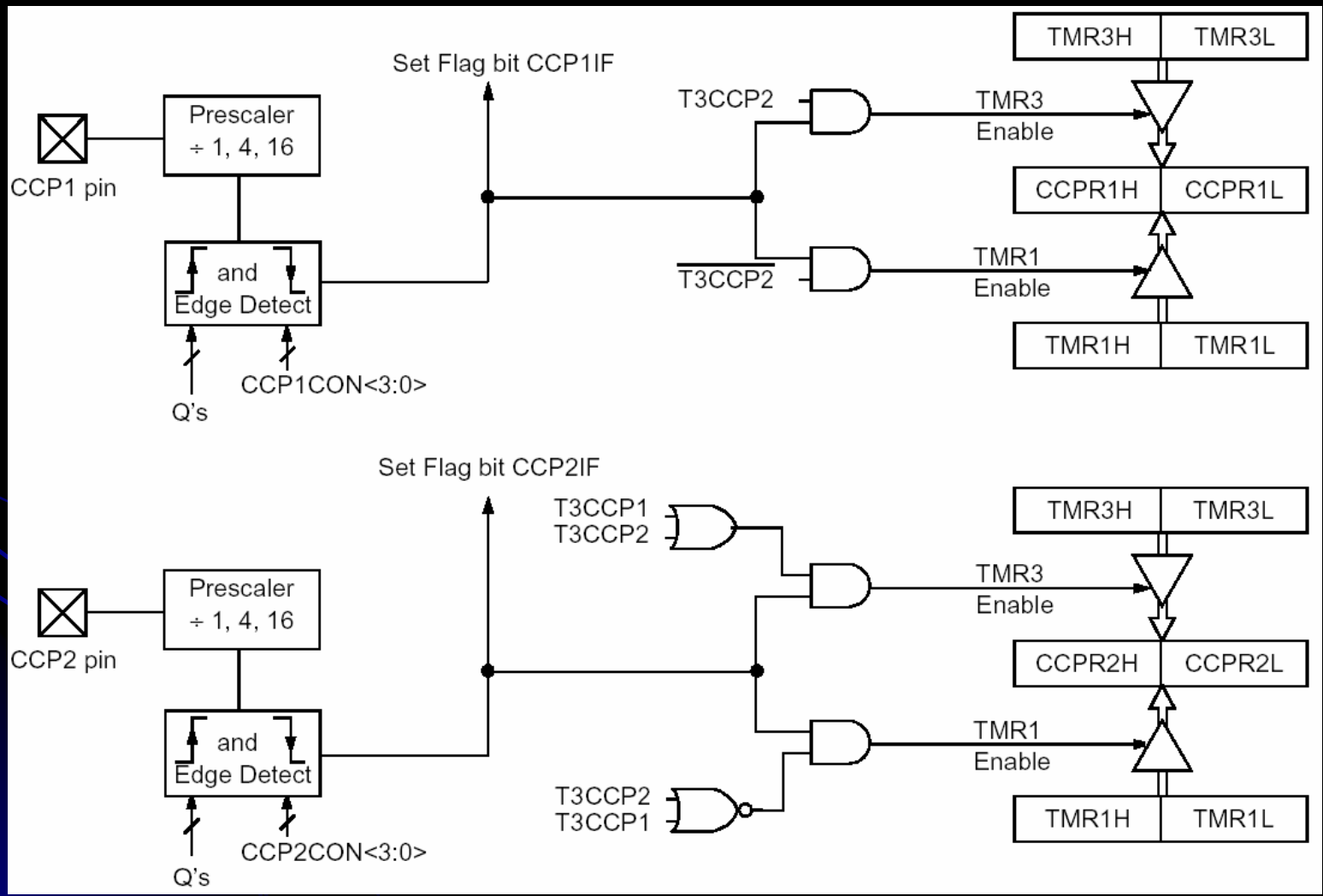


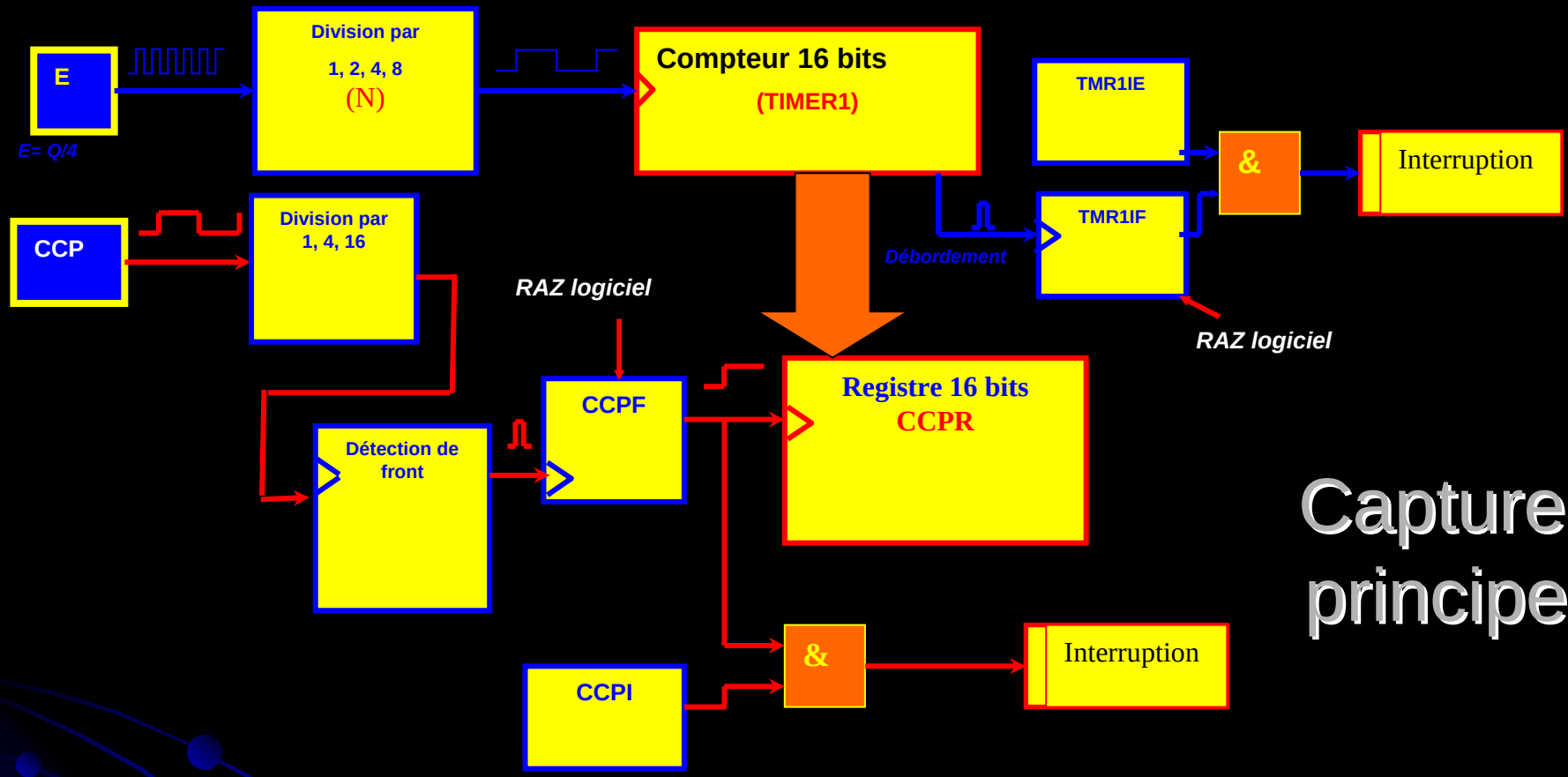


Interruptions

```
#include <p18f452.h>
void traiteIT(void);
#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
}
```

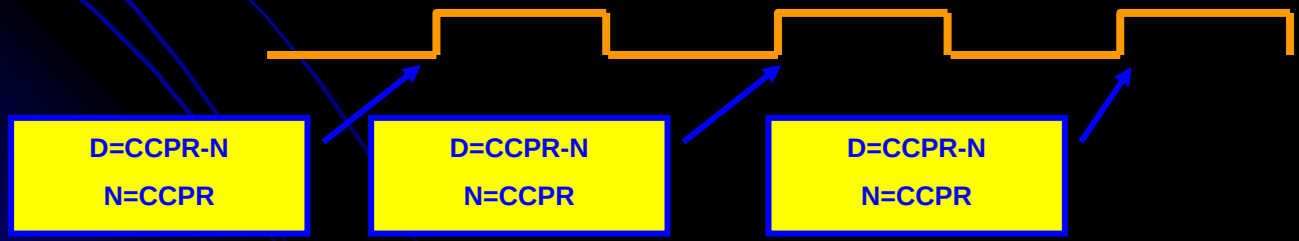
CAPTURE


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

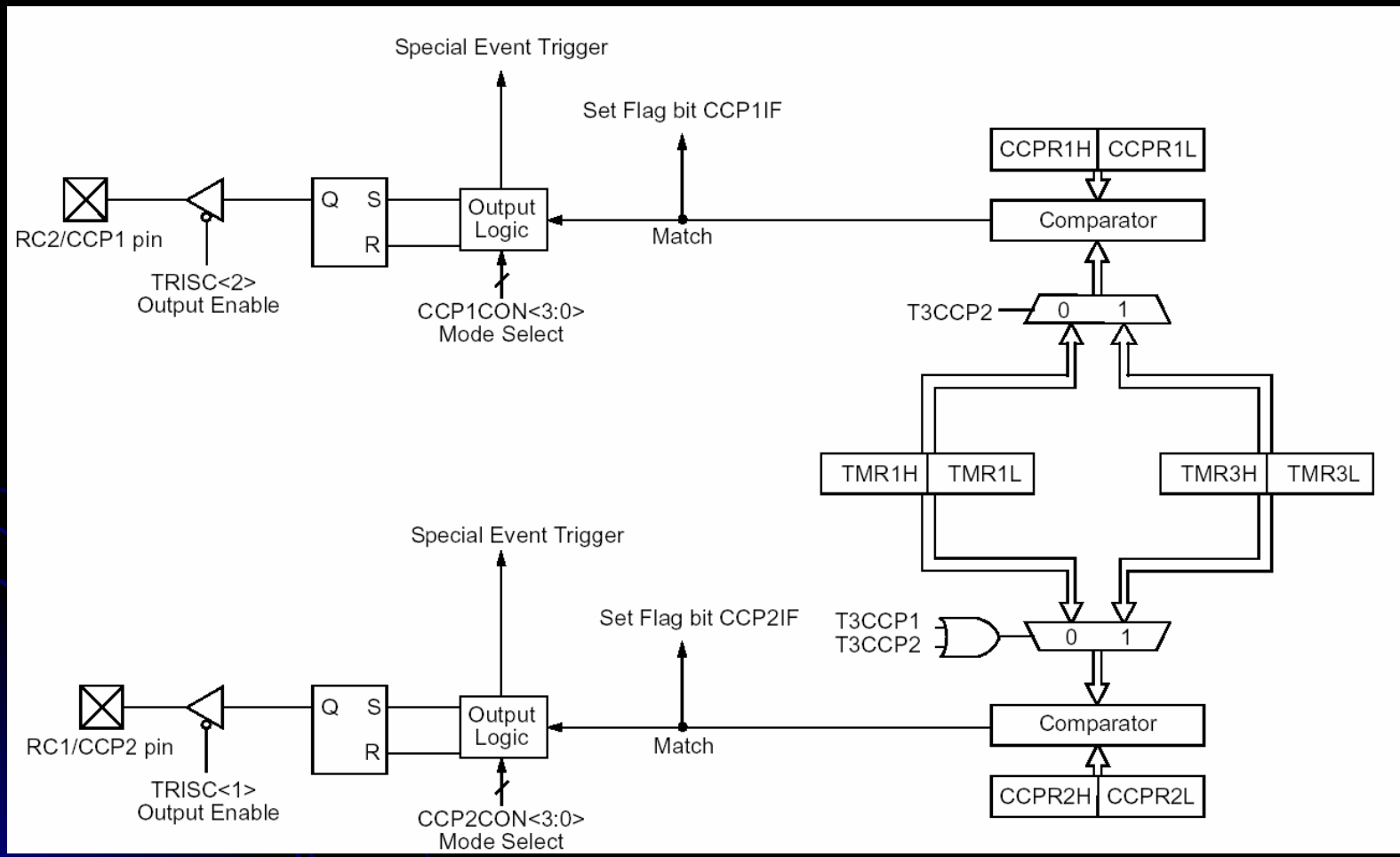


Capture - principes

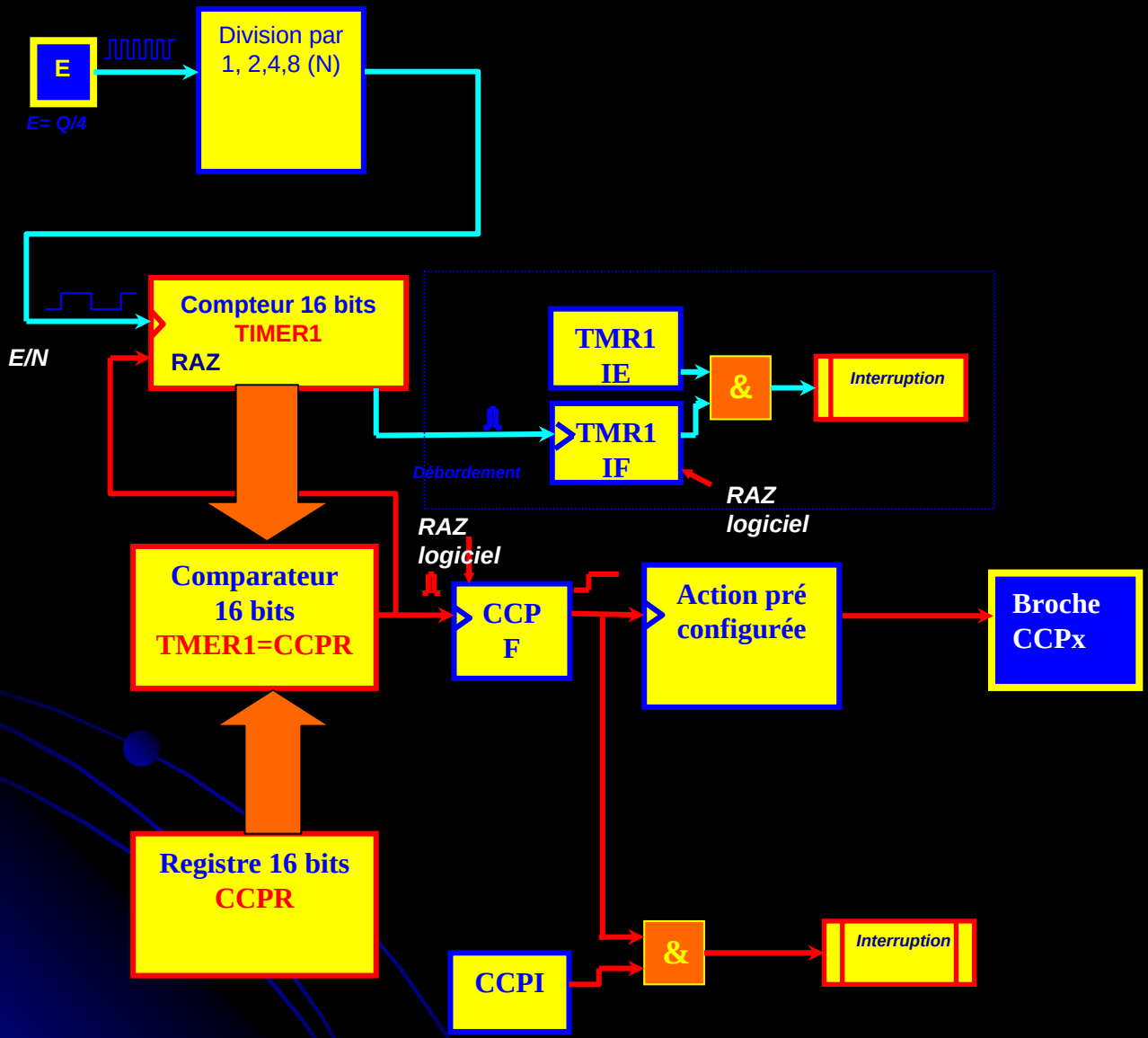
T=D*E/N



COMPARE

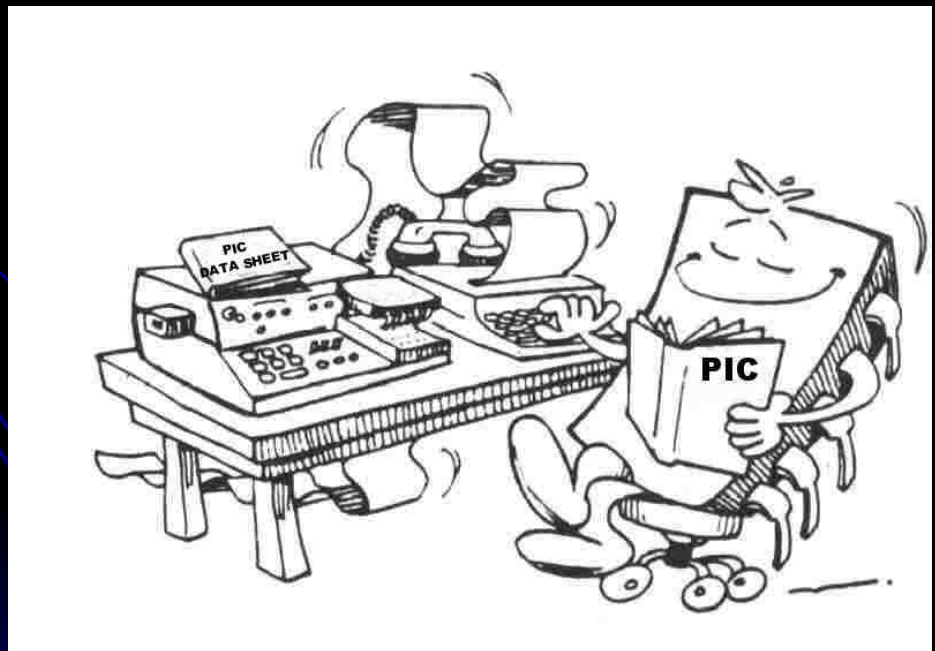

Special Event Trigger
Set Flag bit CCP1IF
CCPR1H
CCPR1L
Comparator
RC2/CCP1 pin
Q
S
R
Output Logic
Match
TRISC<2>
Output Enable
CCP1CON<3:0>
Mode Select
T3CCP2
0
1
TMR1H
TMR1L
TMR3H
TMR3L
Special Event Trigger
Set Flag bit CCP2IF
T3CCP1
T3CCP2
0
1
RC1/CCP2 pin
Q
S
R
Output Logic
Match
Comparator
TRISC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

COMPARE - Principes



Exercices avec interruptions

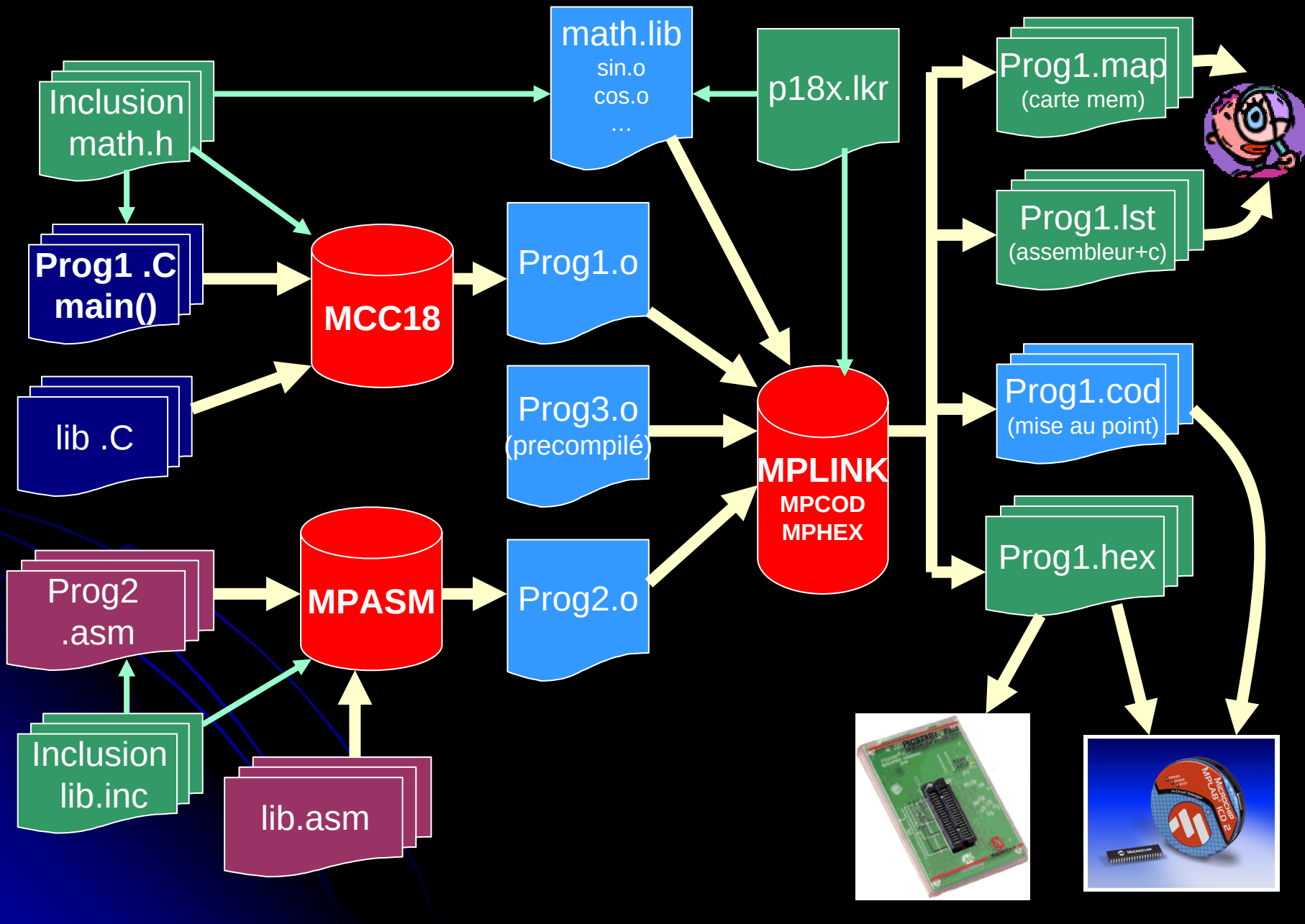
- **Production de temporisation**
- **Modification de rapport**
- **Mesure de durée (fréquencemètre)**
- **Réalisation d’une horloge temps réel**



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

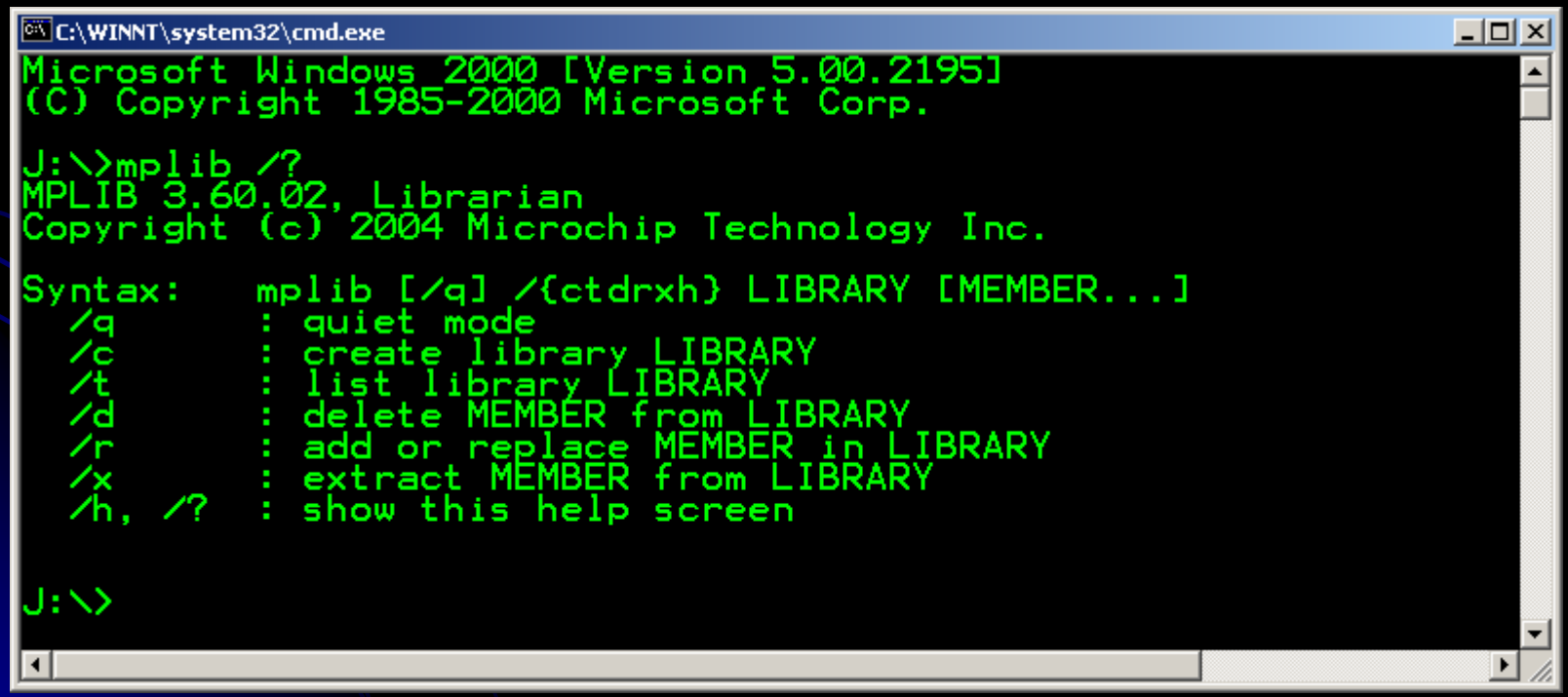
MPLIB - LANCEMENT

MPLIB est un programme « console »

Sous WINDOWS :

Démarrer – exécuter – cmd

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



Utilisation d'une librairie

Les bibliothèques personnelles doivent être déclarées au compilateur

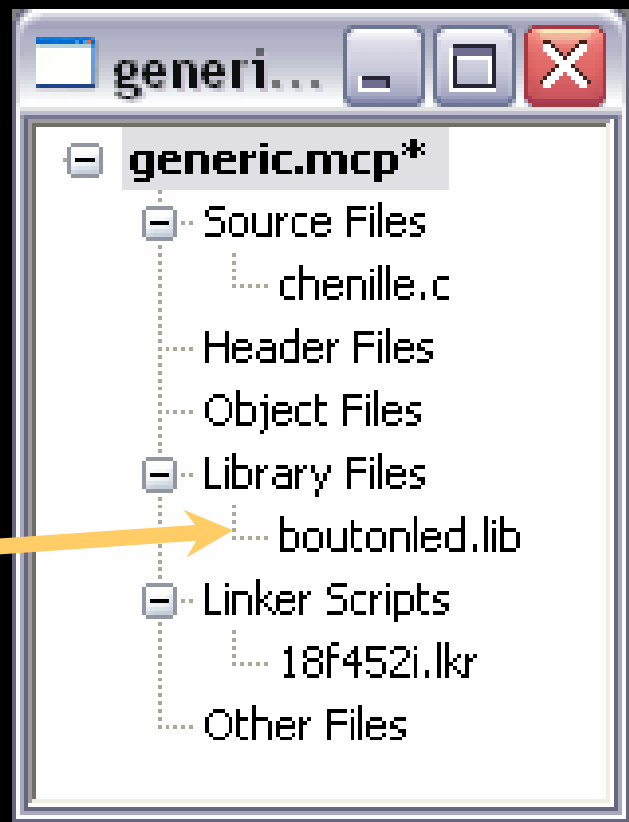
Ajouter les lignes suivantes dans p18fxx.lkr

LIBPATH .;..\persolib

FILES boutonled.lib

Ou

Inscérer la librairie dans le gestionnaire de projet MPLAB



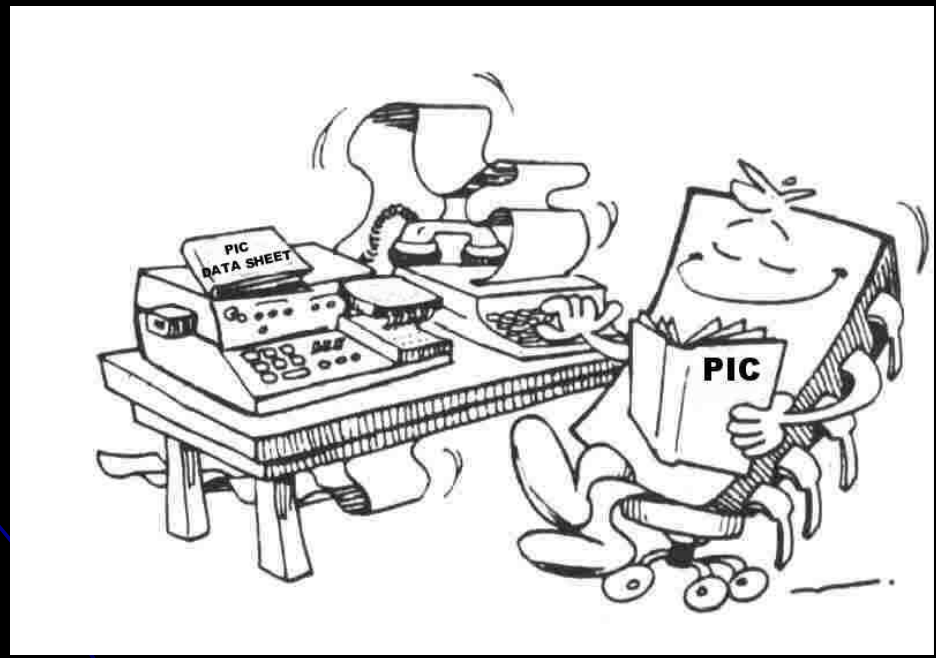
Créer une bibliothèque

- 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

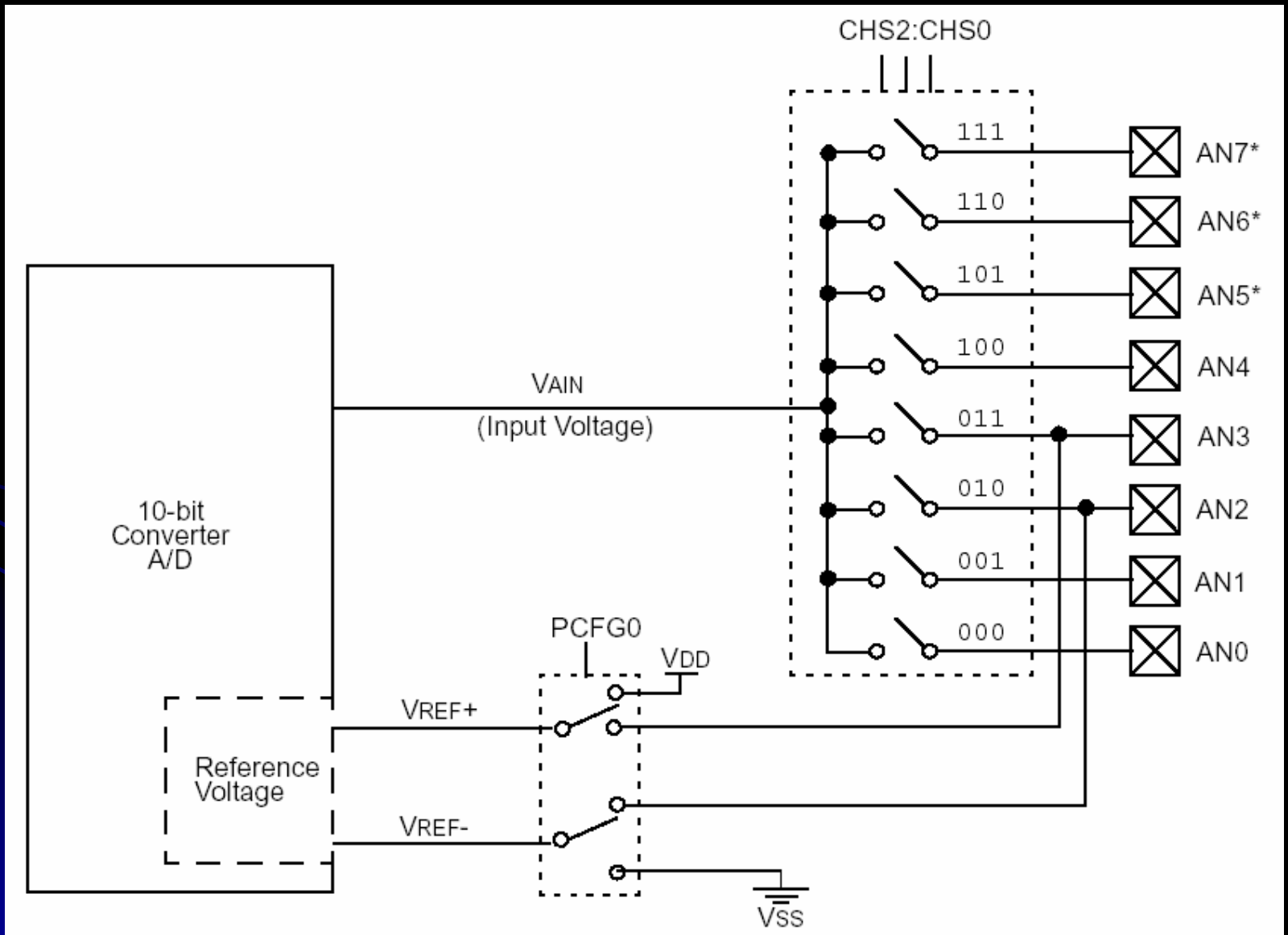
Exercices avec MPLIB

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

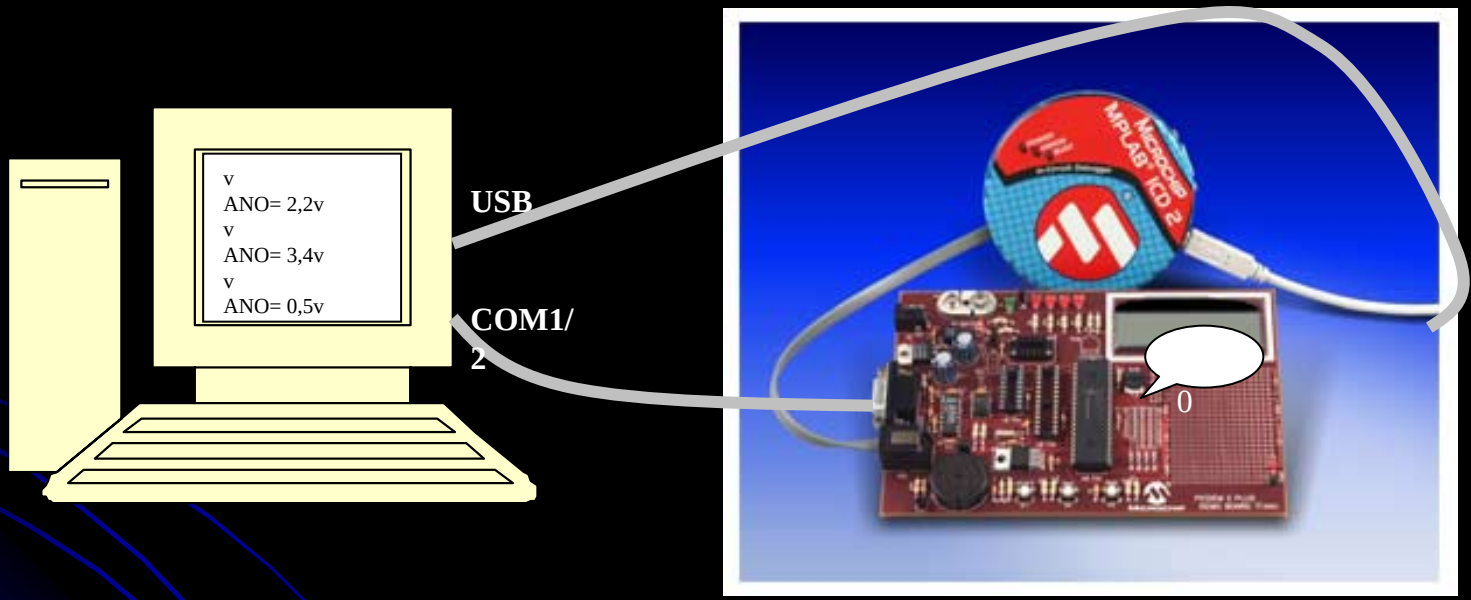


Mise en œuvre des périphériques intégrés sur PIC18

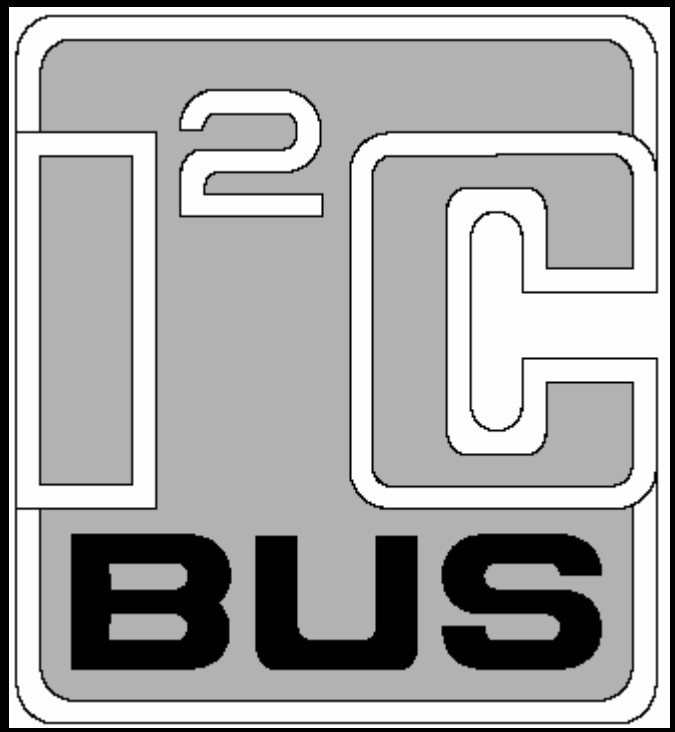
ADC 10 bits


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

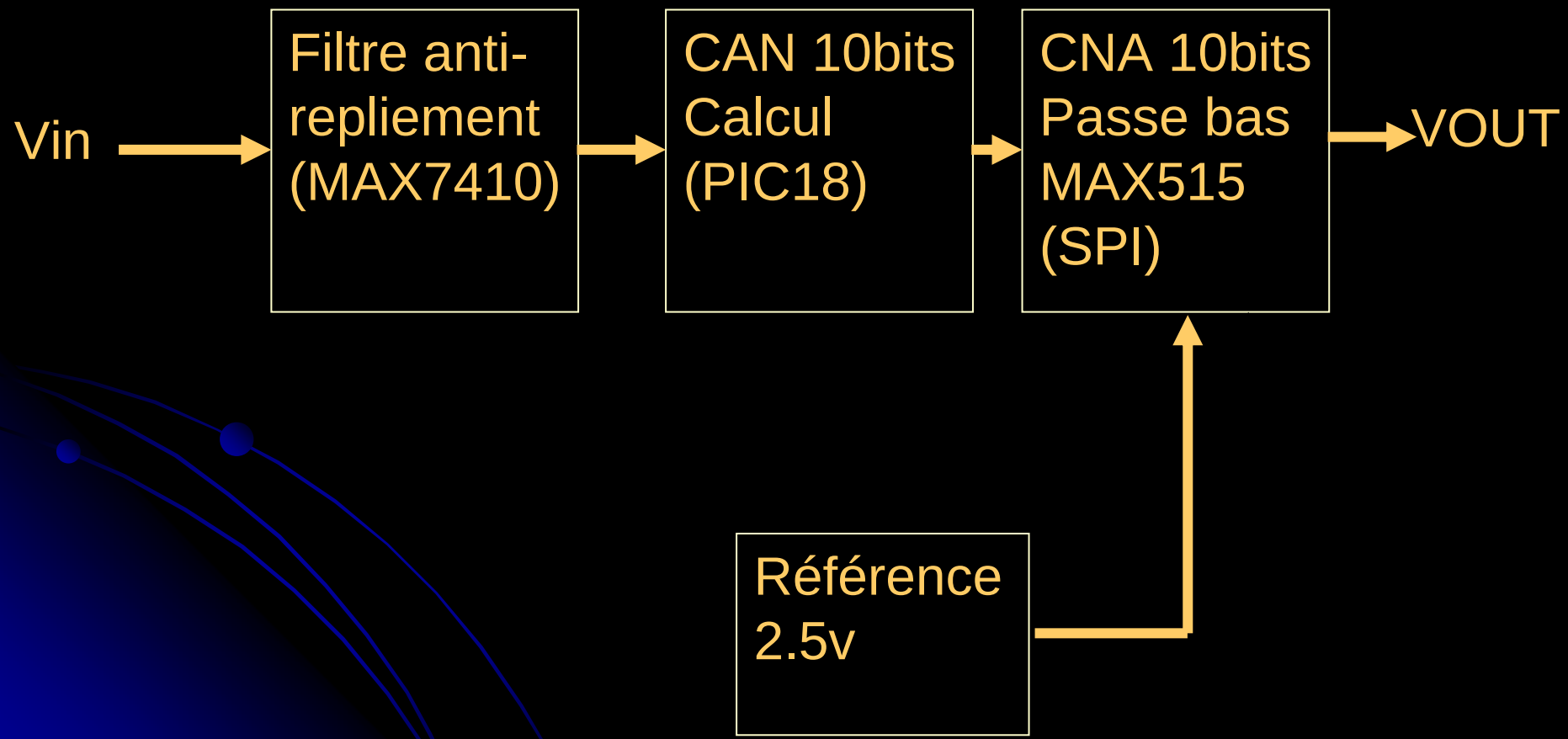
USART



I2C

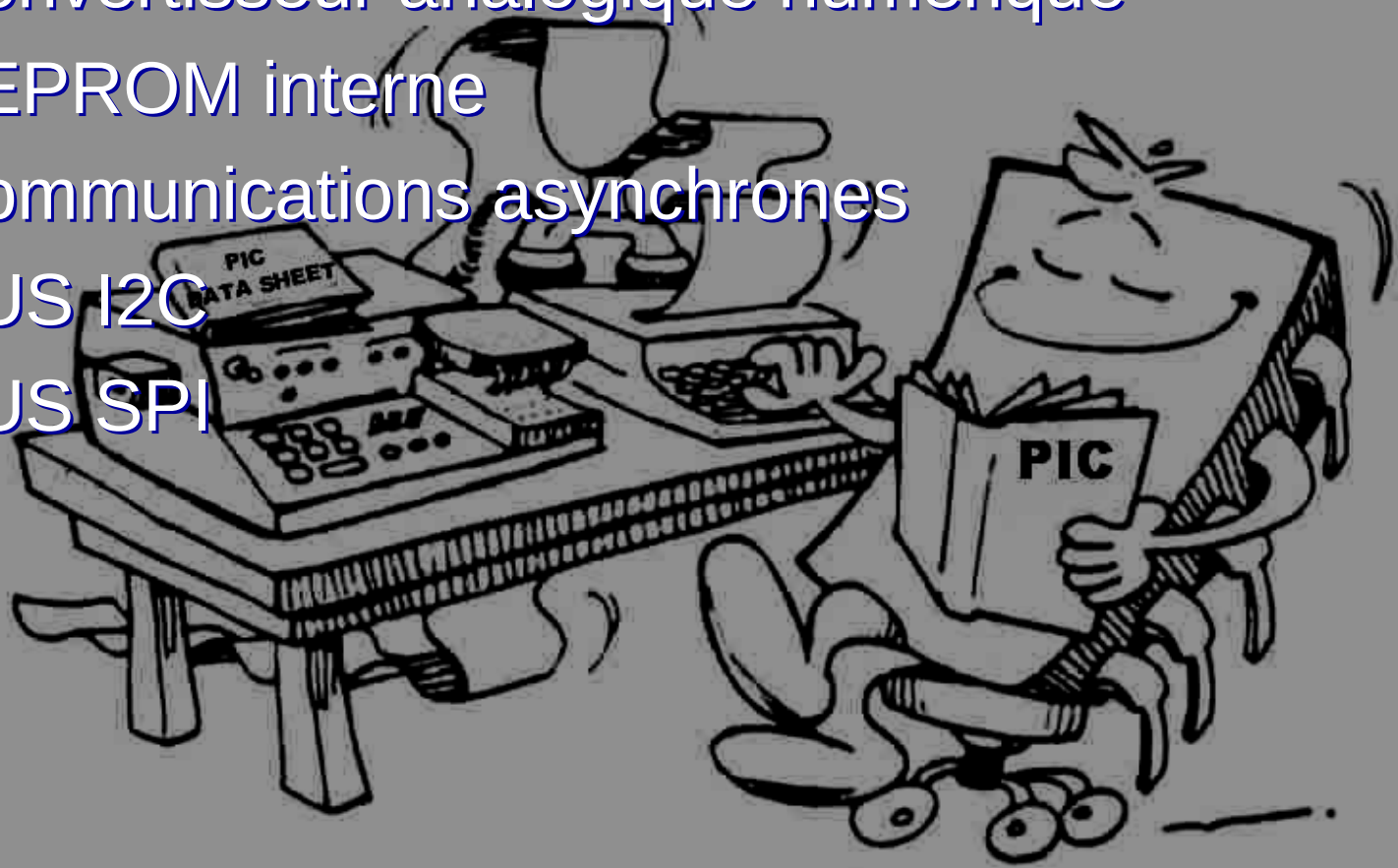


SPI ex: filtre numérique



Exercices sur …

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



Merci pour votre attention