DUT Génie Électrique – Département GEII2

Génie Electrique – Semestre 1

II1 - Travaux Dirigés

Informatique Industrielle – Partie 1

Auteur(s) : Mise à jour H. Discours
Date – Version : Novembre 2022

LISTE DES OBJECTIFS
II1-1 : Évaluer une expression comportant des opérateurs de bits ou de décalage
II1-2 : Savoir lire un port d'entrée d'un microcontrôleur
II1-3 : Savoir tester un ou plusieurs bits sur un port d'entrée, indépendamment des autres
II1-4 : Savoir écrire un ou plusieurs bits sur un port de sortie, sans modifier les autres
II1-5 : Passer, en C, de la valeur issue du CAN à la valeur de la tension électrique associée
II1-6 : Savoir traduire en C un cahier des charges simple avec lecture d'entrées et écriture de sorties
II1-7 : Savoir câbler un bouton-poussoir sur une entrée d'un microcontrôleur
II1-8 : Savoir câbler une LED sur une sortie d'un microcontrôleur

1° SEANCE DE TD

1. Révisions

a) Révisions de numération : Compléter le tableau suivant

Décimal	Binaire (8 bits)	Hexadécimal (8 bits = 2 chiffres)
1	0000 0001	0x01
2		
4		
		0x55
15		
	1111 0000	
		0xAA
128		
		0xCE
	1001 1001	

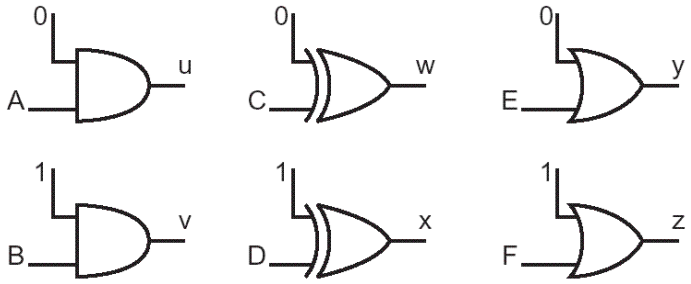
b) Révisions sur les opérateurs logiques :

A, B, C, D, E, F sont des bits (0 ou 1). Compléter les équations ci-dessous :

$A.0 = \ldots$ $B.1 = \ldots$ $C + 0 = \ldots$ $D + 1 = \ldots$ $E \oplus 0 = \ldots$ $F \oplus 1 = \ldots$

c) Révisions sur les opérateurs logiques et leurs symboles :

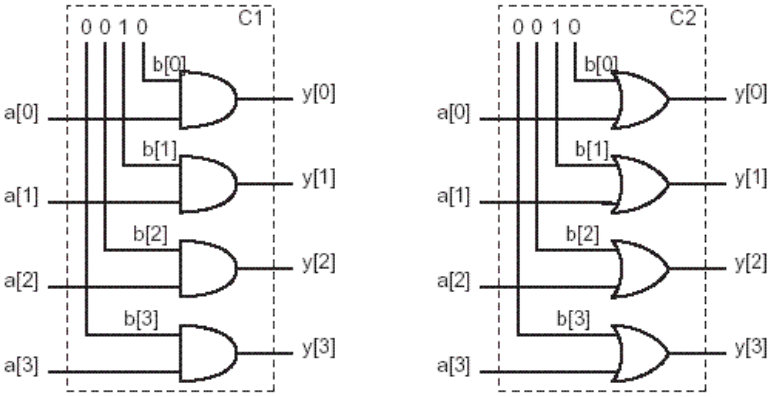
Pour chacun des schémas ci-dessous, préciser l'état de sa sortie



2. Exercices

Exercice 1 : Bits et masques

Sur les schémas logiques suivants, le port a[ ] représente les entrées (4 bits) de chaque bloc (C…), le port y[ ] les sorties (4 bits). Le nombre b[ ] (sur 4 bits) est une valeur interne à chaque bloc.



Q1.1 Pour le bloc C1 : Quelle est la valeur de y[0] ?
La valeur de y[1] est-elle liée à celle de a[1] ?
Quelle est la valeur de y[2] ?
Quelle est la valeur de y[3] ?

Si par exemple a prend la valeur 1110, compléter le tableau suivant :

	[3]	[2]	[1]	[0]
a[]	1	1	1	0
b[]	0	0	1	0
y[] = a[] ET b[]				

La sortie y est-elle modifiée si a[0] change ? Si oui quel bit ?
La sortie y est-elle modifiée si a[1] change ? Si oui quel bit ?
La sortie y est-elle modifiée si a[2] change ? Si oui quel bit ?
La sortie y est-elle modifiée si a[3] change ? Si oui quel bit ?

Quelle valeur doit prendre b pour que la sortie y ne « lise » que le bit a[3] ?

Q1.2 Pour le bloc C2 : La valeur de y[0] est-elle liée à celle de a[0] ?
Quelle est la valeur de y[1] ?
La valeur de y[2] est-elle liée à celle de a[2] ?
La valeur de y[3] est-elle liée à celle de a[3] ?

Si par exemple a prend la valeur 1110, compléter le tableau suivant :

	[3]	[2]	[1]	[0]
a[]	1	1	1	0
b[]	0	0	1	0
y[] = a[] OU b[]				

La sortie y est-elle modifiée si a[0] change ? Si oui quel bit ?
La sortie y est-elle modifiée si a[1] change ? Si oui quel bit ?
La sortie y est-elle modifiée si a[2] change ? Si oui quel bit ?
La sortie y est-elle modifiée si a[3] change ? Si oui quel bit ?

Quelle valeur doit prendre b pour ne forcer à 1 que le bit a[3] ?

Q1.3 Pour les deux schémas étudiés précédemment, b[] joue le rôle de « MASQUE ». Conclure sur le comportement d'un masque en ET et d'un masque en OU.

Q1.4 Dessiner des blocs de logique combinatoire :

Bloc C3 permettant de mettre à un le bit n°4 sur un bus de 8 bits, sans modifier les autres bits.

Bloc C4 permettant de mettre à zéro le bit n°6 sur un bus de 8 bits, sans modifier les autres bits.

Exercice 2 : Opérations logiques en langage C

Ci-dessous un « morceau » de programme écrit en langage C :

```
char s1, s2, s3, s4, s5, s6, s7, s8, s9 ;
s1 = 0x22 | 0x01;
s2 = 0x23 | 0x01;
s3 = 0x00 | 0x22;
s4 = 0x31 | 0x22;
s5 = 0xFF & 0xFE;
s6 = 0x03 & 0xFE;
s7 = 0xFF & ~0x01;
s8 = 0xAA & 0x00;
s9 = 0xAA & 0xF0;
```

1) Rappeler la taille (en bits et en octet) d'une variable de type ***char***

2) Calculer les valeurs de tous les s…. Donner le détail du calcul en binaire.

Exercice 3 : Masques logiques en langage C

Ci-dessous un « morceau » de programme écrit en langage C :

```
char s21, s22, s23, s24, s25, s26, s27, s28, s29 ;

// ici affectations de valeurs à s21, s22, s23, s24, s25
(quelconques)

s21 = s21 | 0x01;
s22 = s22 | 0x05;
s23 = s23 & 0xfe;
s24 = s24 & ~0x01;
s25 = s24 & ~0x0F;
```

1) Que peut-on dire de s21, s22, s23, s24, après l'opération de masquage ?

2) Écrire l'instruction en C qui permet de :

- mettre à 1 le bit N°3 de s26, sans modifier les autres,
- mettre à 0 le bit N°5 de s27, sans modifier les autres,
- mettre à 1 les bits N°4 et N°7 de s28, sans modifier les autres,
- mettre à 0 les bits N°4 et N°7 de s29, sans modifier les autres,

Exercice 4 : Tests avec masques en langage C

Pour tout l'exercice, la variable ***z*** est de type ***char***.

1) Expliquer et justifier le sens des messages des lignes de code ci-dessous :

```
if ((z & 0x01) == 0) { printf("bit 0 de z est à 0"); }
if ((z & 0x01) == 0x01) { printf("bit 0 de z est à 1"); }
if ((z & 0x01) != 0) { printf("bit 0 de z est à 1"); }
if ((z & 0x08) == 0) { printf("bit 3 de z est à 0"); }
if ((z & 0x20) != 0) { printf("bit 5 de z est à 1"); }
if ((z & 0xF0) == 0xF0) { printf("z est de la forme 1111 xxxx"); }
if ((z & 0x13) == 0x01) { printf("z est de la forme xxx0 xx01"); }
```

2) Compléter sur le même modèle les lignes de code ci-dessous :

```
if ((z & 0x40) == 0) {
   printf("....................................."); }
if ((z & 0x02) != 0) {
   printf("....................................."); }
if ((z & 0xA0) == 0x80) {
   printf("....................................."); }
```

3) Écrire en langage C, l'expression qui permet de vérifier que :

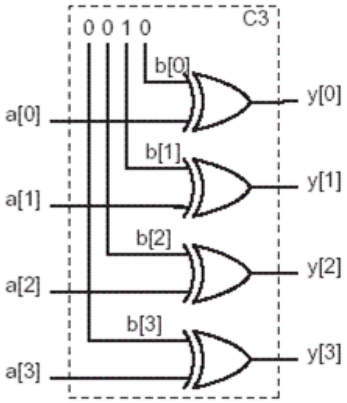
- le bit n°4 de la variable z est à 1, en utilisant un test ==
- le bit n°4 de la variable z est à 1, en utilisant un test !=
- la variable z est de la forme 0xxx xxxx (deux solutions possibles)
- qui permet de vérifier que la variable z est de la forme xx00 xxxx
- qui permet de vérifier que la variable z est de la forme xxxx xx10
- les bits n°0 **ET** n°1 de la variable z sont à 1
- le bit n°0 **OU** le bit n°1 de la variable z est à 1, en utilisant un test !=
- le bit n°0 **OU** le bit n°1 de la variable z est à 0, en utilisant un test !=
- les bits n°0 **ET** n°1 de la variable z sont à 0 (équivalent d'une fonction NOR).

3. Exercices supplémentaires (en fin de TD s'il reste du temps, évaluation, dans le RER, à la maison…)

Exercice I

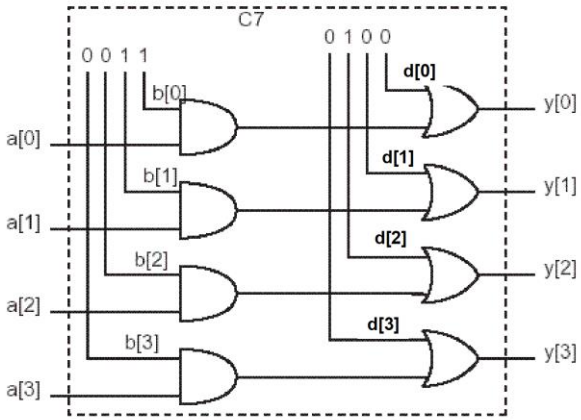
1) Quelle opération (mise à 1, mise à 0, inversion), est réalisée par ce bloc combinatoire ?
   Préciser les indices des bits modifiés par cette opération.

2) Que faut-il changer pour que le bloc inverse les bits n°2 et n°3, sans modifier les autres bits ?



Exercice II

Quelle est l'opération réalisée par le bloc ci-dessous ?



Exercice III

Ci-dessous un « morceau » de programme écrit en langage C :

```
char s1, s2, s3, s4, s5, s6, s7 ;
s1 = 0x00 ^ 0x01;
s2 = 0x01 ^ 0x01;
s3 = 0x00 ^ 0x0F;
s4 = 0x31 ^ 0x0F;
s5 = 0x12 >> 2;
s6 = 0x12 << 2;
s7 = 0x12 << 3;
```

Calculer les valeurs des s…. Donner le détail du calcul en binaire.

Exercice IV

Ci-dessous un « morceau » de programme écrit en langage C :

```
char s21, s22, s23, s24;
s21 = s21 ^ 0x01;
s22 = s22 ^ 0x0F;
```

1) Que peut-on dire de s21 et s22 après l'opération de masquage ?

2) Écrire l'instruction en C qui permet de :

- inverser le bit n°3 de s23, sans modifier les autres,
- inverser les bits n°4 et n°7 de s24, sans modifier les autres,

Exercice V

Donner le contenu le plus pertinent pour les messages à afficher :

```
if ((z & 0x18) != 0x00) {
   printf("....................................."); }
if ((z & 0x3E) != 0x00) {
   printf("....................................."); }
if ((z & 0xA1) != 0xA0) {
   printf("....................................."); }
```

Exercice VI

Pour un char z, écrire en langage C, l'expression qui permet de vérifier que :

- le bit n°3 de la variable z est à 0, en utilisant un test ==
- le bit n°3 de la variable z est à 0, en utilisant un test !=
- la variable z est de la forme xxx1 xxx0
- les bits n°1 **ET** n°5 de la variable z sont à 1
- le bit n°1 **OU** le bit n°5 de la variable z est à 1
- le bit n°1 **OU** le bit n°5 de la variable z est à 0
- les bits n°1 **ET** n°5 de la variable z sont à 0.

2° SEANCE DE TD

1. Ai-je bien compris le cours ?

a) A quoi servent les entrées logiques (TOR) d'un microcontrôleur ?

b) A quoi servent les entrées analogiques d'un microcontrôleur ?

c) A quoi servent les sorties logiques (TOR) d'un microcontrôleur ?

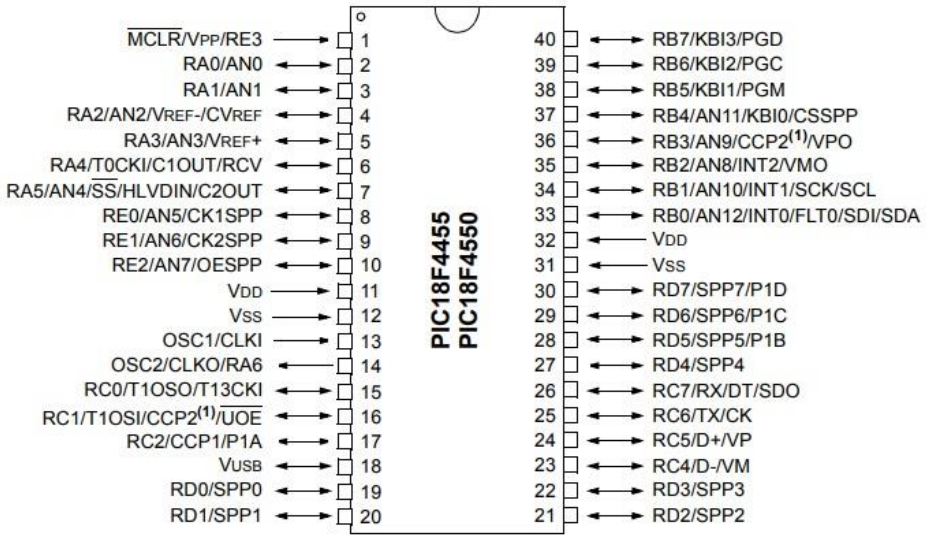
d) C'est quoi un port d'E/S pour un microcontrôleur ?

2. Exercices

Exercice 1 : Entrées / Sorties du microcontrôleur PIC18F4550

Parmi la multitude de microcontrôleurs disponibles sur le marché, à l'IUT pour le Semestre 1 nous utiliserons un petit modèle de chez Microchip, le PIC18F4550 (pour les autres semestres nous en étudierons d'autres, certains beaucoup plus « puissants », d'autres beaucoup plus « petits »…)

Brochage du microcontrôleur étudié :



Q1.1 Pour pouvoir fonctionner, le composant a besoin d'énergie et donc d'être alimenté. Donner les numéros et les noms des broches d'alimentation (aide : ce sont les mêmes noms que pour les AOP) (sauf que c'est 0/+5 VDC).

Q1.2 Toutes les autres broches peuvent assurer plusieurs fonctions, c'est à l'utilisateur de choisir. Certaines broches peuvent servir d'entrée analogique. Elles sont identifiées par un nom commençant par AN, suivi d'un chiffre. Combien le microcontrôleur en possède-t-il ?

Q1.3 Sauf deux cas particuliers (OSC1 et Vusb), toutes les broches peuvent servir d'entrées / sorties logiques. Pour les identifier, les E/S logiques ont un nom commençant par la lettre R, suivie d'une lettre indiquant le port, suivie d'un numéro indiquant l'emplacement dans le port.

Compléter les cases grisées des tableaux suivants :

	Port A							
Nom	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
N° broche	N'existe pas	14	7	6		4	3	2

	Port B							
Nom	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
N° broche	40		38	37	36	35	34	33

	Port C							
Nom	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
N° broche	26	25	24	23	N'existe pas	17	16	15

	Port D							
Nom	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0
N° broche	30	29	28	27	22	21	20	19

	Port E							
Nom	RE7	RE6	RE5	RE4	RE3	RE2	RE1	RE0
N° broche	N'existe pas	N'existe pas	N'existe pas	N'existe pas	1	10	9	8

Q1.4 Dans un port, les broches d'E/S logiques peuvent fonctionner chacune indépendemment en entrée ou en sortie. Par exemple RC0 peut être configurée en entrée et RC1 en sortie.
Pour faire définir soit Entrée ou soit Sortie, dans le programme il faut configurer un registre (mot de 8 bits) spécial appelé « TRIS ». Et cela pour chaque port. Il existe donc TRISA, TRISB, TRISC…
Chaque bit du registre TRIS correspond à la broche de même numéro sur le composant.
Un bit à 0 permet de définir une broche en sortie (0 ressemble à la lettre O de Ouput).
Un bit à 1 permet de définir une broche en entrée (1 ressemble à la lettre I de Input).

Exemple d'instruction dans un programme en langage C :

```
TRISA = 0xF0 ;
```

Le registre TRISA contiendra donc après l'exécution du programme :

	TRISA							
	1	1	1	1	0	0	0	0

Les broches du port A seront donc configurées de cette façon :

	PORT A							
Nom broche	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
Fonction	ENTREE	ENTREE	ENTREE	ENTREE	SORTIE	SORTIE	SORTIE	SORTIE

Exercice : Si un programme contient l'instruction TRISB = 0x03 ;
Quelle est la conséquence sur la configuration des broches ?

Q1.5 Quelle soit en entrée ou en sortie, l'état logique d'une broche est lié à la tension sur cette broche : 0 logique pour 0V
1 logique pour + 5V

L'état logique des broches est accessible dans le programme, dans un registre (mot de 8 bits) spécial appelé « PORT ». Et cela pour chaque port. Il existe donc PORTA, PORTB…
Chaque bit du registre PORT correspond à la broche de même numéro sur le composant.

Exemple d'états pour les broches du port A :

Nom broche	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
Etat	0V	0V	+5V	+5V	0V	0V	+5V	+5V

Le registre PORTA contiendra donc pendant l'exécution du programme :

PORTA							
0	0	1	1	0	0	1	1

Combiné à cela, on rappelle qu'une broche peut être en entrée ou en sortie. Sur l'exemple étudié (TRISA=0xF0), on retrouve donc sur le port A
Les broches de sorties RA0 et RA1 sont à +5V
Les broches de sorties RA2 et RA3 sont à 0V
Les broches d'entrée RA4 et RA5 sont à +5V
Les broches d'entrée RA6 et RA7 sont à 0V

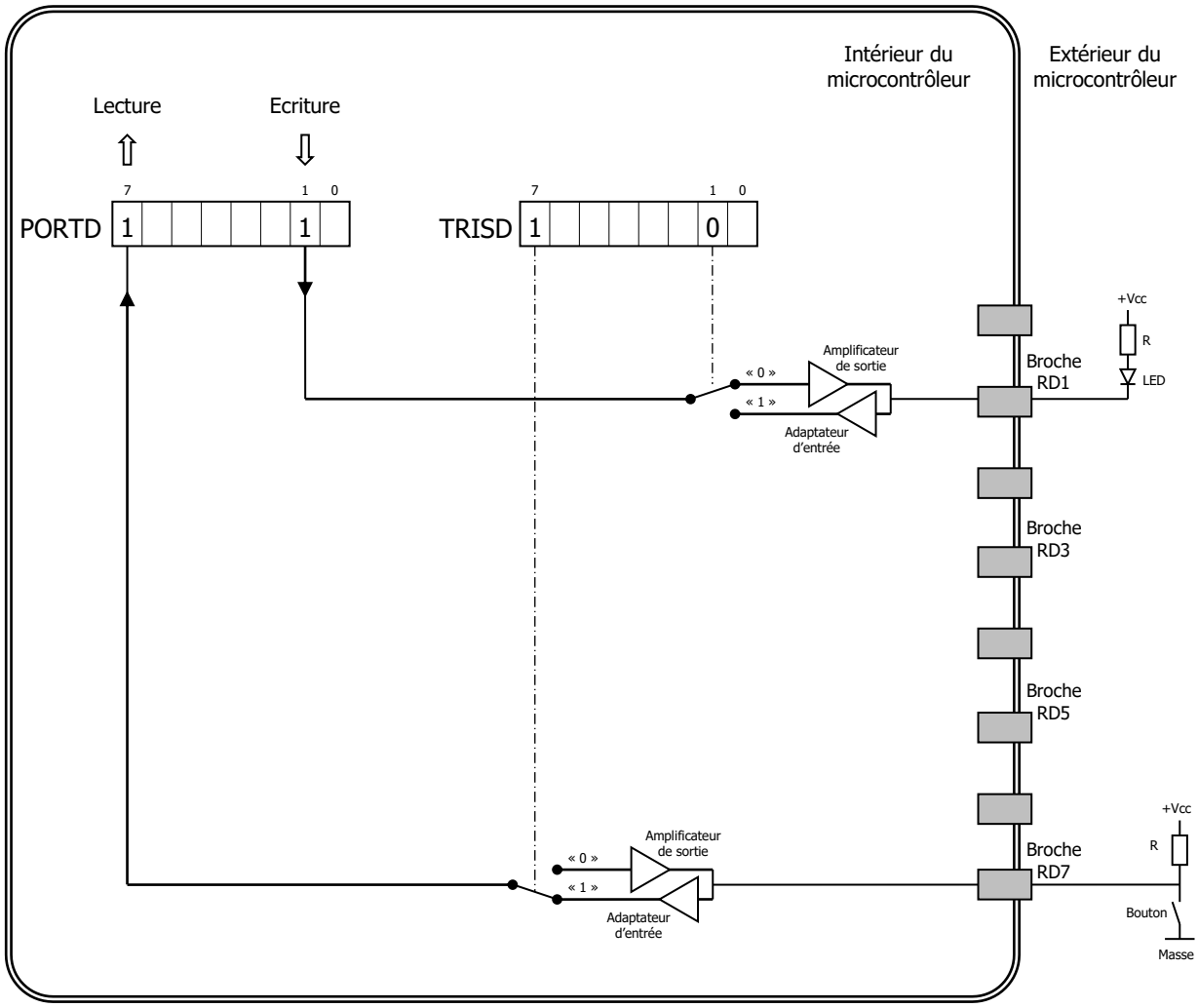
Exercice : Si un programme contient l'instruction TRISB = 0x22 ; et que pendant l'exécution du programme PORTB a pour valeur 0xF0, que peut-on dire sur l'état des broches ?

Q1.6 Lorsqu'une broche est en entrée, elle reçoit une tension (0 ou 5V) venue de l'extérieur du composant. C'est le « truc » branché qui impose la tension au microcontrôleur.
Dans le cas où l'on branche un bouton sur une entrée, il existe deux câblages possibles. Rappeler ces deux possibilités.

Q1.7 Lorsqu'une broche est en sortie, c'est elle, qui fournit une tension (0 ou 5V). C'est donc le microcontrôleur qui impose la tension au « truc » raccordé.
Dans le cas où l'on branche une LED sur une sortie, il existe deux câblages possibles. Rappeler ces deux possibilités.

Q1.8 La figure donnée page suivante montre une représentation simplifiée du fonctionnement d'un port d'entrée/sortie du microcontrôleur. On peut également voir sur cette figure un exemple de broche utilisée en entrée et une autre utilisée en sortie.

a) Quelle broche est utilisée en entrée ? Quelle broche est utilisée en sortie ?
b) A quoi sert l'instruction (langage C) suivante : `TRISD = TRISD | 0x80 ;`
Pourquoi utiliser un masque ?
c) A quoi sert l'instruction (langage C) suivante : `TRISD = TRISD & ~0x02 ;`
Pourquoi utiliser un masque ?
d) A quoi sert l'instruction (langage C) suivante : `PORTD = PORTD & ~ 0x02 ;`
Dans ce cas, au niveau de la broche, le potentiel est-il à +Vcc ou à la masse ? le courant sort-il du microcontrôleur ? rentre-t-il ? ou est-il nul ? En déduire l'état de la LED.
e) A quoi sert l'instruction (langage C) suivante : `PORTD = PORTD | 0x02 ;`
Dans ce cas, au niveau de la broche, le potentiel est-il à +Vcc ou à la masse ? le courant sort-il du microcontrôleur ? rentre-t-il ? ou est-il nul ? En déduire l'état de la LED.



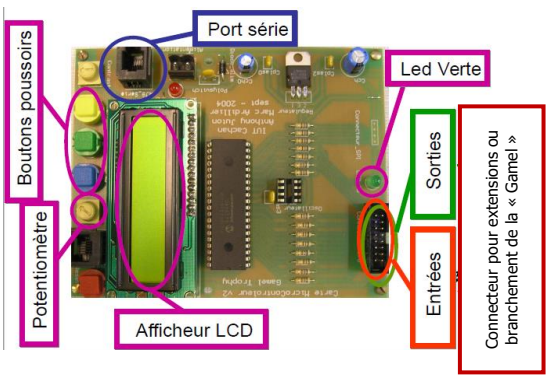
f) A quoi sert l'instruction (langage C) suivante : `if ( (PORTB & 0x80) == 0x00)`
Si la condition est vraie, le bouton est-il appuyé ? ou relâché ? Expliquer.

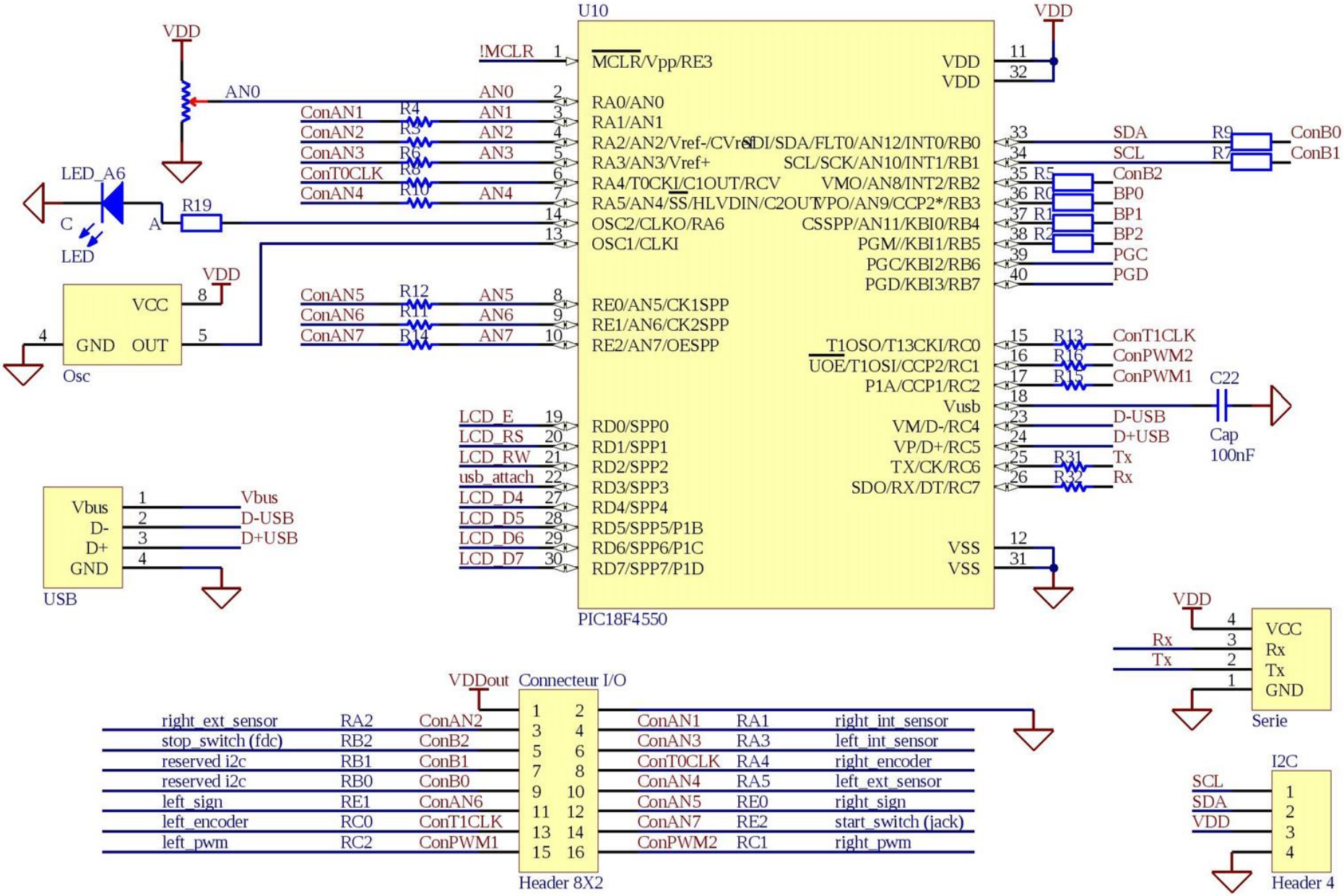
g) On désire câbler un nouveau bouton, câblage « pull-down », sur la broche RD5.
- Compléter le schéma ci-dessus.
- Donner l'instruction (langage C) pour configurer correctement TRISD.
- Donner l'instruction (langage C) qui permet de tester si le bouton est appuyé.

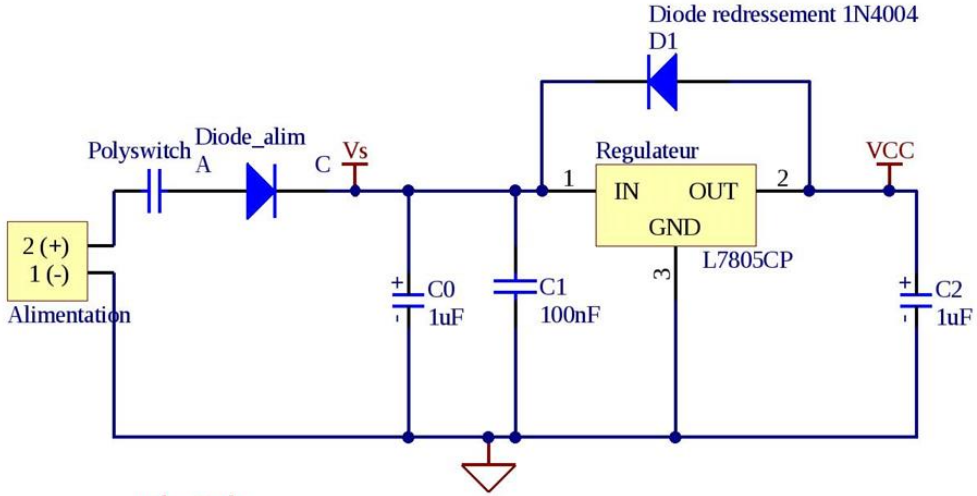
h) On désire câbler une nouvelle LED, sur la broche RD3, s'allumant si cette broche est à « 1 »
- Compléter le schéma ci-dessus.
- Donner l'instruction (langage C) pour configurer correctement TRISD.
- Donner l'instruction (langage C) pour allumer la LED.

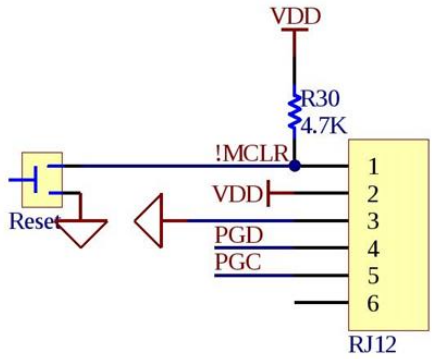
Exercice 2 : La carte microcontrôleur de l'IUT

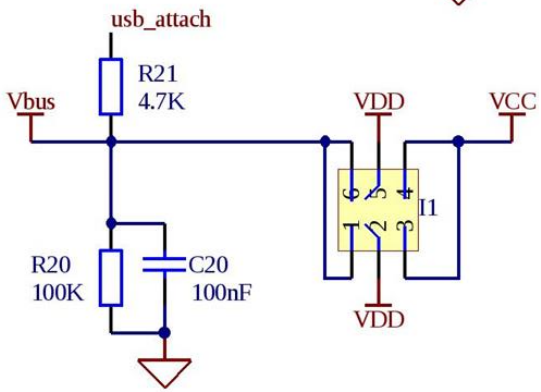
Pour les TP et l'E&R nous allons utiliser une carte à base du microcontrôleur PIC18F4550. Le schéma de cette carte est donné sur les 2 pages suivantes.

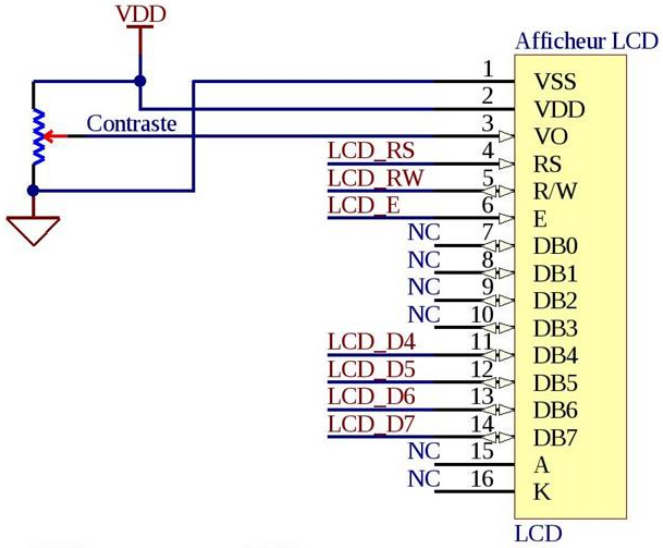


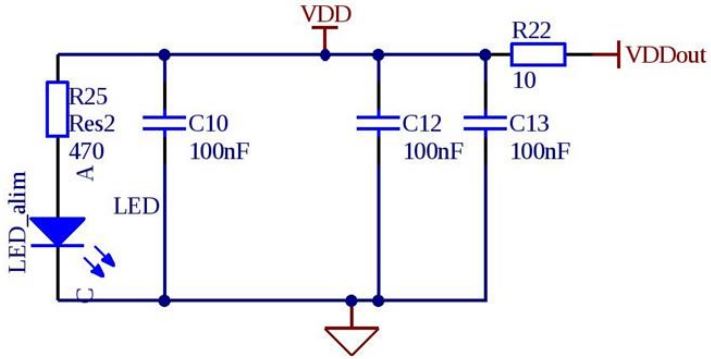

U10
PIC18F4550
!MCLR
MCLR/Vpp/RE3
VDD
VDD
VDD
VDD
AN0
RA0/AN0
ConAN1
R4
AN1
RA1/AN1
ConAN2
R5
AN2
RA2/AN2/Vref-/CVref
SDI/SDA/FLT0/AN12/INT0/RB0
SDA
R9
ConB0
ConAN3
R6
AN3
RA3/AN3/Vref+
SCL/SCK/AN10/INT1/RB1
SCL
R7
ConB1
ConT0CLK
R8
RA4/T0CKI/C1OUT/RCV
VMO/AN8/INT2/RB2
ConB2
ConAN4
R10
AN4
RA5/AN4/SS/HLVDIN/C2OUT
VPO/AN9/CCP2*/RB3
BP0
R19
OSC2/CLKO/RA6
CSSPP/AN11/KBI0/RB4
BP1
OSC1/CLKI
PGM//KBI1/RB5
BP2
PGC/KBI2/RB6
PGC
PGD/KBI3/RB7
PGD
LED_A6
LED
VDD
VCC
GND
OUT
Osc
ConAN5
R12
AN5
RE0/AN5/CK1SPP
ConAN6
R11
AN6
RE1/AN6/CK2SPP
ConAN7
R14
AN7
RE2/AN7/OESPP
T1OSO/T13CKI/RC0
ConT1CLK
UOE/T1OSI/CCP2/RC1
R16
ConPWM2
P1A/CCP1/RC2
R15
ConPWM1
C22
Vusb
Cap
100nF
LCD_E
RD0/SPP0
D-USB
LCD_RS
RD1/SPP1
VM/D-/RC4
D+USB
LCD_RW
RD2/SPP2
VP/D+/RC5
Tx
usb_attach
RD3/SPP3
TX/CK/RC6
R31
Rx
LCD_D4
RD4/SPP4
SDO/RX/DT/RC7
R32
LCD_D5
RD5/SPP5/P1B
LCD_D6
RD6/SPP6/P1C
VSS
VSS
LCD_D7
RD7/SPP7/P1D
Vbus
D-
D+
GND
USB
Vbus
D-USB
D+USB
VDD
VCC
Rx
Tx
GND
Serie
VDDout
Connecteur I/O
right_ext_sensor
RA2
ConAN2
ConAN1
RA1
right_int_sensor
stop_switch (fdc)
RB2
ConB2
ConAN3
RA3
left_int_sensor
reserved i2c
RB1
ConB1
ConT0CLK
RA4
right_encoder
reserved i2c
RB0
ConB0
ConAN4
RA5
left_ext_sensor
left_sign
RE1
ConAN6
ConAN5
RE0
right_sign
left_encoder
RC0
ConT1CLK
ConAN7
RE2
start_switch (jack)
left_pwm
RC2
ConPWM1
ConPWM2
RC1
right_pwm
Header 8X2
I2C
SCL
SDA
VDD
Header 4

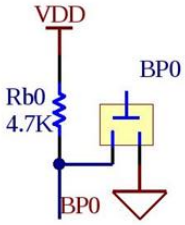

Diode redressement 1N4004
D1
Polyswitch
Diode_alim
A
C
Vs
Regulateur
IN
OUT
GND
L7805CP
1
2
3
VCC
2 (+)
1 (-)
Alimentation
C0
1uF
C1
100nF
C2
1uF

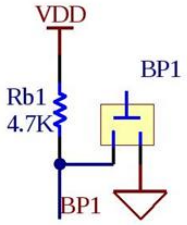

VDD
R30
4.7K
!MCLR
VDD
PGD
PGC
Reset
1
2
3
4
5
6
RJ12

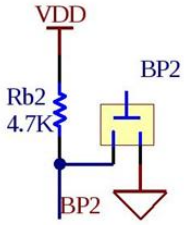

usb_attach
R21
4.7K
Vbus
VDD
VCC
I1
R20
100K
C20
100nF
VDD


VDD
Contraste
Afficheur LCD
1 VSS
2 VDD
3 VO
LCD_RS 4 RS
LCD_RW 5 R/W
LCD_E 6 E
NC 7 DB0
NC 8 DB1
NC 9 DB2
NC 10 DB3
LCD_D4 11 DB4
LCD_D5 12 DB5
LCD_D6 13 DB6
LCD_D7 14 DB7
NC 15 A
NC 16 K
LCD


VDD
R22
10
VDDout
R25
Res2
470
C10
100nF
C12
100nF
C13
100nF
LED_alim
LED
A
C


VDD
Rb0
4.7K
BP0
BP0


VDD
Rb1
4.7K
BP1
BP1


VDD
Rb2
4.7K
BP2
BP2

Analyse du schéma de la carte :

Q2.1 Que trouve-t-on de branché sur la broche 2 du microcontrôleur ?
Cette broche est-t-elle utilisée en entrée ? en sortie ? logique ? analogique ?

Q2.2 Que trouve-t-on de branché sur la broche 14 du microcontrôleur ?
Cette broche est-t-elle utilisée en entrée ? en sortie ? logique ? analogique ?
Donner l'instruction (langage C) pour configurer correctement TRISA.
Donner l'instruction (langage C) pour allumer la LED.

Calculer la valeur de la résistance (R19), sachant que la DEL est allumée correctement si elle est traversée par un courant I_F = 5 mA et présente une chute de tension à ses bornes de V_F = 1V.

Q2.3 Que trouve-t-on de branché sur les broches 36, 37, 38 du microcontrôleur ?
Ces broches sont-elles utilisées en entrée ? en sortie ? logique ? analogique ?
Donner l'instruction (langage C) pour configurer correctement TRISB.

Avec les valeurs de Rb0, Rb1, Rb2 données sur le schéma, calculer le courant circulant dans celles-ci. Ce courant circule-t-il lorsque les boutons sont passants (appuyés) ou ouverts (relâchés) ?

Donner l'instruction (langage C) pour tester en une seule fois si BP0 et BP1 sont appuyés en même temps que BP2 est relâché.

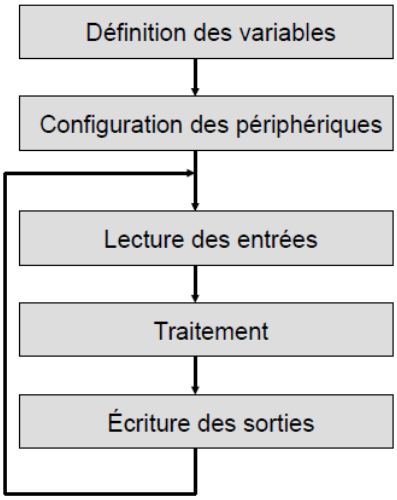
3. Exercices supplémentaires (en fin de TD s'il reste du temps, évaluation, dans le RER, à la maison…)

Commencer le TD3

3° SEANCE DE TD

1. Ai-je bien compris le cours ?

Les programmes d'acquisition et de traitement de données avec un microcontrôleur respectent en général l'organigramme ci-contre.



a) Pourquoi est-il nécessaire de faire en boucle les opérations de [Lecture des entrées /traitement/ écriture des sorties] ?

b) Les instructions agissant sur les registres TRIS sont-elles dedans cette boucle ?

2. Exercices

Exercice 1 : Programmation de traitement combinatoire

On donne le programme (Langage C) ci-dessous :

```
1 #include <xc.h>
2
3 void main(void)
4 {
5      char bp0, led ;
6
7      TRISA = 0xBF ;
8      TRISB = 0xFF ;
9
10     while(1)
11     {
12              bp0 = PORTB & 0x08 ;
13
14              if (bp0 == 0x08)  /* si bouton poussoir relâché */
15              {
16                  led = 0x00 ;
17              }
18              else
19              {
20                  led = 0x40 ;  /* pour allumer la led */
21              }
22
23              PORTA = led;
24     }
25}
```

Q1.1. Pourquoi `TRISA, TRSIB, PORTA, PORTB` ne sont pas déclarés comme variables ?

Q1.2. Ajouter des commentaires aux lignes 7 et 8, pour indiquer le mode de fonctionnement des broches concernées.

Q1.3. Quelles lignes d'instruction sont réalisées indéfiniment en boucle ?
Quelle instruction est utilisée pour réaliser cette boucle ?

Q1.4. Ajouter un commentaire à la ligne 12. Sur quelle broche du microcontrôleur se trouve vraisemblablement le bouton bp0 ?

Q1.5. Déduire de la ligne 14, si le câblage du bouton est de type « pull-up » ou « pull-down ».

Q1.6. Selon les lignes 20 et 23, déduire sur quelle broche du microcontrôleur se trouve câblée la led et son type de câblage (tirage masse ou +Vcc).

Q1.7. Globalement que va réaliser ce programme ?

Q1.8. Encadrer sur le programme les 5 zones correspondantes aux 5 étapes de l'organigramme général, donné dans le cours et rappelé en début d'exercice.

Q1.9. On remplace l'instruction de la ligne 14 par `if (bp0 != 0x00)`
Le fonctionnement est-il différent ?

Q1.10. La fin du programme est remplacée par :

```
15              {
16                  PORTA = PORTA & ~0x40 ; /* pour éteindre la led */
17              }
18              else
19              {
20                  PORTA = PORTA | 0x40 ;  /* pour allumer la led */
21              }
22
23
24     }
25    }
```

Le fonctionnement est-il identique ?
Qu'apporte l'utilisation d'opérations de masquage ?

Q1.11. Un bouton bp1, câblage « pull-up », est rajouté sur RB4. Modifier le programme pour que la led s'allume, si et seulement si, les deux boutons sont activés.

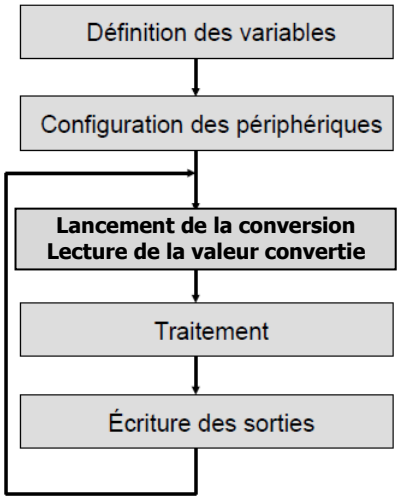
Q1.12. Modifier ce programme, de manière à vérifier : « La led est allumée si et seulement si l'un des deux boutons au moins est enfoncé ».

Q1.13. Une nouvelle led, type de câblage identique à la première, est rajoutée sur RA1. Modifier le programme pour cette nouvelle led prenne toujours l'état inverse de la première.

Exercice 2 : Programme avec conversion analogique-numérique

Les entrées d'un microcontrôleur peuvent être des entrées Tout ou Rien, comme les boutons poussoirs, mais peuvent être aussi des entrées analogiques (capteur de température, potentiomètre…). La phase de lecture des entrées est alors légèrement modifiée.

Un programme réalisant la quantification de la tension analogique sur l'entrée AN0, respecte l'organigramme ci-contre.



Voici un premier programme, qui fait appel à une bibliothèque de fonctions déjà écrites.

```
#include <xc.h>
#include "iut_adc.h"

void main(void)
{
    int n ;

    /* initialisation du convertisseur, ne se fait qu'une seule
fois */
    adc_init(0);

    while (1)
    {
        /* Lancement de la conversion et lecture de
l'entrée AN0 */
        n = adc_read(0);
    }
}
```

1) Reconnaître dans ce programme, les différentes parties de l'organigramme (il en manque…)

2) A quoi sert le `#include "iut_adc.h"`

3) A quoi sert le rebouclage ? Que se passerait-il s'il n'était pas là ?

4) Modifier ce programme pour :

- calculer la tension ve (type ***float***) correspondant au résultat de la conversion
- qu'une DEL connectée sur le bit 6 du port A s'allume si la tension ve est supérieure à 2 Volts

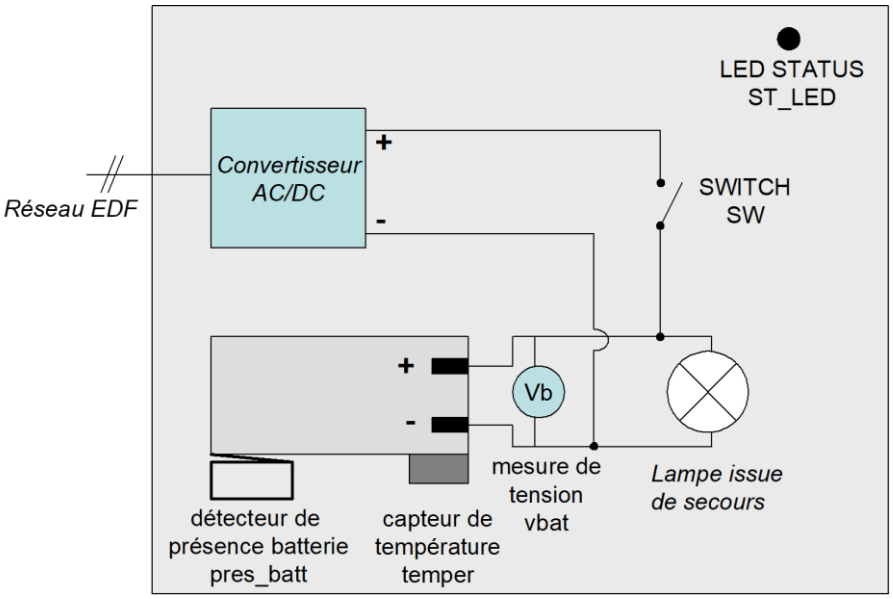
Exercice 3 : Etude de cas – Exercice de synthèse

L'étude concerne un dispositif de type « BAES », un Bloc Autonome d'Eclairage et de Sécurité, permettant de guider les personnes vers les issues de secours en cas d'urgence.

Le dispositif est constitué :

⇨ D'une batterie qui alimente la lampe, même en cas de panne d'électricité.

⇨ D'un convertisseur AC/DC permettant la recharge de la batterie à partir du secteur.

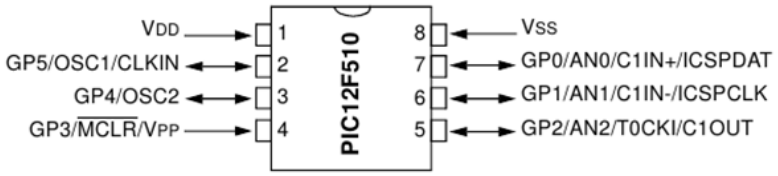


⇨ Liste des capteurs, actionneurs et voyant :

- Un capteur de température accolé à la batterie, qui délivre un signal appelé ***temper***, proportionnel à la température de la batterie. 0,1 V correspond à 1°C.
- Un détecteur de présence batterie : Le signal ***pres_batt*** vaut 1 si une batterie est présente. ***pres_batt*** vaut 0 sinon. Le signal est produit par un contact, associé à une résistance de tirage. Le contact est fermé si la batterie est présente.
- Un diviseur de tension qui renvoie une tension ***vbat*** égale à un tiers de la tension de la batterie.
- Une led appelée « Led Status » qui indique l'état du système. Si le signal ***ST_LED*** sortant du microcontrôleur vaut 1, la led est allumée, sinon elle est éteinte.
- Un contact (switch) permet de relier ou non le convertisseur AC/DC à la batterie. Ce contact est commandé électriquement (on parle de contacteur), à partir du microcontrôleur. Si l'ordre ***SW*** vaut 1, le contact est fermé, sinon il est ouvert.

⇨ Le fonctionnement est géré par un microcontrôleur. Pour des raisons de coût et d'encombrement, il a été choisi d'utiliser le « tout petit » microcontrôleur PIC12F510 (moins de 1 euro).

Brochage du microcontrôleur :



Quelques remarques :

- Le microcontrôleur possède une horloge interne. Il n'y a pas besoin de rajouter un quartz externe. Les 6 broches d'entrées/sorties sont disponibles. (VDD est reliée à l'alimentation et VSS à la masse).
- Chaque broche est configurée en entrée TOR par défaut.
- GP0..GP5 indiquent des entrées/sorties TOR (Comme RB3, RA6… de l'autre microcontrôleur). On peut les configurer en entrée ou en sortie de la même manière en fixant la valeur de la variable ***TRISGP***. On peut lire ou écrire les entrées/sorties TOR avec la variable ***PORTGP***.
- AN0, AN1 et AN2 indiquent des entrées qui peuvent être configurées en entrées analogiques en utilisant la bibliothèque "`iut_adc.h`".

`adc_init(0);`	pour configurer AN0 en entrée analogique
`adc_init(1);`	pour configurer AN0 et AN1 en entrées analogiques
`adc_init(2);`	pour configurer AN0, AN1 et AN2 en entrées analogiques

Travail demandé :

1) Recenser les entrées/sorties nécessaires et indiquer si elles sont logiques ou analogiques.
2) Associer chaque signal à une broche du microcontrôleur.
3) Dessiner le schéma de câblage de l'ensemble.
4) Proposer un programme répondant au cahier des charges suivant :
   - Lorsque la batterie est présente ET que la température de celle-ci est inférieure à 40°C ET que la tension est inférieure à 12V, on ferme le switch pour charger la batterie. La led status reste éteinte dans ce cas.
   - Lorsque la batterie est présente ET que la température de celle-ci est inférieure à 40°C ET que la tension est supérieure à 13V, on ouvre le switch pour éviter une usure précoce de la batterie due à un courant de charge permanent. La led status reste éteinte dans ce cas.
   - Lorsque la batterie est présente ET que la température de celle-ci est supérieure à 40°C, on ouvre le switch et on allume la led pour signaler un défaut de batterie.
   - Lorsque la batterie est absente, on ferme le switch, de sorte d'alimenter la lampe d'issue de secours à partir du convertisseur AC/DC. On signale l'absence de batterie en allumant la led.
5) Ajouter la fonctionnalité suivante à votre programme :

   Si la batterie indique une tension inférieure à 10V, on considère qu'elle doit être remplacée. Pour l'indiquer, la led clignote avec une période de 1 s.

   Pour cette question, on pourra utiliser la fonction `void delay_ms(short delay)` qui provoque une attente de la durée indiquée en paramètre (en ms).