Date : **18/11/21**

université PARIS-SACLAY IUT DE CACHAN Département GEii 2	**Informatique Industrielle 2** # Travaux dirigés	**S1** **Techniques Numériques**

Auteur(s) : Y. G.

Référence : II2 – TD – version 2021

Un aide mémoire des fonctions des bibliothèques IUT pour la carte microcontrôleur GamelTrophy utilisée en E&R est fourni à la fin du texte de ce TD.

PARTIE A – RÉVISIONS

EXERCICE I - TESTS ET MASQUES

La variable *z* est de type *char*.

Expliquer et justifier le sens des messages des lignes de code ci-dessous.

```
if ((z & 0x40) != 0) { printf("z6 est à 1"); }
if ((z & 0x02) == 0) { printf("z1 est à 0"); }
if ((z & 0x3C) == 0x3C) { printf("z est de la forme xx11 11xx"); }
if ((z & 0x96) == 0x14) { printf("z est de la forme 0xx1 x10x"); }
if ((z & 0x81) != 0) { printf("z7 est à 1 OU z0 est à 1"); }
if ((z & 0x60) != 0x40) { printf("z6 est à 0 OU z5 est à 1"); }
```

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

```
if ((z & 0x40) == 0) { printf("....................................."); }
if ((z & 0x01) != 0) { printf("....................................."); }
if ((z & 0xA0) == 0x80) { printf("....................................."); }
if ((z & 0x0A) != 0) { printf("....................................."); }
if ((z & 0x11) != 0x10) { printf("....................................."); }
```

Attention, celles-ci sont plus compliquées :

```
if ((z & 0x43) != 0x00) { printf("....................................."); }
if ((z & 0xE3) != 0x00) { printf("....................................."); }
if ((z & 0x1B) != 0x0B) { printf("....................................."); }
```

Écrire en langage C, l'expression

- qui permet de vérifier que la variable z est de la forme xxx1 xxxx
- qui permet de vérifier que la variable z est de la forme xx0x 0xxx
- qui permet de vérifier que la variable z est de la forme x101 x0xx

EXERCICE II - CONVERTISSEUR ANALOGIQUE-NUMÉRIQUE

On dispose d'un Convertisseur Analogique Numérique 10 bits dont la tension d'entrée peut varier de 0 à 5Volt.

1. Tracer sa fonction de transfert (courbe donnant N en fonction de Ve)

2. On applique à l'entrée du CAN une tension de 1,8 Volt. Quel nombre (en base 10 et en binaire) obtient-on à la sortie du CAN ? Même question si on applique une tension de 4,2 Volt.

3. La tension Ve est issue d'un capteur de température qui délivre une tension de 50 mV/°C
Tracer la courbe donnant N en fonction de la température.
Quelle est la plus petite température mesurable ?
Quelle est la plus grande température mesurable ?
Quelle est le plus petit écart de température mesurable ?

4. Le code ci-dessous permet de récupérer et d'afficher la valeur du potentiomètre branché sur la voie 0 du convertisseur analogique-numérique de la carte microcontrôleur utilisée en E&R. Ce potentiomètre délivre une tension comprise entre 0 et 5V.

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

void main(void)
{
      int pot;

      lcd_init();
      adc_init(0);

      while (1)
      {
            pot = adc_read(0);
            lcd_position(0, 0);
            lcd_printf("%4d", pot);
      }
}
```

Commenter ce code en utilisant la documentation des fonctions fournie à la fin du texte de ce TD. Le format `"%4d"` permet d'afficher un `int` sur 4 caractères. Expliquer ce choix de format par rapport à la résolution du CAN.

5. Le capteur de température est branché sur la voie 1 (AN1) du convertisseur analogique-numérique de la carte microcontrôleur utilisée en E&R. Écrire un programme qui affiche la valeur de température mesurée sur l'écran LCD de la carte microcontrôleur.

EXERCICE III - PWM

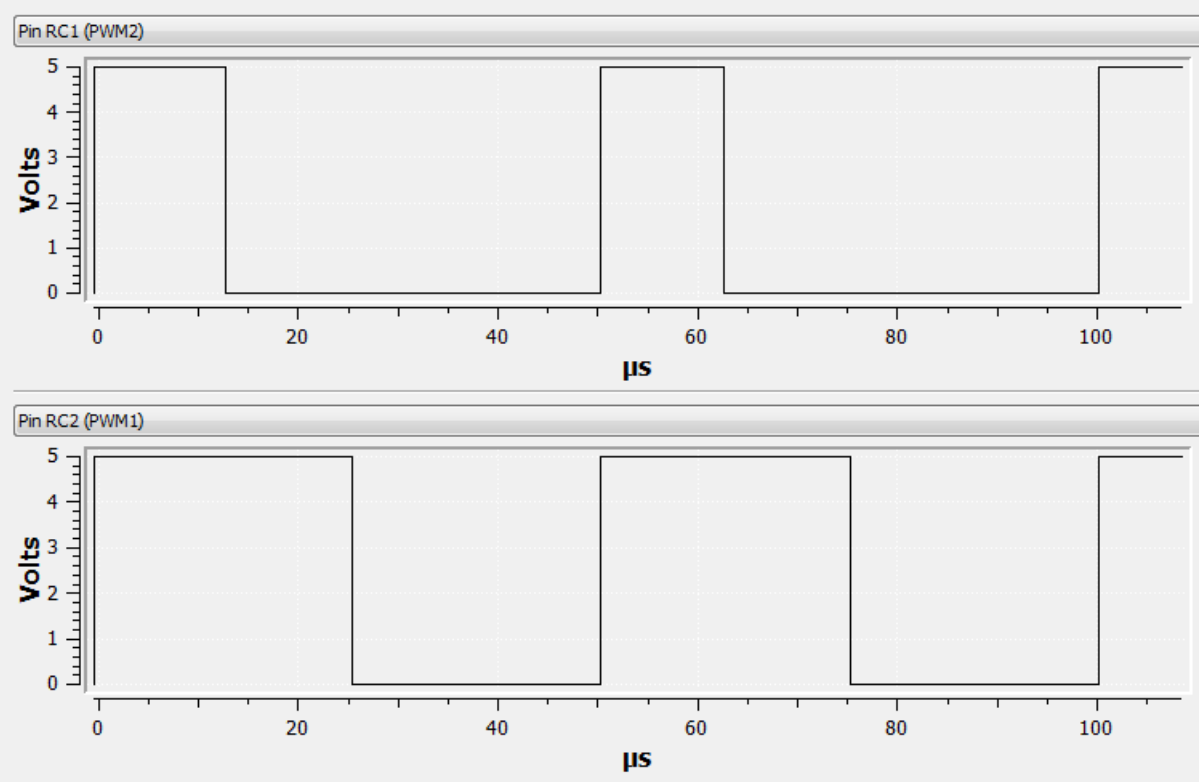
```
#include <xc.h>
#include "iut_pwm.h"

void main(void)
{
    pwm_init(149, 2);

    pwm_setdc1(300);
    pwm_setdc2(150);

    while (1)
    {
    }
}
```

Le code précédent produit les signaux suivants :



Commenter ce code en utilisant la documentation des fonctions fournie à la fin du texte de ce TD. Expliquer en particulier les valeurs de la fréquence et des rapports cycliques obtenues.

PARTIE B – MACHINES À ÉTATS

EXERCICE IV - MACHINE À CAFÉ

Une machine permet de faire soit un café, soit un chocolat. Un gobelet est placé manuellement sous la machine et l'utilisateur appuie sur le bouton « café » ou sur le bouton « chocolat ».

La machine dispose des capteurs suivants :

- un bouton pour faire un café associé à la variable `boutonCafe` qui prend la valeur 1 quand on appuie sur le bouton et 0 sinon
- un bouton pour faire un chocolat associé à la variable `boutonChocolat` qui prend la valeur 1 quand on appuie sur le bouton et 0 sinon
- un détecteur de présence d'un gobelet associé à la variable `gobeletEnPlace` qui prend la valeur 1 si un gobelet est présent et 0 sinon
- un détecteur de quantité de poudre versée associé à la variable `finPoudre` qui prend la valeur 1 si suffisamment de poudre a été versée et 0 sinon
- un détecteur de quantité d'eau chaude versée associé à la variable `finEauChaude` qui prend la valeur 1 si suffisamment d'eau chaude a été versée et 0 sinon

La machine dispose des actionneurs suivants :

- une vanne pour verser la poudre de café associée à la variable `poudreCafe` qui doit valoir 1 pour ouvrir la vanne et 0 pour la fermer
- une vanne pour verser la poudre de chocolat associée à la variable `poudreChocolat` qui doit valoir 1 pour ouvrir la vanne et 0 pour la fermer
- une vanne pour verser l'eau chaude associée à la variable `vanneEauChaude` qui doit valoir 1 pour ouvrir la vanne et 0 pour la fermer

Cette machine dispose également d'un afficheur.

Elle est contrôlée par le code suivant :

```
void main()
{
    // Définitions et initialisations des variables
    char etat = 0;
    char boutonCafe, boutonChocolat;
    char gobeletEnPlace, finPoudre, finEauChaude;
    char poudreCafe, poudreChocolat, vanneEauChaude;

    // Configuration des périphériques

    while (1) {

        // lecture des capteurs

        printf("Etat %d - ", etat);
        switch (etat) {
        case 0 :
            printf("Attente gobelet en place\n");
            if (gobeletEnPlace == 1) {
                etat = 1;
            }
            break;
        case 1 :
            printf("Attente ordre\n");
            if (boutonCafe == 1) {
                etat = 2;
            } else if (boutonChocolat == 1) {
                etat = 3;
            }
            break;
        case 2 :
            printf("Poudre café\n");
            poudreCafe = 1;
            if (finPoudre == 1) {
                etat = 4;
            }
            break;
        case 3 :
            printf("Poudre chocolat\n");
            poudreChocolat = 1;
            if (finPoudre == 1) {
                etat = 4;
            }
            break;
        case 4 :
            poudreCafe = 0;
            poudreChocolat = 0;
            printf("Eau chaude\n");
            vanneEauChaude = 1;
            if (finEauChaude == 1) {
                etat = 5;
            }
            break;
        case 5 :
            vanneEauChaude = 0;
            printf("Attente gobelet retiré\n");
```

```
            if (gobeletEnPlace == 0) {
                etat = 0;
            }
            break;
        default :
            // ce cas ne devrait jamais se produire
            etat = 0;
        } // fin du switch
    } // fin du while
} // fin du main
```

Dessiner la machine à états qui correspond à ce code.

EXERCICE V - CARTE MICRO-CONTRÔLEUR (1)

On donne le code suivant pour la carte micro-contrôleur utilisée en étude et réalisation.

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

void main(void)
{
    char etat = 0;
    char bp0, bp1, bp2, led;
    int pot;

    TRISA = 0xBF;
    TRISB = 0xFF;
    adc_init(0);
    lcd_init();

    while (1)
    {
        lcd_position(0, 0);
        lcd_printf("Etat = %d", (int)etat);
        bp0 = PORTB & 0x08;
        bp1 = PORTB & 0x10;
        bp2 = PORTB & 0x20;
        pot = adc_read(0);
        switch (etat) {
            case 0 :
                led = 0;
                if (bp0 == 0) {
                    etat = 1;
                }
                break;
            case 1 :
                led = 1;
                if (bp1 == 0) {
                    etat = 2;
                } else if (bp2 == 0) {
                    etat = 3;
                }
                break;
            case 2 :
                led = 1;
                if (pot > 768) {
                    etat = 4;
                }
```

```
                break;
            case 3 :
                led = 1;
                if (pot < 256) {
                    etat = 4;
                }
                break;
            case 4 :
                led = 0;
                if ((pot > 410) && (pot < 614)) {
                    etat = 0;
                }
                break;
            default :
                etat = 0;
        }
        if (led != 0) {
            PORTA = PORTA | 0x40;
        } else {
            PORTA = PORTA & ~0x40;
        }
    }
}
```

Commenter les lignes de code.

Dessiner la machine à états réalisée par ce programme.

EXERCICE VI - CARTE MICRO-CONTRÔLEUR (2)

On donne le code suivant pour la carte micro-contrôleur utilisée en étude et réalisation.

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

void main(void)
{
    char etat = 0;
    char bp0, bp1;
    int pot;

    TRISA = 0xBF;
    TRISB = 0xFF;
    adc_init(0);
    lcd_init();

    while (1)
    {
        bp0 = PORTB & 0x08;
        bp1 = PORTB & 0x10;
        pot = adc_read(0);
        switch (etat) {
            case 0 :
                lcd_position(0, 0);
                lcd_printf("Menu 1  ");
                if (pot > 350) {
                    etat = 1;
                } else if (bp0 == 0) {
                    etat = 10;
```

```
                }
                break;
            case 1 :
                lcd_position(0, 0);
                lcd_printf("Menu 2  ");
                if (pot > 670) {
                    etat = 2;
                } else if (pot < 320) {
                    etat = 0;
                } else if (bp0 == 0) {
                    etat = 20;
                }
                break;
            case 2 :
                lcd_position(0, 0);
                lcd_printf("Menu 3  ");
                if (pot < 650) {
                    etat = 1;
                } else if (bp0 == 0) {
                    etat = 30;
                }
                break;
            case 10 :
                lcd_position(0, 0);
                lcd_printf("Select 1");
                if (bp1 == 0) {
                    etat = 0;
                }
                break;
            case 20 :
                lcd_position(0, 0);
                lcd_printf("Select 2");
                if (bp1 == 0) {
                    etat = 1;
                }
                break;
            case 30 :
                lcd_position(0, 0);
                lcd_printf("Select 3");
                if (bp1 == 0) {
                    etat = 2;
                }
                break;
            default :
                etat = 0;
        }
    }
}
```

Commenter les lignes de code.

Dessiner la machine à états réalisée par ce programme.

Modifier la machine à états pour ajouter un « Menu 4 » et écrire le code correspondant.

Modifier la machine à états pour ajouter deux sous-menus au « Menu 2 » et écrire le code correspondant.

EXERCICE VII - GAMELLE : DÉPART AU JACK ET ARRÊT SUR OBSTACLE

La fiche jack est connectée sur la broche 1 du port B en pull-up avec 10 kΩ. On récupère un 0 logique si la fiche jack est enfoncée et un 1 logique sinon.

Le capteur de fin de course est connecté sur la broche 2 du port B en pull-up avec 10 kΩ. On récupère un 0 logique si le robot percute un obstacle et un 1 logique sinon.

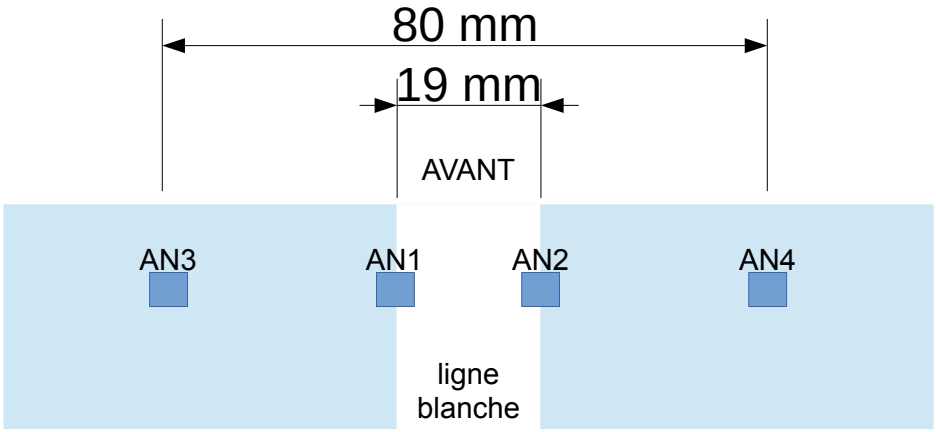
Les moteurs seront commandés en mode PWM avec une fréquence de 20 kHz et un rapport cyclique de 0,25. La commande du hacheur du moteur gauche est reliée à PWM1 et celle du moteur droit à PWM2.

Compléter le code de la machine à états de la partie 2 du cours.

EXERCICE VIII - GAMELLE : CAPTEURS DE DÉTECTION DE LIGNE

On ajoute à la Gamelle de l'exercice précédent une carte électronique avec 4 capteurs CNY70 disposés comme ci-dessous :

Donner la liste des capteurs.



vue de dessus

Proposer un programme qui affiche sur l'écran LCD les valeurs de tous les capteurs.

Donner la liste des actionneurs.

Proposer un programme qui active le moteur gauche quand on appuie sur BP0, qui active le moteur droit quand on appuie sur BP1 et qui active les deux moteurs quand on appuie sur BP2. Le rapport cyclique envoyé aux hacheurs (préactionneurs des moteurs) sera contrôlé par le potentiomètre branché sur AN0.

EXERCICE IX - GAMELLE : SUIVI DE LIGNE

On utilise une Gamelle équipée de l'ensemble des capteurs et des actionneurs des 2 exercices précédents.

Au-dessus de la moquette, les capteurs fournissent une valeur inférieure à 250 et au-dessus de la ligne blanche, les capteurs fournissent une valeur supérieure à 700.

Compléter la machine à états de l'exercice VII pour que la Gamelle puisse suivre la ligne.

Écrire le code correspondant.

Réfléchir à une machine à états pour la prise de raccourci.

Aide mémoire des fonctions des bibliothèques IUT pour la carte microcontrôleur GamelTrophy

Fonctions pour l'écran LCD

`void lcd_init(void);`

Cette fonction initialise l'écran LCD. Il faut absolument l'appeler dans votre programme avant tout emploi d'une autre fonction de la bibliothèque. Une seule exécution de cette fonction est suffisante.

`void lcd_position(char ligne, char colonne);`

Cette fonction positionne le curseur pour l'affichage. Le coin supérieur gauche de l'afficheur est considéré à la ligne 0 et à la colonne 0. Au prochain appel d'une fonction d'affichage comme *lcd_printf* ou *lcd_putc*, le premier caractère s'inscrira à l'endroit désigné par *lcd_position*.

`void lcd_putc(char lettre);`

Cette fonction affiche un caractère à la position du curseur et décale le curseur d'une case vers la droite pour le prochain affichage.
Les caractères spéciaux suivants sont interprétés : `'\n'` pour passer à la ligne, `'\f'` pour effacer l'écran et `'\b'` pour reculer d'une case

`void lcd_clear(void);`

Efface l'écran.

`void lcd_printf(const rom char *f, ...);`

Cette fonction permet d'effectuer des affichages sur l'écran LCD avec la même syntaxe que la fonction *`printf`* usuelle du langage C. Rappel des formats les plus couramment utilisés :

- `%d` pour une afficher en base 10 une variable de type `int`
- `%x` pour afficher en hexadécimal une variable de type `int`
- `%f` pour afficher une variable de type `float`
- `%s` pour afficher une chaine de caractères

Fonctions pour le convertisseur analogique-numérique

`void adc_init(char numero_dernier_canal);`

Cette fonction initialise le convertisseur analogique-numérique (temps d'acquisition et temps de conversion). Vous pouvez utiliser le nombre d'entrées analogiques que vous souhaitez, à condition de les prendre successives, à partir de AN0 et de donner en paramètre de la fonction *adc_init* le numéro du dernier canal que vous utilisez.

`int adc_read(char numero_canal);`

Cette fonction déclenche la conversion analogique-numérique sur le canal indiqué en paramètre. Elle renvoie ensuite cette valeur sous forme d'entier 16 bits. Seuls les 10 bits de poids faible sont utilisés (les 6 bits de poids fort sont toujours nuls). Le temps de conversion est d'environ 25µs.

Fonctions pour les sorties PWM

`void pwm_init(char period, char nb_canaux);`

Cette fonction initialise les sorties PWMs. C'est cette fonction qui configure le timer2 pour le fonctionnement des PWMs.
Si nb_canaux = 2 alors les deux sorties sont activées, sinon seule la sortie PWM1 est activée. Vous entrerez en paramètre de cette fonction la période de vos signaux PWMs, en nombre de pas de **333 ns**. Le paramètre *period* est un entier de 8 bits. Pour Fosc = 48 MHz (en utilisant *iut_init.c*), la fréquence de fonctionnement des PWMs est donnée par la formule :

$$f = (3.10^6 / (period+1))$$

Par exemple, si *period* = 149 alors la fréquence f sera de 20 kHz.

`void pwm_setdc1(unsigned int cycles_etat_haut);`

Cette fonction définit le rapport cyclique de PWM1 (C2).

`void pwm_setdc2(unsigned int cycles_etat_haut);`

Cette fonction définit le rapport cyclique de PWM2 (C1).

Pour ces deux dernières fonctions, *cycles_etat_haut* représente le temps à l'état haut, en nombre de pas de **83,3 ns**. *cycles_etat_haut* est un entier de 10 bits.
On peut calculer le rapport cyclique en fonction du paramètre *period* spécifié pour la fonction *pwm_init* avec la formule :

$$\text{rapport cyclique} = cycles\_etat\_haut / \mathbf{(}\ 4 \times (\ period + 1)\ \mathbf{)}$$

Par exemple, si *period* = 149 alors un rapport cyclique de 1 correspond à *cycles_etat_haut* = 600, un rapport cyclique de 0,5 correspond à *cycles_etat_haut* = 300 et un rapport cyclique de 0,25 correspond à *cycles_etat_haut* = 150.