Université Djilali BOUNAAMA, Khemis Miliana
مليانة خميس بونعامة جيلالي جامعة
Faculté des Sciences et de la Technologie
والتكنولوجيا العلوم كلية

Série d'exercices 2: Les entrées/sorties digitales et analogiques

Exercice 1 : Configuration d'une broche en sortie pour faire clignoter une LED

On utilise un microcontrôleur AVR (par exemple ATmega328P). On connecte une LED avec une résistance sur la broche PD5.

1. Écrire le code de configuration pour que PD5 soit en sortie.
2. Écrire une fonction qui fait clignoter la LED avec un délai de 500 ms.

Exercice 2 : Utilisation d'un bouton poussoir avec pull-up interne

Énoncé :
Un bouton poussoir est connecté entre la broche PB0 et la masse.

1. Configurer PB0 en entrée avec pull-up interne activé.
2. Lire l'état du bouton et allumer une LED sur PB5 quand le bouton est appuyé (état logique 0).
3. Expliquer pourquoi on utilise un pull-up.

Exercice 3 : Gestion des rebonds (debouncing) logiciel

Un bouton sur PD2 doit incrémenter un compteur affiché sur 4 LEDs (PD4 à PD7). Le compteur va de 0 à 15. Implémenter un anti-rebond par temporisation (20 ms).

Exercice 4 : Acquisition analogique avec CAN (résolution 10 bits)

On utilise le CAN du même microcontrôleur sur le canal ADC0 (broche PC0). La tension de référence est AVcc = 5 V.

1. Configurer le CAN pour une résolution 10 bits.
2. Lire la valeur analogique et la convertir en tension (0–5 V).
3. Envoyer le résultat en mV via la liaison série (simulé ici par un affichage).

Exercice 5 : Filtrage logiciel par moyenne glissante sur 8 échantillons

La mesure analogique sur ADC0 est bruitée. Implémenter un filtre moyenne glissante sur 8 échantillons pour lisser la valeur. Afficher la tension filtrée.

Exercice 6 : Combinaison entrée numérique + CAN avec changement de référence

Énoncé :
On veut mesurer une tension de 0 à 2,5 V sur ADC0, mais avec une meilleure précision. On utilise la référence interne 2,56 V (ATmega328P).

1. Configurer la référence interne 2,56 V.
2. Lire ADC0 et calculer la tension.
3. Si la tension dépasse 2,0 V, allumer une LED sur PB5.

Université Djilali BOUNAAMA, Khemis Miliana
مليانة خميس بونعامة جيلالي جامعة
Faculté des Sciences et de la Technologie
والتكنولوجيا العلوم كلية

Solution Série 2

Exercice 1 :

```
#include <avr/io.h>
#include <util/delay.h>

int main(void) {
    // 1. Configuration : PD5 en sortie (bit 5 du port D)
    DDRD |= (1 << PD5);   // DDRD = Data Direction Register D

    while (1) {
        // 2. Allumer la LED
        PORTD |= (1 << PD5);
        _delay_ms(500);
        // Éteindre la LED
        PORTD &= ~(1 << PD5);
        _delay_ms(500);
    }
    return 0;
}
```

Exercice 2 :

```
#include <avr/io.h>


int main(void) {
    // Configuration
    DDRB &= ~(1 << PB0);   // PB0 en entrée
    PORTB |= (1 << PB0);   // Activer pull-up interne sur PB0


    DDRB |= (1 << PB5);    // PB5 en sortie pour LED


    while (1) {
        if (!(PINB & (1 << PB0))) {  // Bouton appuyé (0 logique)
            PORTB |= (1 << PB5);     // Allume LED
        } else {
            PORTB &= ~(1 << PB5);    // Éteint LED
        }
```

```
    }
    return 0;
}
```

Explication du pull-up :

Le pull-up interne maintient la broche à un niveau haut (1) quand le bouton est ouvert. Quand on appuie, la broche est reliée à la masse (0). Sans pull-up, la broche serait flottante et donnerait des lectures aléatoires.

Exercice 3 :

```
#include <avr/io.h>
#include <util/delay.h>

int main(void) {
    uint8_t compteur = 0;

    // Bouton sur PD2 en entrée avec pull-up
    DDRD &= ~(1 << PD2);
    PORTD |= (1 << PD2);

    // LEDs sur PD4..PD7 en sortie
    DDRD |= (0xF0);  // bits 4-7

    while (1) {
        if (!(PIND & (1 << PD2))) { // appui détecté
            _delay_ms(20);          // attendre la fin du rebond
            if (!(PIND & (1 << PD2))) { // vérifier toujours appuyé
                compteur++;
                if (compteur > 15) compteur = 0;
                // Affichage sur les 4 bits de poids fort
                PORTD = (PORTD & 0x0F) | (compteur << 4);
            }
```

```
            // Attendre relâchement
            while (!(PIND & (1 << PD2)));
            _delay_ms(20);
        }
    }
    return 0;
}
```

Exercice 4 :

```
#include <avr/io.h>
#include <stdio.h>

void ADC_init(void) {
    // Référence : AVcc, canal 0, activation
    ADMUX = (1 << REFS0);   // AVcc avec condensateur externe sur AREF
    ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1); // div par 64
    // ADEN = ADC Enable, ADPS = prescaler pour 16 MHz -> 250 kHz (ok)
}

uint16_t ADC_read(void) {
    ADCSRA |= (1 << ADSC);   // Démarrer conversion
    while (ADCSRA & (1 << ADSC)); // Attendre fin
    return ADC;              // ADC est un registre 16 bits
}

int main(void) {
```

```
    ADC_init();
    uint16_t valeur_brute;
    uint16_t tension_mV;

    while (1) {
        valeur_brute = ADC_read();
        // 10 bits -> 0 à 1023 pour 0 à 5000 mV
        tension_mV = (valeur_brute * 5000UL) / 1023;
        // Ici on pourrait envoyer tension_mV sur UART
        // Pour l'exercice, on suppose un affichage
    }
    return 0;
}
```

Exercice 5 :

```
#include <avr/io.h>

#define TAILLE_FILTRE 8

uint16_t moyenne_glissante(uint16_t nouvelle_valeur) {
    static uint16_t buffer[TAILLE_FILTRE];
    static uint8_t index = 0;
    static uint16_t somme = 0;
    static uint8_t plein = 0;

    somme -= buffer[index];
    buffer[index] = nouvelle_valeur;
    somme += nouvelle_valeur;
    index = (index + 1) % TAILLE_FILTRE;
    if (!plein && index == 0) plein = 1;

    if (plein) return somme / TAILLE_FILTRE;
    else return somme / (index + 1); // moyenne partielle
```

```
}

// ADC_init() et ADC_read() comme exercice 4

int main(void) {
   ADC_init();
   uint16_t brut, filtree_mV;

   while (1) {
      brut = ADC_read();
      filtree_mV = (moyenne_glissante(brut) * 5000UL) / 1023;
      // Afficher filtree_mV
   }
   return 0;
}
```

Exercice 6 :

```
#include <avr/io.h>


void ADC_init_ref_interne(void) {
   // Référence interne 2.56V (REFS1=1, REFS0=1) pour ATmega328P
   ADMUX = (1 << REFS1) | (1 << REFS0);
   // Canal 0 par défaut (MUX0..3 = 0)
   ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1);
}


uint16_t ADC_read(void) {
   ADCSRA |= (1 << ADSC);
   while (ADCSRA & (1 << ADSC));
   return ADC;
}


int main(void) {
   // LED PB5 sortie
   DDRB |= (1 << PB5);
   ADC_init_ref_interne();
```

```
    uint16_t brut;
    uint16_t tension_mV; // max 2560 mV

    while (1) {
        brut = ADC_read();
        // 10 bits -> 0 à 1023 pour 0 à 2560 mV
        tension_mV = (brut * 2560UL) / 1023;

        if (tension_mV > 2000) {   // > 2,0 V
            PORTB |= (1 << PB5);
        } else {
            PORTB &= ~(1 << PB5);
        }
    }
    return 0;
}
```