Electronique numérique : Introduction à la programmation µC : Partie pratique

Chacun des exercices devra être testé sur l'arduino mega et devra être intégré au rapport final. Le code sera obligatoirement commenté en anglais. Le rapport devra également comporter les rappels théoriques nécessaires à la compréhension (ceci vous sera d'utilité lors de l'examen oral).

L'utilisation de « Copilot » ou de « ChatGpt » est acceptée. Cependant, une explication fine et précise du code sera demandée lors de l'examen oral.

1. Exercices sur les GPIO :

Les énoncés

Câbler un bouton poussoir (muni d'une pull-up externe) sur une GPIO (pin 2 de l'arduino méga) et utiliser la led intégrée sur la carte (pin 13)

1.1. Réaliser un schéma (dans le rapport) de la connexion du BP sur l'entrée de l'arduino en reprenant le schéma équivalent d'une entrée.

1.2. Faire changer l'état de la led à chaque appui sur le BP en utilisant :

- Les instructions Arduino, par lecture du port (boucle de pooling)
- Les instructions Arduino, par interruption (utilisation attachInterrupt(…))
- Les registres d'IO, par lecture du port (**voir questions d'analyse**)
- Les registres d'IO, par interruption (**voir questions d'analyse**)

1.3. Faire clignoter une LED avec une période d'une seconde en utilisant :

- Les instructions Arduino, fonction delay()
- Les instructions Arduino, fonction millis()
- Les registres d'IO, fonction millis() (**voir questions d'analyse**)

Bonus : Refaire un des tests avec le BP (au choix) mais sans pull-up externe (utilisation d'une pull-up interne)

Questions d'analyse

1. Compréhension des codes

- Quelle est la différence entre EICRA et EICRB ?
- Pourquoi utilise-t-on INT4_vect et non INT0_vect ? Quelles sont les possibilités sur l'arduino Mega ?
- Utilisez-vous « volatile » dans vos déclarations de variables ? Si oui, que signifie-t-elle ?

- L’instruction sei() est-elle indispensable ici ? Que fait elle exactement ? Faire le lien avec le registre réellement utilisé (voir datasheet).
- Expliquer clairement la partie du code utilisant la fonction millis()
- Expliquer (si utilisés) en détails les lignes de code utilisant les opérateurs &, |, ~, <<

2. Lien avec le matériel

- Identifier les ports et bits associés à la pin 2 et à la pin 13 sur l’ATmega2560.
- Expliquer le rôle de la résistance de pull-up externe.

3. Lien avec le datasheet (toujours indiquer où trouver l’information dans le datasheet)

Rechercher dans le datasheet de l’ATmega2560 :

- La table des vecteurs d’interruption.
- La configuration des registres EICRB, EIMSK, DDRx, PORTx.
  Que configure-t-on exactement via le registre EICRB et EIMSK (faire le lien avec le code) ?
- Le rôle des registres ISC4x pour INT4 (faire le lien avec le code).

4. Modification du code existant

- Utiliser une autre interruption externe et vérifier que cela fonctionne.

2. Exercices sur les timers et PWM :

Les énoncés

2.1 Faire clignoter une LED (pin 13) avec une période d'une seconde :

- Registres d'I/O, Timer 1, débordement, par polling
- Registres d'I/O, Timer 1, CTC, par polling
- Registres d'I/O, Timer 1, débordement, par interruption
- Registres d'I/O, Timer 1, CTC, par interruption

2.2 Générer un signal d'horloge de 100 kHz (méthode au choix) sur une sortie au choix (autre que la pin 13). Vérifier la fréquence du signal généré à l'aide d'un oscilloscope.

2.3 Générer un signal PWM (utilisation de la fonction analogWrite) et faire varier le duty cycle en boucle (toutes les 100 ms via la fonction delay). Visualiser le signal à l'aide de l'oscilloscope.

2.4 Ajouter un filtre passe bas R-C afin d'obtenir une tension DC réglable. Vérifier le fonctionnement à l'oscilloscope.

Questions d'analyse

1. Compréhension des codes
    - Par pooling :
        - Expliquer ce qu'on peut configurer via les registres TCCR1A/B
        - Expliquer l'utilité d'un « prescaler ».
        - Expliquer le calcul permettant de trouver la période de débordement de votre timer et donc la valeur à placer dans le registre TCNT1 en mode normal et OCR1 en mode CTC.
        - Que se passe-t-il si on oublie de réinitialiser le flag TOV1
        - Expliquer les différences au niveau du code entre le mode normal et le mode CTC
        - Pourquoi utiliser le mode CTC plutôt que le mode normal ?
        - Que se passe-t-il si le polling est trop lent et rate le flag (OCF1A et TOV1) ?
    - Par interruption :
        - Quel est le rôle des vecteurs d'interruption TIMER1_OVF_vect et TIMER1_COMPA_vect ?
        - Pourquoi faut-il recharger TCNT1 dans l'interruption ?
        - Que se passe-t-il si on oublie d'activer TIMSK1 |= (1 << TOIE1) ?

2. Liens vers le datasheet (toujours indiquer où trouver l'information dans le datasheet)

    - Dans quelle section du datasheet trouve-t-on la description des registres TCCR1A, TCCR1B, TCNT1, OCR1A, TIFR1 ?
    - Quelle est la table des vecteurs d'interruption pour le Timer 1 ? Quelle routine serait appelée si on utilisait une interruption ?
    - Quels sont les bits à configurer pour activer le mode CTC dans TCCR1B ?

3. Exercices sur les ADCs :

Les énoncés

3.1 Réaliser la mesure d'une tension réglable par un potentiomètre. On prendra une mesure toutes les secondes. La valeur sera envoyée via UART et récupérée sur la console de l'IDE Arduino.

Plusieurs versions à réaliser :

- Avec les instructions arduino, utilisation de millis()
- Avec les registres : par polling
- Avec les registres : par interruption sur la fin de conversion

Questions d'analyse

1. Compréhension des codes

    - Pourquoi utilise-t-on ADSC pour démarrer une conversion ?
    - Que signifie le bit ADIE dans ADCSRA ?
    - Quelle est la différence entre ADC et les registres ADCL/ADCH ?

2. Lien avec le matériel
    - A quoi sert cette tension de référence configurée via le registre ADMUX ? Quelles sont les possibilités dans l'arduino MEGA?
    - Comment choisir le prescaler pour la fréquence d'horloge du convertisseur ? Quel impact va avoir cette fréquence au niveau de la conversion ?

3. Liens vers le datasheet (toujours indiquer où trouver l'information dans le datasheet)
    - Quel est le rôle du registre ADMUX ?
    - Quelle est la différence entre ADSC et ADIE ?
    - Comment est structurée la donnée convertie dans les registres ADCL et ADCH ?
    - Quel est le vecteur d'interruption associé à l'ADC ?

Exercice de synthèse :

- Réaliser un programme permettant de générer une horloge réglable entre 2 Hz et 100 kHz. Le consigne de réglage sera fournie à l'aide d'un potentiomètre (exemple : potentiomètre : 0V → 5V donnera une fréquence de sortie : 2 Hz → 100 kHz).

- BONUS : permettre de monter plus haut en fréquence en gardant notre limite basse de 2 Hz minimum

La seule contrainte est d'utiliser les registres et ceci pour les GPIOs, le timer et l'ADC. Des preuves de fonctionnement seront demandées lors de l'examen via une démo et via des captures d'écran de l'oscilloscope dans le rapport)