Polytechnique Montréal

Département de génie informatique et génie logiciel

INF1900: Projet initial de système embarqué

Évaluation écrite du volet technique - session automne 2018

Lundi 15 octobre 2018, 18h30

Enseignant : Jérôme Collin, ing., M. Sc. A.

Directives:

- l'évaluation est sur 20 points et est évaluée sur MoodleQuiz;
- la pondération pour la session est de 25%;
- la documentation, le robot et la calculatrice ne sont pas permis;
- il y a toujours un seul choix valide parmi les 5 proposé pour les questions de 1 à 15.

Ne pas répondre à la question ne fait perdre aucun point. Une bonne réponse donne 1.0 point. Une mauvaise réponse fait perdre 0.2 point.

- la question 16 demande de faire 5 associations;
- la durée est de 50 minutes.

Question 1 (1 point) [Pour évaluation de la qualité de l'ingénieur 2]

La minuterie timer1 du ATmega324PA génère une interruption en mode CTC après un certain temps dans la configuration actuelle dans un code existant. On veut cependant que l'interruption soit générée plus tôt par rapport à nos besoin dans un nouveau programme. Toutes ces stratégies sont valables pour générer l'interruption plus tôt dans le programme à concevoir en modifiant le code existant, sauf une, laquelle ?

1) Ajuster TCNT1 au départ à une valeur différente.
2) Utiliser un diviseur d'horloge (*prescaler*) différent de la minuterie.
3) Ajuster le OCR1A (ou le OCR1B) utilisé à une valeur différente.

4) Activer la minuterie plus tôt dans le programme.
5) Utiliser le mode fast-PWM plutôt que le mode CTC (Clear Timer on Compare Match).

Question 2 (1 point) [Pour évaluation de la qualité de l'ingénieur 12]

Toutes les affirmations suivantes sont vrai concertant les broches de la puce ATmega324PA de la carte mère utilisée en laboratoire, exceptée une, laquelle ?

1) Elle possède 40 broches.
2) 32 broches sont utilisées pour les 4 ports A, B, C et D.
3) La majorité des broches des ports A, B, C et D ont des rôles double, voir triple selon l'utilisation des périphériques internes et de leur configuration.
4) Une marque en demi-lune sur le puce aide à repérer la broche numéro 1.
5) Aucune broche n'est connectée avec le ATmega8.

Question 3 (1 point) [Pour évaluation de la qualité de l'ingénieur 2]

En considérant ces deux lignes de code provenant de deux programmes différents:

```
1)  if ( PIND == 0b00000100 ) {
2)  if ( PIND & 0x04 == 0x04 ) {
```

Qu'est-ce que l'on peut affirmer à propos de ces lignes qui est faux (une seule réponse) ?

1) À la rigueur, pour la deuxième, la comparaison est superflue et on pourrait écrire seulement: `if ( PIND & 0x04 ) {`
2) On devrait éviter d'utiliser la notation binaire de la constante dans la première ligne.
3) La première est sensible aux activités sur les bits autres que D2 du port D.
4) Sans que ce soit peut-être une bonne pratique dans la présente situation, on pourrait éviter d'utiliser la notation hexadécimale dans la seconde et utiliser simplement le chiffre 4 directement et le résultat serait le même.
5) On a nécessairement affaire ici à des lignes de code qui gèrent une interruption externe INT0 associée au bouton-poussoir sur la carte mère utilisée en laboratoire.

Question 4 (1 point)

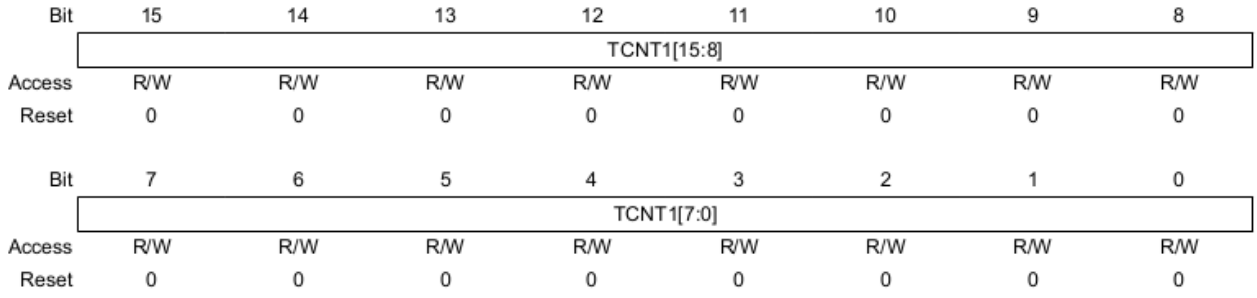
Pour écrire une machine à états finis logicielle de façon claire, on peut recommander tous les points suivants, sauf un, lequel ?

1) Utiliser une type *enum* pour définir la variable qui représente l'état dans la machine pour bien lister les états possibles.
2) Les sorties sont toujours identiques aux états. On obtient donc les sorties directement si les états sont clairs sans avoir à les décoder.
3) On peut coder une machine de Moore ou de Mealy sans problème de manière logicielle.
4) Pour coder les changements d'états, il est préférable, pour plus de clarté dans le code, d'avoir une structure de type *switch-case*.

5) On inclut en commentaire un tableau des états (équivalent au diagramme des états avec des «bulles» et des «flèches») pour documenter la machine à états à programmer.

Question 5 (1 point)

En considérant cette image, une seule des affirmations suivantes est vrai. Précisez laquelle.



Bit	15	14	13	12	11	10	9	8
	TCNT1[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TCNT1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

1) On doit ajuster ces deux registres à des valeurs précises et constantes par programmation avant de générer des signaux PWM de façon matérielle.
2) Sur la figure, on devrait ajouter des noms de bits pour toutes les positions qui sont définis dans la libraries AVRLibC pour aider à la configuration de ces registres.
3) Celui du haut peut rester inutilisé si on travail en mode 8 bits.
4) Un oscilloscope peut aider à suivre facilement le résultat produit par ces registres.
5) On a besoin de ces deux registres de façon indépendante l'une de l'autre pour arriver à générer deux signaux PWM distincts de la minuterie 1.

Question 6 (1 point) [Pour évaluation de la qualité de l'ingénieur 12]

Tous ces paramètres sont à ajuster pour obtenir une configuration logicielle fonctionnelle avec un USART d'un ATmega324PA, sauf un, lequel ?

1) On doit ajuster la vitesse de transmission (*baud rate*).
2) On doit préciser si la transmission sera en 7, 8 ou 9 bits.
3) On doit spécifier le nombre de bit d'arrêt (*stop bit*).
4) On doit dire si oui ou non on utilise le bit de parité.
5) Préciser le voltage utilisé pour les signaux de transmission.

Question 7 (1 point) [Pour évaluation de la qualité de l'ingénieur 12]

De la mémoire externe sur la carte mère utilisée en laboratoire, on peut considérer comme étant vrai toutes les affirmations suivantes, sauf une, lequel ?

1) C'est une mémoire ayant un accès par protocole série, et non parallèle.
2) On peut facilement retirer cette puce de son support.

3) Il faut mettre en place un cavalier (*jumper*) pour l'utiliser.
4) Elle conserve ses valeurs même si la tension qui l'alimente est coupée.
5) En utilisant cette mémoire externe, on ne peut plus utiliser la mémoire flash interne au ATmega324PA.

Question 8 (1 point)

Les affirmations suivantes concernent le PWM (*Pulse Width Modulation*). Elles sont toutes vraies, sauf une, laquelle ?

1) Le pourcentage de PWM s'évalue bien avec un oscilloscope.
2) Le PWM peut être généré indépendemment par une minuterie.
3) Le PWM peut être généré par un programme sur un processeur (CPU).
4) Le PWM peut contrôler autant l'intensité d'une diode électro-luminescente (DEL) que la vitesse d'un moteur.
5) La fréquence de l'onde PWM doit toujours être réglée à 10 KHz précisément pour faire tourner des moteurs électriques.

Question 9 (1 point) [Pour évaluation de la qualité de l'ingénieur 2]

Qu'est-ce que fait l'expression suivante exactement:

```
PORTB |= 3 << 1 ;
```

1) Les 2ème et 3ème bits à partir de la droite sont à un. On ne sait pas pour les autres.
2) Les 2ème et 3ème bits à partir de la droite sont à un. Les autres sont à zéro.
3) Le 4ème bit à partir de la droite est à un. On ne sait pas pour les autres.
4) Tous les bits sont inversés, sauf les 2ème et 3ème à partir de la droite qui sont à un.
5) Le 4ème bit est à zéro. On ne sait pas pour les autres.

Question 10 (1 point) [Pour évaluation de la qualité de l'ingénieur 2]

Que vaut la variable `result` à la fin des opérations suivantes:

```
uint16_t result = 0x00FF ^ 0x00FF;
result |= 0x42;
result = result << ( 0x03 || 0x01 );
```

1) result = 0x0084
2) result = 0x0210
3) result = 0x03FC
4) result = 0xFC00

5) result = 0xFF21

Question 11 (1 point)

Lorsqu'on écrit une routine d'interruption ISR, tous ces conseils sont bons, sauf un, lequel ?

1) Il faut aller chercher dans la documentation de AVRLibC pour trouver le bon identificateur d'interruption à placer en argument de la définition de la fonction.
2) Si on utilise des variables globales, il est recommander d'utiliser le qualificatif *volatile* pour définir nos variables.
3) Il faut avoir une instruction #include <avr/interrupt.h> au haut du fichier source.
4) Il serait bon de regrouper la configuration de tous les registres à initialiser pour le type d'interruption voulu dans un seul endroit dans le code pour que ce soit très lisible, avec de bons commentaires.
5) Écrire la ligne de code qui appelle la fonction ISR avec un bon commentaire au-dessus.

Question 12 (1 point)

On peut affirmer que le pont-en-H utilisé sur le robot en laboratoire a les propriétés ou comportements suivants sauf un qui est faux, lequel ?

1) Est normalement alimenté par une pile rectangulaire de 9 volts.
2) Permet de contrôler en vitesse et en direction de rotation par 2 fils pour chaque roue.
3) A un fusible qui peut «brûler» s'il y a un trop fort courant de façon à protéger le reste du circuit.
4) Un de ses rôles est d'offrir aux moteurs une tension et un courant plus élevé que ce que la carte mère peut fournir par ses ports.
5) Inverser les deux fils sortant d'un des borniers verts inversera le sens de rotation d'une roue.

Question 13 (1 point)

Il y a certains problèmes lorsqu'il faut interfacer un interrupteur mécanique à un microcontrôleur. Toutes ces affirmations sont correctes sauf une, laquelle ?

1) On peut corriger les rebonds par programmation mais on pourrait le faire matériellement si on ne veut pas le faire dans le code en utilisant plutôt une bascule ou un condensateur.
2) Les rebonds sont difficilement détectables si on travail seulement avec une DEL (diode électro-luminescente) en sortie.
3) Les rebonds sont le fruit d'un contact métal sur métal.
4) On pourrait simuler et tester un mécanisme d'anti-rebond en générant un signal en sortie où serait produit quelques petits pulses avant la stabilisation du signal. Très facile à faire de façon logicielle.
5) Le multimètre n'est pas conçu pour facilement détecter les rebonds.

Question 14 (1 point) [Pour évaluation de la qualité de l'ingénieur 2]

En considérant le code suivant :

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

int main()
{
   DDRA = 0xff;
   DDRB = 0xff;
   DDRC = 0xff;
   DDRD = 0xff;
   unsigned long compteur=0;

   for(;;)
   {
      compteur++;
      PORTD = compteur;
      PORTC = compteur >> 8;
      PORTB = compteur >> 16;
      PORTA = compteur >> 24;
   }
   return 0;
}
```

On peut affirmer que tous ces énoncés sont vrais, sauf un, lequel ?

1) Enlever la première ligne avec l'instruction include générerait une erreur de compilation parce que le compilateur ne saurait pas ce que DDRA veut dire.
2) On aimerait pouvoir utiliser un type *uint8_t* pour définir la variable «compteur» mais la situation exige l'emploi d'un entier plus long.
3) Lorsque le compteur atteint sa limite, le programme se termine.
4) On peut facilement faire un circuit avec des composantes matérielles de base de la logique numérique vues en INF1500 pour arriver à produire le même résultat.
5) Ce code n'utilise ni la scrutation, ni les interruptions.

Question 15 (1 point) [Pour évaluation de la qualité de l'ingénieur 12]

Une des affirmations suivantes n'est pas vrai concernant la commande «git pull», laquelle ?

1) suppose qu'il y a déjà eu un «git clone» de fait précédemment.
2) peut générer un conflit dans l'entrepôt local.
3) suppose une interaction de la copie locale avec un entrepôt, fort probablement distant.
4) devra être suivi immédiatement par une commande «git push».
5) devra être suivi par une commande «make» si l'entrepôt contient du code source en C/C++ pour régénérer les binaires présents localement.

Question 16 (5 points)

Pour chacun des cinq énoncés, de A à E tout au bas (1 point chacun), associer le bon numéro en choisissant parmi les choix proposés. Bien inscrire le numéro au bas.

Choix possibles:

1 – AVRDude

2 – Compteur de programme (Program Counteur – PC)

3 – make

4 – SPI

5 – serieViaUSB

6 – One Wire

7 – ALU

8 – EEPROM

9 – Test de continuité

10 – INT0_vect

11 – make fusesM8

12 – Debug

13 – ATmega8

14 – Reset

15 – I^2C / TWI

16 – RAM

17 – OC1A

18 – TIMER1_COMPA_vect

19 – Registre d'instructions (Instruction Register – IR)

20 – IntEN

A) À utiliser sur le PC avec le RS-232 : _____

B) Protocole utilisé pour communiquer avec la mémoire externe : _____

C) Contient le micrologiciel (*firmware*) : _____

D) Permet à ses entrées des opérations sur les 32 registres de base d'un AVR : _____

E) Assimilable à une interruption de la priorité la plus élevée, sans possibilité d'être désactivée (*disable*) et dont le comportement est toujours le même:

_____