Polytechnique Montréal

Département de génie informatique et génie logiciel

Cours INF1900:
Projet initial de système embarqué

Travail pratique 7

Makefile et production de librairie statique

Par l'équipe

No 5180

Noms:

SIMON Sanmar
NGUYEN Vy-Phuong
LEGRAND Nicholas
PRINVILLE Kervin

Date:
09 mars 20

Partie 1 : Description de la librairie

Décrire la librairie construite et formée (définitions, fonctions ou classes, utilité, etc...) pour que cette partie du travail soient bien documentées pour la suite du projet pour le bénéfice de tous les membres de l'équipe.

Pour faciliter la réutilisation d'anciens programmes, nous avons réunis les fonctions les plus intéressantes dans une librairie. Pour former cette librairie nous avons parcouru les anciens TD, du premier au sixième. Ainsi cette librairie est formée des fonctions suivantes :

- **amber.cpp ET amber.h.** Cette classe est jugée utile car nous estimons que c'est la solution la plus adéquate pour afficher la DEL en couleur ambré. Elle comporte notamment des fonctions **red()** et **green()** qui affichent la DEL en rouge et vert. Ce fichier est également utile car c'est une utilisation de la DEL qui reviens souvent.
- **starTimer.cpp ET startTimer.h.** Cette classe ajuste les différents registres du TIMER1 de sorte à lancer une minuterie en mode **CTC** (Clear to compare match). Lorsque le compteur de la minuterie atteint la valeur qui est transmise en paramètre, alors une interruption est déclenchée.
- **adjustmentPWM.cpp ET adjustmentPWM.h.** Cette classe ajuste les registres du TIMER1 pour lancer la minuterie en mode Phase-correct PWM. Il génère un signal PWM de période a et b égales (temps passé au niveau bas, et période totale de l'onde) transmis en paramètre. Ce fichier génère donc un signal PWM de **façon matérielle**
- **initInterrupt.cpp ET initInterrupt.h.** Cette classe est de toute importance. Elle est en charge d'ajuster les registres utiles au problème, mais également de définir quelles broches sont utilisés en sortie ou en entrée.
- **can.cpp ET can.h.** La classe can permet l'utilisation du convertisseur analogique-numérique du ATMEGA324 PA. Cette classe est absolument nécessaire si l'on veut interagir avec le monde réel via des capteurs, et donc traiter des signaux analogiques.
- **memoire_24.cpp ET memoire_24.h.** Cette classe permet la lecture et l'écriture de données dans la mémoire EEPROM de la carte mère. Cette classe peut donc être utile lorsqu'il s'agit de stocker des données.
- **uart.cpp.** Ce fichier source assure la transmission entre le ATMEGA8 et l'ordinateur.

Une fois la librairie crée, il faut maintenant apporter des légères modifications au Makefile fournis au début du trimestre. Le nouveau Makefile permettra de lier notre librairie au nouveau code développé.

Partie 2 : Décrire les modifications apportées au Makefile de départ

Décrire les quelques modifications apportées au Makefile de la librairie pour démontrer votre compréhension de la formation des fichiers. Faire de même pour les modifications apportées au Makefile du code (bidon) de test qui utilise cette librairie.

Modifications sur le Makefile de la librairie :

Voici les modifications apportées au Makefile de la librairie :

1. Modification de la valeur TRG, donc remplacer TRG = $(PROJECTNAME).out par TRG = $(PROJECTNAME).a. Ceci est important puisque qu'il faut créer une librairie au lieu d'un fichier exécutable.
2. - Création de la variable AR = ar pour pouvoir compiler les fichiers et obtenir une librairie.
3. Modification de la variable PRJSRC pour qu'elle soit désormais PRJSRC = $(wildcard *.cpp). Nous n'avons donc plus besoin de mettre le nom de chaque fichier puisque la commande compilera tous les fichiers .cpp du dossier courant.

Modifications sur le Makefile du code :

Voici les modifications apportées au Makefile du code (bidon) :

1. LIBS = -L ../Lib/ -l Librairie.a, cette ligne nous permet de lier notre librairie au code à compiler. -L précise l'adresse de la librairie, et -l son nom.
2. Ar=ar cette ligne nous permet de déclarer le fichier comme une librairie statique.
3. -o par -crs Cette ligne nous permet de créer le .a au lieu du .out.
4. Nous avons enlevé le install puisqu'il est inutile pour une librairie statique.

Le rapport total ne doit par dépasser 7 pages incluant la page couverture.

Barème: vous serez jugé sur:

- *La qualités et le choix de vos portions de code choisies (5 points sur 20)*
- *La qualités de vos modifications aux Makefiles (5 points sur 20)*
- *Le rapport (7 points sur 20)*
  - *Explications cohérentes par rapport au code retenu pour former la librairie (2 points)*
  - *Explications cohérentes par rapport aux Makefiles modifiés (2 points)*
  - *Explications claires avec un bon niveau de détails (2 points)*
  - *Bon français (1 point)*
- *Bonne soumission de l'ensemble du code (compilation sans erreurs, …) et du rapport selon le format demandé (3 points sur 20)*