CEG 3536 – Architecture d'ordinateur II
Labo 2 – L'interface au matériel –Le clavier

Objectifs:

L'introduction de l'interface du matériel Motorola 9S12DG256 afin de réaliser une application d'un clavier numérique.

Préparation:

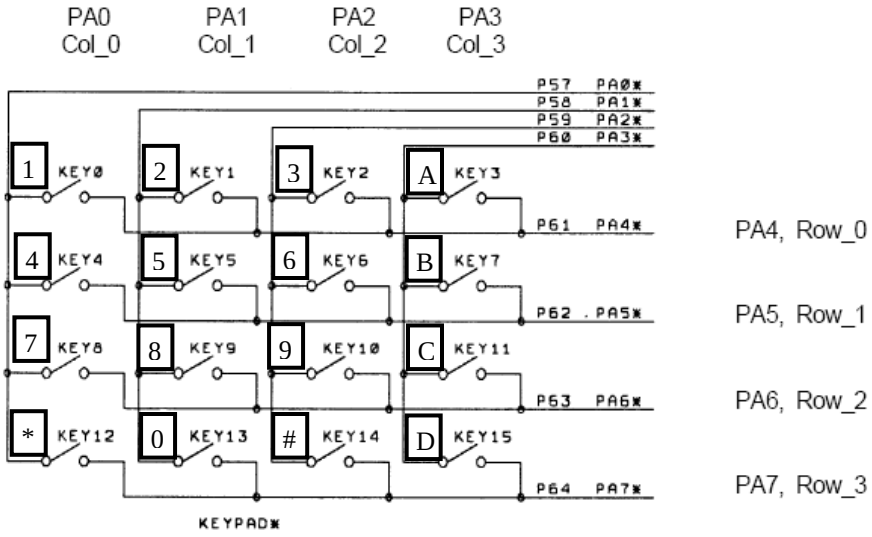
- Des points sont affectés à la préparation au labo.
- S.V.P. lisez ce document au complet avant de venir à la session de laboratoire.
- Préparez une ébauche du rapport de laboratoire.
- Lire au sujet des ports parallèles dans le Chapitre 11 du texte et les sections 3.1.1, 3.1.3, et 3.1.10 dans le document « S12MEBIV3.pdf » qui fait partie des spécifications du MC9S12DG256. De l'information sur l'interface avec les claviers est fournie dans la section 18.2 (An Array of switches) de votre texte et la section 4.4 du manuel d'utilisateur de la carte Dragon-12.
- Participez au Tutorat 3 qui discute les ports parallèles.
- Préparez la conception de votre système (matériel et logiciel) :
    - Préparez un schéma du circuit que vous utiliserez dans ce lab (i.e. comment le clavier est branché au microcontrôleur y compris la présence de résistance de report à la source).
    - Documentez la conception des modules de logiciel dans l'ébauche du rapport de laboratoire (utilisant C comme pseudo-code pour définir des algorithmes).
- Codez et assemblez vos programmes avec le MiniIDE.
- Montrez votre travail de préparation (rapport de laboratoire et le code de vos modules) à l'AE du lab dès votre arrivée.

Équipement utilisé:

- Windows PC
- Carte d'entraînement Dragon-12

Description:

Dans ce laboratoire vous aurez à développer du logiciel capable de lire du clavier de la carte Dragon-12 branché au microcontrôleur via le port parallèle et intégrer votre module dans le logiciel du système d'alarme. S.V.P. consultez la documentation du Dragon-12 pour plus de détails au sujet des diverses composantes utilisées dans le circuit. En particulier révisez dans le manuel du Dragon-12 (Section 4.4) l'opération du clavier connecté au port parallèle A tel que montré dans la Figure 1 ci-dessous. Notez que la documentation du Dragon-12 offre une direction pour faire la lecture du clavier – soyez conscient que l'approche proposée par la documentation Dragon-12 NE sera PAS utilisée dans ce lab.



Keypad connections:
PA0 connects COL0 of the keypad
PA1 connects COL1 of the keypad
PA2 connects COL2 of the keypad
PA3 connects COL3 of the keypad
PA4 connects ROW0 of the keypad
PA5 connects ROW1 of the keypad
PA6 connects ROW2 of the keypad
PA7 connects ROW3 of the keypad

Figure 1 – Connexion du clavier au port a

Notes au sujet des commutateurs de contact (clefs du clavier numérique)

Les commutateurs de contact sont typiques dans les claviers. Une fermeture de commutateur peut être détectée par le MCU (microcontroller unit) avec un circuit simple montré dans la figure 2. La résistance de rappel à la source offre une valeur logique de 1 (5 volts) quand le commutateur est ouvert, et une valeur logique de 0 (GND) quand il est fermé. Puisque les ports E/S du MCU contiennent des résistances de rappel à la source lorsque les broches sont configurées comme des entrées, de telles résistances externes ne sont pas nécessaires. Le circuit du commutateur au MCU peut donc être simplifié tel que démontré dans la figure 3.

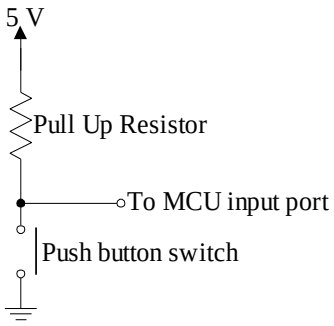


Figure 2: Un commutateur de clavier

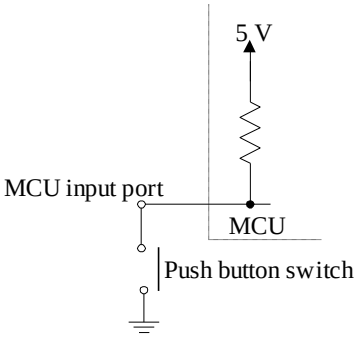


Figure 3: Un commutateur avec résistance de rappel du MCU

Malheureusement, les commutateurs ne sont pas idéaux parce qu'ils ne créent pas un beau 1 (5V) ou 0 (0V) lorsque pressés ou relâchés. Le microcontrôleur qui échantillonne le clavier très rapidement voit les actions du clavier comme étant très lent. Dans les faits, durant une action de commutation, le commutateur oscille (« bondit ») entre les états de connexion et de déconnexion après le contact initial ou la séparation initiale avant de s'établir. Ce processus aléatoire prend environ 2 à 20 ms. La figure 4 démontre l'oscillation de commutateur lorsqu'une clef est pressé et relâcher. Le MCU doit s'assurer que la connexion physique est fermée et bien établie. Ceci est fait en attendant 10 ms après avoir vu le premier contact ou la première séparation, et de confirmer que la même clef a été pressée et d'éviter de percevoir les bondissements comme plusieurs tapes de clefs. Une fois que le bord descendant de la commutation est détecté, le MCU doit attendre le bord ascendant. Autrement, le MCU peut encore interpréter une tape de clef comme plusieurs tapes.

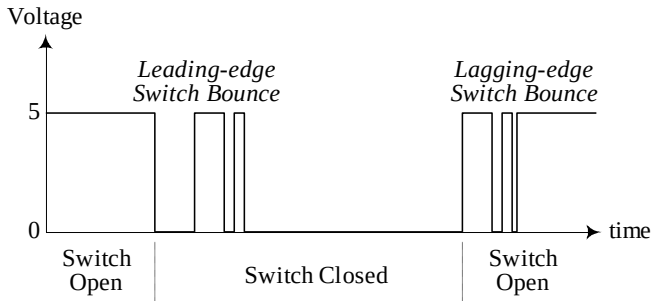


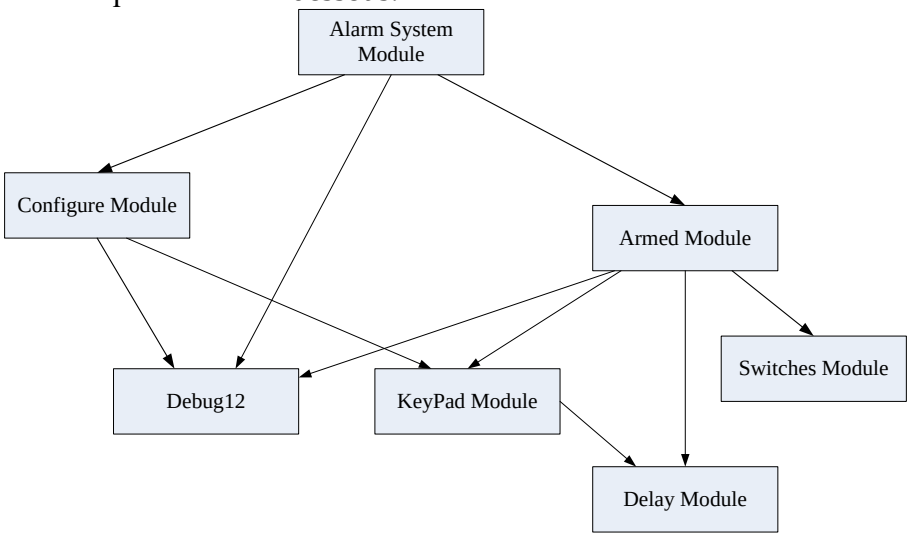
Figure 4: Oscillation de commutation

Une approche pour lire le clavier

Lorsqu'une clef est pressée sur le clavier, un code distinctif apparaîtra dans le registre de donné du Port A quand un zéro est écrit dans le bit associé à la broche connecté à la rangé de la clef pressée (ceci prend pour acquis que les broches connectées aux rangées du clavier sont des broches de sorties). Utilisez un tableau pour traduire la valeur lue du registre de donnée du Port A à un caractère indiqué sur la carte Dragon-12 (La figure 1 ci-dessus montre comment les divers caractères sont associés aux clefs du clavier). Prenez le temps de penser à une procédure de lecture du clavier, c'est-à-dire quelles étapes doit prendre l'UCT pour déterminer quand et quelle clef est pressée. Le tutorat 3 devra aider avec des idées. Ajoutez votre tableau et approche de lecture dans votre rapport.

Le logiciel du système d'alarme

Votre tâche principale est de développer le module *KeyPad* et de l'intégrer dans le logiciel du système d'alarme tel que montré ci-dessous.



Tous modules sont fournis à l'exception de deux. Les modules que vous devez développés sont stockés dans de fichiers séparés et doivent fournir les fonctions suivantes :

1. KeyPad.asm: Contient le code pour le module Keypad. Ce module contiendra un sous-programme pour initialiser le matériel (Port A) connecté au clavier et un sous-programme pour lire une clef du clavier (retourne le code ASCII de la clef tapée). Assurez vous que le $2^{ième}$ sous-programme retour la valeur de la clef une fois la clef relâchée (ceci veut dire que vous devez composer avec le bondissement des les deux extrémités de la tape de clef).
   a. void initKeyPad(): initialise le matériel (Port A) connecté au clavier.
   b. char readKey(): lire une clef du clavier (retourne le code ASCII de la clef tapée). La valeur de la clef est retournée une fois la clef relâchée (ceci veut dire que vous devez composer avec le bondissement des les deux extrémités de la tape de clef).
   c. char pollReadKey(): vérifie si un clef est pressée et retourne sa valeur; autrement retourne le caractère nul. INDICE : utilisé un délai d'une milliseconde pour l'antirebond du clavier et appelez *readKey* pour complété l'anti-rebond et traduire de code de clavier.
   d. En assembleur utilisez le registre B pour retourner la valeur du caractère par *readKey* et *pollReadKey*.
2. Delay.asm: Modifiez le module Delay du labo 1 pour inclure une fonction/sous-programme additionnel(le), *void delayms(int num)*, qui crée un délai de *num* millisecondes où *num* est un paramètre du sous-programme. Cette fonction/sous-programme sera utilisée par le module Keypad qui a besoin de délais pour composer avec les rebondissements des contacts du clavier. Utilisez le registre D pour envoyer la valeur de *num* à *delayms*.

Le module Switches est un nouveau module qui n'était pas présent dans le labo 1. Sa fonction permet d'observer l'état des commutateurs connectés au PORT H du microcontrôleur. Ces commutateurs représentent les commutateurs des portes et fenêtres surveillés par le système d'alarme. Pour garder le system plus simple, l'anti-rebond n'est pas implémenté dans ce module.

Conception du module Switch

Le module contient les fonctions/sous-programmes suivants:

```
/*---------------------------------------
 * Function: initSwitches
 * Parameters: none
 * Returns: nothing
 * Description: Initialises the port for monitoring the switches
 *              and controlling LEDs.
 *---------------------------------------*/
void initSwitches()
{
   DDRH = 0; // set to input (switches)
   PERH = 0xff; // Enable pull-up/pull-down
   PPSH = 0xff; // pull-down device connected to H
                // switches ground the pins when closed.
}

/*------------------------
 * Function: getSwStatus
 * Parameters:  none
 * Returns: An 8 bit code that indicates which
 *          switches are opened (bit set to 1).
 * Description: Checks status of switches and
 *              returns bytes that shows their
 *              status.
 *---------------------------*/
byte getSwStatus()
{
    return(PTH);
}
```

Fichiers *Include*

`Reg9S12.inc`: Ce fichier contient les EQUATES pour définir les symboles des adresses des registres de périphériques et peut être utilisé dans les fichiers des modules divers. Cette approche peut causer un problème à l'assemblement de votre code. Par exemple, les symboles des adresses de registres des périphériques se retrouvent dans le fichier `Reg9S12.inc`. Si ce fichier se fait inclure dans les deux fichiers *Keypad.asm* et *Delay.asm*, à l'assemblage du fichier *Alarm.asm,* plusieurs erreurs se produiront puisqu'il y aura de la redéfinition de symboles. Pour résoudre ce problème, des directives conditionnelles permettent l'inclusion du contenu d'un fichier seulement s'il ne l'a pas déjà été inclus. Voici l'approche :

1. Dans le fichier `Reg9S12.inc`, définissez un symbole de contrôle tel que « `REG9S12 EQU 1` ».
2. Ajoutez maintenant les directives suivantes en début et fin du fichier `Reg9S12.inc`:

```
ifndef REG9S12  ; include only if not yet included
    <contenu du fichier>
endif
```

   Si le symbole `REG9S12` n'est pas défini, alors le contenu du fichier `Reg9S12.inc` sera inclus, autrement il ne le sera pas, puisque son contenu est déjà présent.

`sections.inc` : Le fichier « sections.inc » contient les définitions pour les diverses sections du projet. L'inclusion de ce fichier dans tous les modules permettra la possibilité de faire l'assemblage des modules individuellement sans générer des erreurs par le manque des définitions des sections. Notez que le fichier contient des directives conditionnelles afin de permettre l'assemblage de tous les modules.

Indices

Voici quelques indices pour vous aidez dans votre développement:

1. Pour expérimenter avec le matériel, vous pouvez modifier les registres des ports parallèles directement. Utilisez la commande MM (memory modify) du D-Bug12 pour initialiser les registres appropriés. Vous pouvez aussi utiliser la même commande pour vérifier le contrôle du clavier (utilisez la commande MD (memory display) pour voir l'effet d'une clef tapée sur le registre de donnée).
2. Partagez la préparation du laboratoire en partageant les modules entre les deux partenaires de votre équipe. Par exemple, une personne peut se charger de la conception et du codage pour le module Délai ainsi que le sous-programme qui traduit le code lu du Port A à un code ASCII, tandis que l'autre personne est responsable de la balance du module KeyPad.

À faire

Préparation:

a. Faite la conception de votre matériel et logiciel (en suivant les lignes directrices données) qui devra faire partie de votre rapport :
    a. Complétez un schéma de circuit pour montrer comment le clavier est branché au microcontrôleur et comment les résistances de report à la source sont utilisées.
    b. Documentez la conception de chaque module de votre logiciel avec le C comme pseudo-code.
b. Préparez votre code assembleur :
    a. Écrivez votre code pour chaque module (selon les directives ci-dessus). Votre code devra être assemblé avant d'arriver au laboratoire.
c. Vous devez montrez l'ébauche de votre rapport de labo qui comprend votre conception initial et code assemblé à l'AE du lab dès votre arrivée (des points sont affectés à ce travail). Il est important de compléter votre préparation car faire le débogage de votre logiciel peut prendre du temps et peut comprendre la révision de votre conception initiale. Il n'est pas possible durant une session de 3 heures de concevoir, créer le logiciel et le faire fonctionner.

Au laboratoire:

a. Chargez votre programme dans le RAM du MC9S12DG256. Roulez votre programme et testez tous les numéros du clavier ainsi que les lettres A et C pour faire fonctionner le système d'alarme.
b. Montrez à votre AE l'opération de votre logiciel. Il notera votre succès.

Dans votre rapport:

a. Inclure votre le schéma complété du circuit (montrez comment le clavier est connecté au microcontrôleur y compris la connexion des résistances de report à la source).
b. Donnez la conception finale de vos modules. Inclure le « pseudo-code » C et une description des algorithmes utilisés. Fournissez votre code source assembleur dans des fichiers séparés (utilisez un fichier zip).