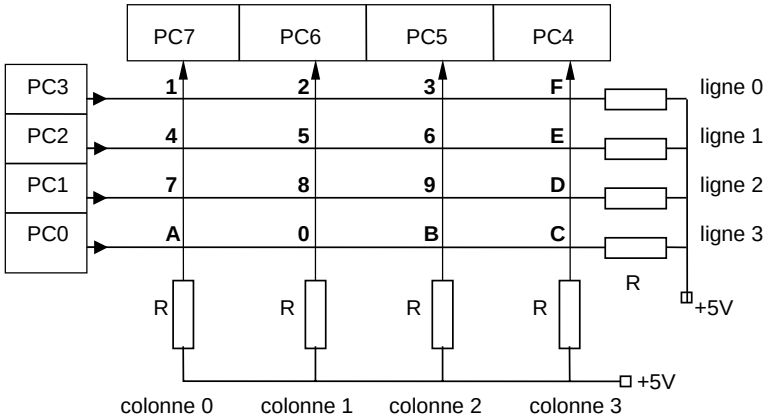
Informatique Industrielle Tronc commun Série Electronique

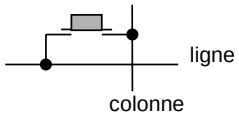
Partiel de Travaux pratiques

Tous documents autorisés *durée 1h30*

On propose de gérer un clavier hexadécimal **par interruption**.
Ce clavier qui comporte 16 touches est relié à un port parallèle bi-directionnel de 8 bits, le **PORTC** .
La scrutation permanente de ce clavier se fera toutes les 20 ms à l'aide d'un timer 16 bits qui devra fonctionner par interruption.
Le clavier 4 par 4, hexadécimal, comporte 16 touches disposées comme suit :



Détail d'une touche :



Fonctionnement du clavier :

Quand la touche n'est pas appuyée, ligne et colonne ne sont pas en contact.
Une lecture de la colonne correspondante donnera donc un niveau 1 car il y a une résistance de rappel au 5V.
Quand la touche est appuyée, ligne et colonne sont alors en contact, une lecture de la colonne correspondante donnera donc un niveau 0, s'il a été au préalable appliqué sur la ligne.

Le port parallèle PORTC possède un registre de direction **DDRC**, qui permet de définir comme entrée ou comme sortie chaque fil. Par exemple la mise à 0 du bit 5 de DDRC définit le fil PC5 comme une entrée, et sa mise à 1 comme une sortie.
Remarque : En salle de TP l'accès à ces 2 registres se fera **directement** à l'aide des équivalences symboliques PORTC et DDRC définies dans le fichier hc11.h .

Q1. Quelle valeur, que l'on appellera SENSLICO, doit-on mettre dans le registre DDRC pour respecter le câblage du clavier ? Ecrire la fonction **InitClavierPOC ( )** qui charge la valeur SENSLICO dans le registre DDRC.

Test d'un appui sur le clavier :

On propose d'écrire une fonction appelée **bAppui( )** , de type booléen qui retourne FAUX (ou 0) si aucune touche n'est appuyée ou VRAI (ou 1) si un appui s'est effectué.
On va donc tester toutes les touches d'un coup en appliquant des 0 en même temps sur les lignes PC3 à PC0 puis en lisant les 4 colonnes PC7 à PC4, également d'un coup.

Q2. Quelle valeur, notée TESTLICO, doit-on écrire dans le registre PORTC pour qu'il en soit ainsi ? Ecrire la fonction de type *Booléen* **bAppui** *(SansType*).

Interruptions du timer 16 bits :

Le clavier doit être scruté par interruptions régulières toutes le 20 ms. On utilise le timer 16 bits du 68HC11 pour générer les interruptions. Ainsi dans la fonction d'interruption on appellera la fonction bAppui ( ) qui teste le clavier.

Pour le Timer 16 bits du 68HC11 on utilise les registres suivants :

- le registre de comparaison 16 bits **TOC1** .
- le registre de masques des iTs **TMSK1** : si b7 = OC1I = 1 alors l'interruption associée est autorisée.
- le registre de drapeaux des iTs **TFLG1** : si b7 = OC1F = 1 alors l'**événement** associé est arrivé.

On place une valeur **fixe** dans le registre TOC1 qui permettra de déclencher une interruption lorsque le compteur TIMER1 = TOC1 alors b7 de TFLG1 passe à 1.

Q3. Ecrire la fonction **InitiTTimer ( )** qui configure l'interruption.

Q4. Ecrire la fonction d'interruption **ItTimerClav ( )** qui accède au clavier toutes les 20 ms. Elle devra en plus positionner la *variable globale* bAppClav à VRAI (1) ou FAUX (0).

Q5. La fonction principale doit simplement tester comment a été positionné la variable globale bAppClav par la fonction d'interruption.
On fera **printf("Appui sur clavier \n\r ")** si la variable globale est à VRAI, sinon on ne fait rien.
La chaîne de caractères se voit dans la fenêtre *Output* du débogueur Noice.
Copier chez vous les 3 fichiers fourni dans ***L:\infind2\clavier*** .
Modifier le source **clavit.c** .

Q6. Tester l'application complète avec le débogueur Noice.
⟹ Faire une démonstration à un enseignant lorsque ça fonctionne.
Préciser l'emplacement et le nom de votre source dans votre répertoire K :.

J.N. Martin le 11/01/01