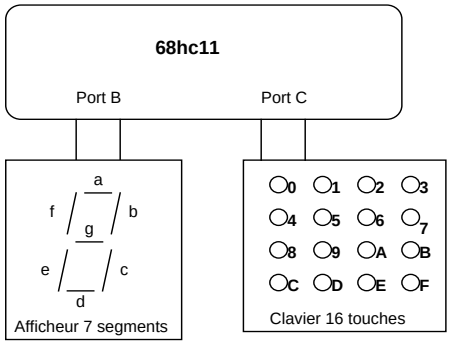
TP n°2 : LECTURE DE CLAVIER MATRICIEL EN SCRUTATION

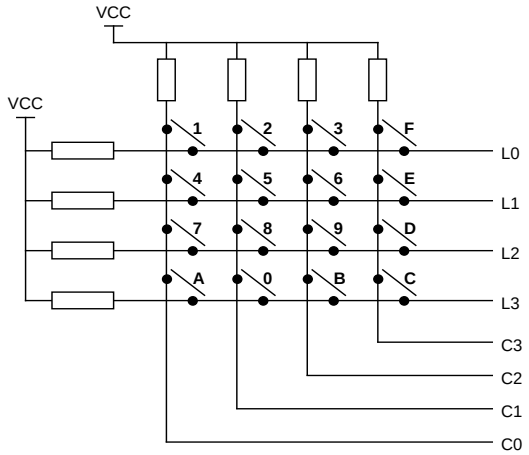
Introduction

Dans ce TP, vous verrez comment on réalise l'acquisition d'une touche d'un clavier matriciel par scrutation, en réalisant une lecture permanente de l'état du clavier, en attendant l'appui sur une touche. Vous mettrez ensuite ceci en application en affichant sur un afficheur 7 segments la touche enfoncée, via un transcodage.

Matériel utilisé



Le clavier utilisé est matriciel: à chacune des 16 touches correspond un **bouton-poussoir** qui réalise un contact entre une ligne et une colonne. Les 4 lignes et les 4 colonnes sont **polarisées** à +Vcc (5V) via des résistances de *Pull-up*, et sont connectées à un port parallèle du processeur (Port C). (Ce port doit obligatoirement être bidirectionnel).



Avantage : on a besoin que d'un port 8 bits au lieu de 16 pour 16 touches (Pour un clavier 64 touches, on aurait eu besoin de 16 entrées au lieu de 64, soit une économie de 48 broches.)

Inconvénient : on ne peut pas avoir **directement** une valeur binaire correspondant à la touche enfoncée. En effet, si toutes les 8 lignes sont en entrée, on lira un état logique '1' (à cause des résistances de Pull Up) QUEL QUE SOIT L'ETAT DU CLAVIER, qu'il y ait une, aucune, ou plusieurs touches enfoncées !

On est donc obligé de réaliser une lecture de la touche enfoncée en deux temps: d'abord on identifie la colonne enfoncée, puis la ligne. On récupère alors un **code-clavier**, caractéristique de la touche enfoncée.

Cette opération est réalisée par l'algorithme présenté en dernière page du TP (méthode du retournement du sens des lignes).

PREPARATION

Lire le document-ressource en fin de TP et répondre aux questions suivantes:

-1) Compléter le tableau suivant donnant le code-clavier en fonction de la touche.

	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	
Touche	**C0**	**C1**	**C2**	**C3**	**L0**	**L1**	**L2**	**L3**	**Hexa**
1	0	1	1	1	0	1	1	1	77
2									
3									
F									
4									
5									
6									
E									
7									
8									
9									
D									
A									
0									
B									
C									

-2) Quel doit être la valeur à programmer dans le registre DDRC pour avoir les lignes en sortie et les colonnes en entrée : _____

PARTIE 1 : GESTION DU CLAVIER EN SCRUTATION

-1.1) Créer un nouveau fichier assembleur en utilisant le fichier modèle :
(Menu démarrer->Programmes->jbug11->Nouveau fichier assembleur)
et le renommer en **TP2.ASM**

-1.2) Ecrire dans le bloc d'initialisations les lignes correspondantes (voir document-ressources)

-1.3) Ecrire dans la zone "programme principal" les lignes implémentant l'algorithme de lecture clavier, et tester en pas à pas, en appuyant de façon permanente sur une touche du clavier. Vérifier sous Jbug11 que la valeur lue à l'étape 6) de l'algorithme correspond bien au code de la touche pressée.

Remarque: afin de limiter les erreurs (et de gagner du temps...), il est recommandé de bien séparer chaque étape dans votre listing, en mettant à chaque fois une ligne de commentaire, par exemple :

```
* etape n°1 : colonnes en entrée, lignes en sortie
              ldaa   #$__
              staa   DDRC
* etape n°2 : ecrire 0 sur le PORTC
              ...
```

-1.4) Détection de l'appui d'une touche

Le clavier peut être au repos. Il ne faut aller au bout de l'algorithme (et donc lire une touche) que si effectivement une touche est enfoncée!

On va effectuer cette détection **après** l'étape n°3 : si on lit sur les colonnes la valeur '1111', c'est qu'aucune touche n'a été enfoncée, il faudra alors **reboucler** à l'étape 3 de l'algorithme.

Rajouter dans votre programme la comparaison et le branchement conditionnel nécessaire pour reboucler si aucune touche n'est enfoncée. Tester en pas à pas: faîtes l'essai clavier au repos, puis en appuyant sur une touche.

-1.5) Implémenter tout le programme principal comme un sous-programme, appelé LECTURE, et le re-tester en pas à pas, avec comme programme principal les lignes suivantes :

```
DEBUT     bsr     LECTURE
          staa    PORTB
          bra     DEBUT
```

Pensez à rajouter, en fin du sous-programme LECTURE, la **ré-initialisation du PortC** (sens des lignes et valeurs en sortie).

Vérifier en pas à pas que l'on obtient bien sur les 8 dels le code clavier correspondant à la touche enfoncée (voir tableau ci-dessus).

-1.6) Affichage de '1' et '2' sur l'afficheur 7 segments.

En utilisant 2 instructions de comparaison suivies chacune d'un branchement conditionnel, modifier le programme principal (et **uniquement celui-ci**) pour qu'un appui :

- sur '1' affiche sur l'afficheur 7 segments un '1' (code 7 seg.: 0000 0110)
- sur '2' affiche un '2' (code 7 seg.: 0101 1011)
- sur une autre touche éteint l'afficheur

Vérifier le fonctionnement en pas à pas, puis lancer une exécution('*Go'*). Le fonctionnement est-il satisfaisant ?

-1.7) Gestion des rebonds

Les problèmes constatés viennent des rebonds multiples des contacts du clavier. La solution consiste à attendre le **relâchement** de la touche avant de sortir de la routine d'interruption.

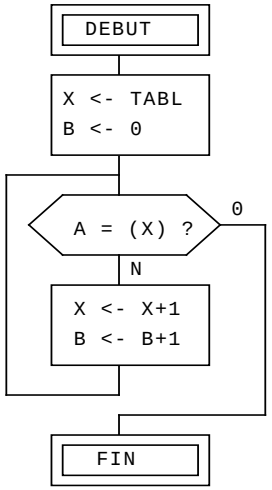
Rajouter juste avant la réinitialisation du port C un test pour attendre que le clavier soit revenu à l'état initial (aucune touche enfoncée=>toutes les colonnes à '1')

Ajouter ensuite un appel au sous-programme TEMPO, pour attendre la **stabilisation du relâchement de la touche** avant le retour au programme principal.

PARTIE 2 : RECHERCHE DU NUMERO DE LA TOUCHE

Dans la partie 1, on a développé un sous-programme qui attend l'appui sur une touche, et qui renvoie le **code-clavier** correspondant à cette touche. Or, celui-ci n'est pas très utile: il serait préférable d'avoir le **numéro** de la touche (0, 1, 2, …). Il faut donc implémenter une routine de transcodage, pour passer du code clavier au numéro de touche.

Ce transcodage va consister à effectuer une recherche dans la table des codes-claviers (voir prépa), et à chercher si on y trouve le code récupéré au clavier. Au fur et à mesure qu'on parcours la table, ligne par ligne, on incrémente un compteur (B). Dès qu'on a trouvé le code, B contient la valeur de la touche entre 0 et 15.



2.1) Implémenter cet algorigramme sous forme d'un sous-programme TRANSCOD, avec comme programme principal les lignes ci-dessous.

```
DEBUT     jsr     LECTURE
          jsr     TRANSCOD
          jsr     AFFICH
          bra     DEBUT
```

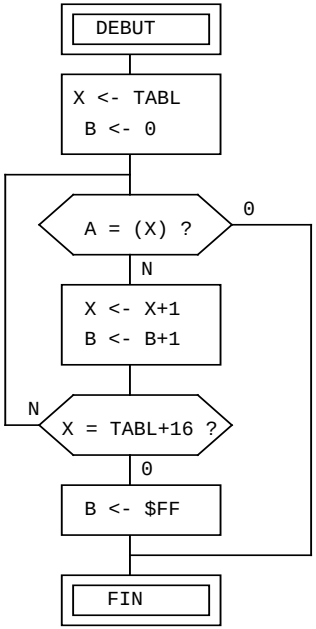
Remarque: le sous-programme AFFICH est celui permettant d'envoyer sur l'afficheur 7 segments le chiffre contenu dans B (voir TP 1ère année). Il faudra bien sûr saisir dans la zone de données la table "7 segments" associée.

Tester en pas à pas, puis en mode "*Run*" (avec la commande *Go*.

2.2) Défaut de cette version: il ne fonctionne pas en cas de code-clavier pas trouvé, par exemple si on appui sur plusieurs touches, le fonctionnement sera **imprévisible**.

Il faut par conséquent rajouter une **détection** de l'arrivée à la fin de la table: si plusieurs touches ont été enfoncées, le code récupéré ne correspondra à rien (= ne sera pas dans la table), et il faudra pouvoir **arrêter la recherche** après la lecture des 16 lignes de la table, et renvoyer un code d'errreur (ici, $FF) pour signaler à l'utilisateur un problème.

Modifier le sous-programme de la façon suivante:



Tester en pas à pas, et verifier qu'on obtient bien $FF en cas d'appui sur plusieurs touches.

Modifier le programme principal (**uniquement**) pour avoir l'allumage du segment g (tiret) en cas d'appui sur plusieurs touches.

Tester, puis imprimer le source, après validation par l'enseignant.

Rappels de notation pour les algorigrammes

`A <- VALEUR` : Donner à A la valeur de l'étiquette VALEUR

`A <- 45` : Donner à A la valeur 45

`A <- (VALEUR)` : Donner à A la valeur contenue à l'adresse VALEUR

`A <- (X)` : Donner à A la valeur contenue à l'adresse contenue dans X

TP n°2 : DOCUMENTS RESSOURCES

Programmation du sens des broches du Port C du 68hc11
registre DDRC (*Data Direction Register port C*)

Valeur du bit dans DDRC	rôle de la broche correspondante sur le PortC
0	Entrée
1	Sortie

Principe d'acquisition de la touche enfoncée (par la méthode du retournement du sens des lignes)

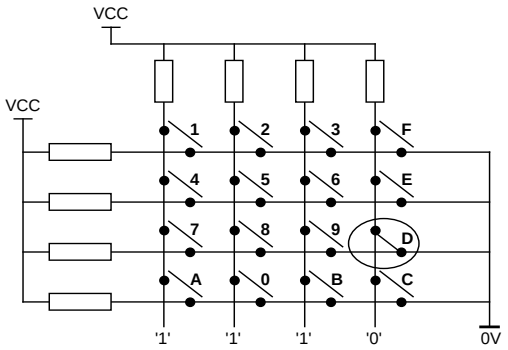
Initialisation

Il faut positionner les **lignes en sortie** et les **colonnes en entrée**, puis fixer les **lignes au niveau 0**. Le clavier au repos, les colonnes présentent alors toutes un niveau haut ('1').

Appui sur une touche

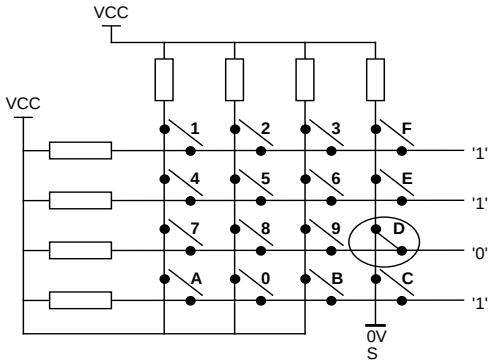
On suppose qu'on appuie sur la touche 'D'

La colonne de la touche pressée va passer au niveau '0' (contact avec la ligne, qu'on a positionné à '0' dans l'initialisation). On peut lire sur les colonnes la valeur '1110', puisque la touche D provoque un contact électrique entre la ligne 2 et la colonne 3. Le schéma équivalent est le suivant :



Après avoir lu le mot binaire des colonnes (1110), on procède au retournement du sens (les lignes passent en entrée et les colonnes en sortie)

On vient ensuite écrire sur les colonnes les niveaux logiques qu'on vient d'y lire. Dans l'exemple présent, les 3 premières colonnes à '1', et la dernière à '0', ce qui est représenté par le schéma équivalent suivant :



On n'a plus qu'à lire les lignes pour récupérer la valeur 1101

L'appui sur la touche 'D' donnera donc le code 1110 pour les colonnes, et 1101 pour les lignes. Ce code est **caractéristique** de cette touche: aucune autre touche ne peut générer ce code.

Pour lui attribuer ensuite une valeur correspondante avec le marquage de la touche('0000 1101' pour $0D par exemple), il faut procéder ensuite à un **transcodage**.

Important: Avant de pouvoir réaliser l'acquisition d'une autre touche, il faut repositionner le port dans son état initial.

Algorithme correspondant :

1) Positionner C3-C0 en entrée et L3-L0 en sortie
2) Ecrire la valeur $00 sur le port C
3) Lire le port C
4) Positionner C3-C0 en sortie et L3-L0 en entrée
5) Ecrire sur le port C la valeur lue au 3)
6) Lire le port C (dans l'accumulateur A)