Université Saint Joseph de Beyrouth

Ecole Supérieure d'Ingénieurs de Beyrouth – ESIB

Département Électrique et Mécanique

Cursus Génie Électrique – Semestre 3

Rapport du Projet
Construction d'un GBF

Trinôme

Saad Louis 180177

Sfeir Antoine 181812

Tabet Lynn 181373

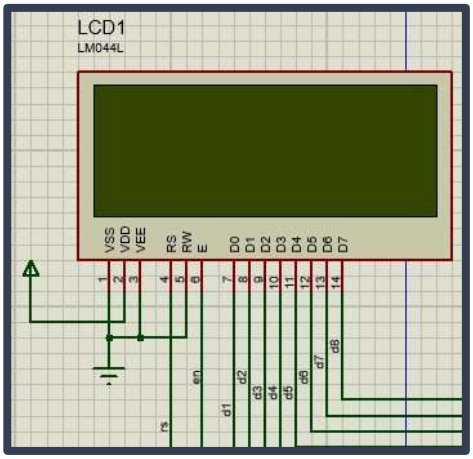
Encadrant

Dr. Jean Sawma

Contents

LCD | LM044L

On utilise ce LCD dans notre projet.
Le raisonnement derrière le fonctionnement de ce composant s'explique ainsi :

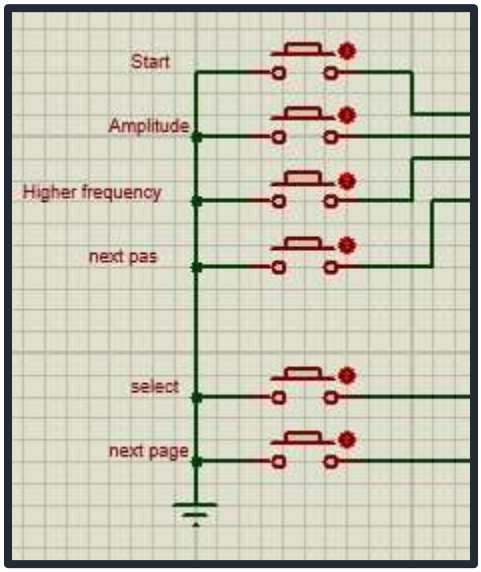


- Les bits D0…D7 permettent au LCD de communiquer avec d'autres composants. Dans notre cas, le LCD recevra des données seulement d'où le branchement de R/W au GROUND. E : fonctionne comme une clock et au falling edge la 'command' ou 'Data' sera admise par le microprocesseur. 'Command' ou 'Data' se décide par le pin RS.
- On a choisi de fonctionner le LCD par la méthode du Delay. En d'autres termes, on envoie une commande ou une data et on attend à ce que ce LCD finisse. On utilise timer1 pour faire ce Delay.
- A noter que certaines commandes prennent plus de temps pour se terminer que les 'data' c'est pourquoi on choisit de mettre plusieurs configurations de timer1. Une de 100µs et une de 1.64 ms.
- On a choisi d'écrire dans le LCD en permanence : dans la boucle infinie. Ainsi, on a pensé à faire un refresh du LCD chaque 100 ms. En d'autres termes après avoir fini l'écriture on demande au LCD d'attendre 100 ms avant de réécrire.

Logique de l'ensemble du projet

Passons maintenant au raisonnement concernant le projet tout est entier en passant par l'usage du LCD dans notre projet.

En regardant le schéma global de notre circuit on remarque la présence de plusieurs boutons. Ces boutons permettront l'explication du fonctionnement du LCD et donneront une vue globale du fonctionnement de ce dispositif.



A- Bouton : Next page

A la pression de ce bouton une nouvelle page sera affichée sur le LCD. Pour coder ceci, on a pensé à créer 2 variables : screen et counter. Ce bouton est connecté à une interruption basse priorité. Quand le bouton 'next page' est pressé la variable counter (initialement à 0) s'incrémente de 1. Puis, selon sa valeur et d'une façon successive les bits de la variable screen deviennent égaux à 1 selon leur tour. Ensuite et à la troisième pression (3 pages) tout est remis à 0.
Cette variable screen sera ensuite testé dans la boucle infinie pour savoir quel page afficher. Sachant qu'on a décidé d'utiliser un affichage continu dans la boucle infini pour répondre rapidement aux demandes de l'utilisateur du dispositif.

Variable **screen** :

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Bit 0 : set => on affiche page 2 ou page 3
Clear => on affiche screen 1
Bit 1 : set => on affiche page 3

B- Bouton : Select

Comme déjà expliqué, on a trois pages. Chaque page représente un signal à afficher. Quand le bouton 'Select' est pressé le signal affiché dans la page doit être choisi pour la sortie. Pour faire cela, on connecte ce bouton à une interruption basse priorité. A l'entrée de cette interruption on teste la variable counter pour savoir quel page on affiche maintenant. Ensuite, on modifie la variable 'select' pour qu'on mémorise le choix de l'utilisateur. Après avoir choisi notre signal, un '#' apparait à côté du signal voulue, ainsi que la LED yellow s'allume quand la page de la forme choisie est sélectionnée.

Variable **select** :

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Cleared : => rien est sélectionné
Bit 0 : set => sinus est sélectionné
Bit 1 : set => carré est sélectionné
Bit 2 : set => triangle est sélectionné

C- Bouton : Higher frequency

Ce bouton-là est relié à une interruption basse priorité. Quand il est pressé une nouvelle constante doit être mis dans timer0h et timer0l. Que mettre ? On prend 2 temps T_1 et T_2 les deux commandés par Timer0.

$$T_1 = \frac{4*prescaler*(2^{16}-Timer01)}{Fosc} \qquad T_2 = \frac{4*prescaler*(2^{16}-Timer02)}{Fosc}$$

$$T_1 - T_2 = \frac{4 * prescaler}{Fosc} * (Timer02 - Timer01)$$

Donc,

$$Timer02 = \frac{(T_1 - T_2) * Fosc}{4 * prescaler} + Timer01$$

Les choix donnés sur le LCD 0.1 ou 0.01 implique un $T_1 - T_2 = constante$ et donc on sait maintenant à chaque pression du bouton quelles valeurs on doit mettre dans timer0h et timer0l.
Et, ces valeurs seront manipulées à travers highvar et lowvar. Ces 2 choix seront manipulés par le bouton 'next pas'.

Pour mettre la nouvelle valeur dans Timer0 on ajouter cette constante $\frac{(T_1 - T_2)*Fosc}{4*prescaler}$ à Timer0 avant. Un problème se pose on doit additionner la plupart du temps deux nombres de 16 bits. Pour faire ceci on utilise une commande 'addwfc'. En d'autres termes, on additionne tout d'abord les 8 bits de poids faibles avec 'addwf' des 2 nombres de 16 bits. Puis, on utilise 'addwfc' pour les bits de poids fort , pour que s'il y avait un carry venant de l'addition des poids faibles qu'il s'ajoute aux poids fort.

NB : Nous sommes en train d'utiliser une fréquence d'opération du pic18f4520 de 20MHz, pour un meilleur comportement.

D- Bouton : Next Pas

Ce bouton est aussi relié à une interruption basse priorité. Quand ce bouton est pressé une variable select_period est incrémenté ou remis à 0 selon le cas. Ainsi en affichant dans la boucle infinie on peut tester et voir si on doit afficher 'selected' ('#') et la tester en pressant le bouton 'Higher frequency' quelle valeur ajouter au Timer0.

E- Bouton : Start

Le bouton start est connecté à une interruption haute priorité. Il permettra selon les variables qu'on a déjà manipuler avec les autres boutons disponibles de sélectionner le signale convenable. Pour comprendre ces instructions on doit expliquer comment les différentes formes sont stockés dans notre projet. En effet, on a pris chaque signal (triangle, carré, sinus), et on a calculé des valeurs de ce signal à un pas constant. Puis, on a stocké ces valeurs dans la mémoire de programme. Ainsi, et à chaque interruption de Timer0 le système prend une valeur après l'autre de la mémoire de programme. Et donc, dans l'interruption de ce bouton, on pointera sur le signal choisi en modifiant tblptrh et on lance le timer0.

F- Bouton : Amplitude

Ce bouton est relié à une interruption basse priorité qui manipulera l'amplitude du signal de sortie. On disposera de 2 variables 'ampl' et 'ampl_save'. 'ampl' sera utilisée pour compter le nombre de fois que ce bouton est pressé, alors que 'ampl_save' manipulera l'amplitude. En effet, à chaque pression du bouton, l'amplitude sera divisée par deux, et cela sera effectuer en faisant une rotation à droite des valeurs déjà stockées. Le nombre de rotation effectuée sera déterminé selon 'ampl' et on utilisera 'ampl_save' pour la décrémenter à chaque fois qu'on entre dans la boucle jusqu'à elle sera nulle et donc la valeur stockée dans la variable 'value' sera mise sur le port C. Cela intervient dans l'interruption du timer 0 lors de l'affichage du signal. Et lorsque toutes les valeurs après rotation deviennent nulle (7 rotations) la variable 'ampl' redevient zéro et on retrouve la valeur de l'amplitude initiale +5V/ -5V.

G- Explication du circuit extérieur au PIC18F4520

1. Dans notre projet, on va stocker les valeurs des signaux sinus, carrés et triangles and la mémoire de programme suivant les adresses suivant (0x700 pour le sinus, 0x800 pour le carré et 0x900 pour le triangle). Ces valeurs vont être appelées suivant le timer0 qu'on gèrera sa première valeur (la valeur d'où il commence à compter), et ensuite on les déposera sur le port C qui est relié à un DAC de 8-bits DAC0808 de tension de référence +10V et 0V, notre zéro sera placé au milieu des valeurs (comme les signaux sont alternatifs) et donc 8-bits donneront 256 ce qui donne que la valeur 127 (0x7f) représentera notre zéro, 255 (0xff) sera la valeur maximale et 0 (0x00) sera le minimum. Mais en faisant cela notre zéro aura une valeur après le DAC de +5V, comme 127 est le milieu et donc sera égale à +10V/2 = +5V, or on doit le ramener à une tension nulle. Cela implique l'utilisation d'un ampli (U4) qui va soustraire une tension manuellement choisie du signal obtenue, et donc en pleine amplitude on doit soustraire +5V du signal.

2. La manipulation de l'amplitude avec le bouton 'amplitude' (décrit avant) du signal sera faite par une division par 2 de l'amplitude précédente, et donc une rotation à droite des valeurs binaires stockées. Mais on doit réajuster notre zéro (Offset) par la résistance variable reliée à l'ampli U4, notre nouveau zéro se trouvera à 63 (0011 1111), qui est une valeur de +2.5V après DAC0808 et donc on doit retrancher +2.5V et non pas +5V, et ensuite de suite pour les autres amplitudes. Ceci peut être vue clairement dans la photo ci-dessous.