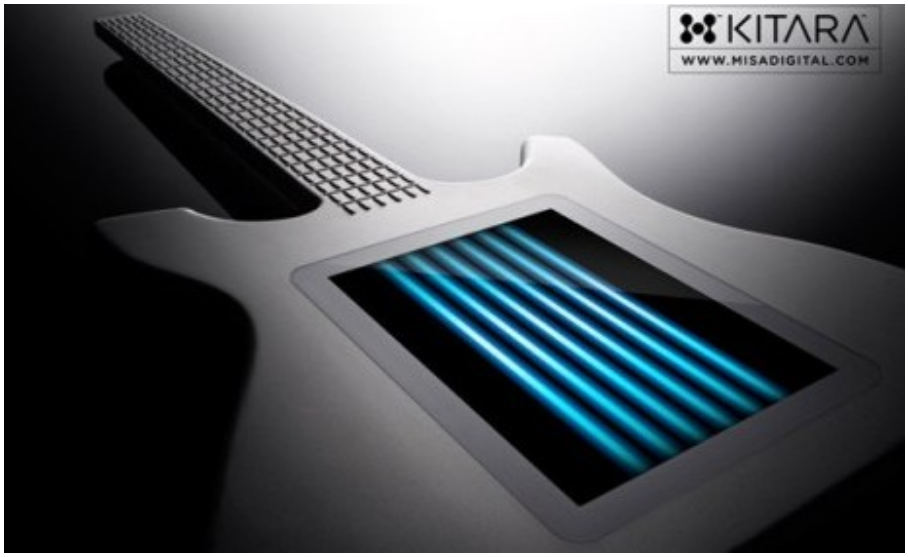
Université François-Rabelais de Tours
Institut Universitaire de Technologie de Tours
Département Génie Électrique et Informatique Industrielle

UNIVERSITE FRANCOIS-RABELAIS
TOURS
Institut Universitaire de Technologie
Département
GENIE ELECTRIQUE ET
INFORMATIQUE INDUSTRIELLE

Projet Tuteuré – Semestre 3

La Guitare électronique



SOUFFEZ Thomas - SAVRY Nicolas
Q2
Promotion 2010-2012

Enseignants :
M LEQUEUX Thierry
M AUGER Phillipe

Université François-Rabelais de Tours
Institut Universitaire de Technologie de Tours
Département Génie Électrique et Informatique Industrielle

UNIVERSITE FRANCOIS-RABELAIS
TOURS
Institut Universitaire de Technologie
Département
GENIE ELECTRIQUE ET
INFORMATIQUE INDUSTRIELLE

Projet Tuteuré – Semestre 3

La guitare électronique

SOUFFEZ Thomas - SAVRY Nicolas
Q2
Promotion 2010-2012

Enseignants :
M LEQUEUX Thierry
M AUGER Phillipe

Sommaire

Introduction du projet

Le but du projet est de créer une guitare électronique : c'est à dire une guitare dont le fonctionnement est basé sur un circuit électrique qui comprendra matériaux électroniques et informatiques. Cette innovation permettra donc d'obtenir une guitare autonome qu'il ne sera pas nécessaire d'accorder, ce qui implique donc un gain de temps pour l'utilisateur ainsi qu'une utilisation simplifiée. De plus, avec son, système de cordes et de frettes, il reflétera au mieux le principe de fonctionnement réel d'une guitare. Pour cela, nous avons donc du établir un cahier des charges afin de répondre aux critères que nous nous étions fixés et qui seront présentés un peu plus tard. Pour innover au mieux le fonctionnement, nous avons fait des recherches afin de nous baser sur ce qui existait déjà et nous avons donc pu créer un système complètement nouveau.

1. Présentation détaillée

1.1. Fonctions et contraintes

Le système étudié devra réaliser les fonctions suivantes :

✗ A partir d'une source d'alimentation continue 0-15V, créer sa propre alimentation continue 5V.

✗ Détecter l'appui sur les cordes et les frète afin de sortir une note musicale cohérente et donc refléter au mieux le fonctionnement réel d'une guitare.

Les contraintes de son fonctionnement sont les suivantes :

✗ Le coût de mise en œuvre ne devra pas excéder 100€.

✗ La maquette devra être suffisamment robuste pour permettre une utilisation sans risque de dégradation.

✗ Le circuit électrique devra inclure une interface PC/microcontrôleur accessible afin de modifier à tout moment le programme informatique.

✗ Le système devra respecter les normes de sécurité et de CEM afin de ne pas déranger les systèmes se trouvant à proximité.

1.2. Maquette à réaliser

La maquette que nous devrons réaliser sera donc la suivante :

Elle devra intégrer tous les éléments qui, ici, sortent de la carte électronique.

✗ Une matrice de cordes et frettes.

✗ 4 autres cordes pour simuler le « grattage » des cordes.

✗ Ainsi qu'une interface pour relier la pile, une interrupteur, une diode, un médiator et une prise Jack.

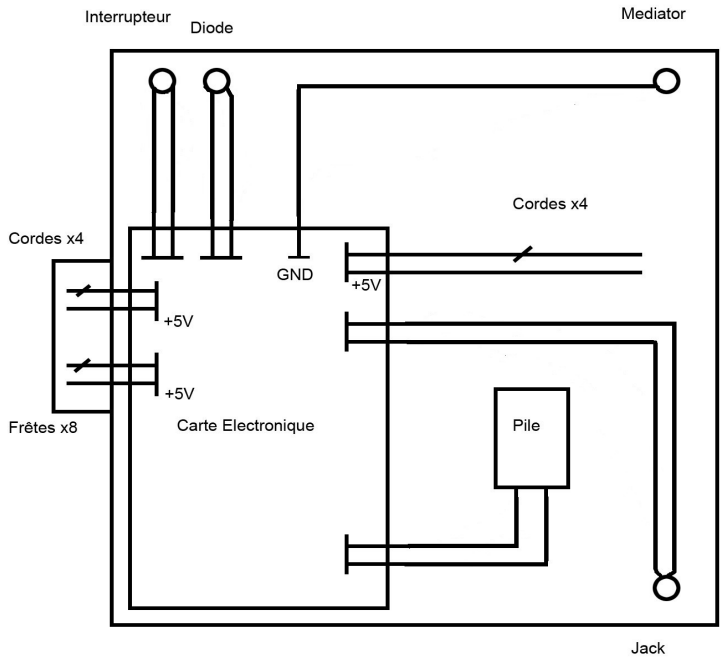


Illustration 1: Schéma de la maquette à réaliser

1.3. Schéma fonctionnel

La guitare devra donc être capable d'effectuer les tâches suivantes :

✗ FP1 : Scrutation colonne : son but est de détecter sur quelle corde et quelle frète l'utilisateur a appuyé sur le manche de la guitare donc quelle note celui-ci veut jouer.

✗ FP2 : Détection de cordes : elle a pour rôle de déterminer quelle corde a été grattée sur la partie principale de la guitare.

✗ FP3 : Traitement des données : Cette fonction traite les données passées par les fonction FP1 et FP2 dans l'optique de créer le signal de sortie.

✗ FP4: Mise en forme du signal : Son rôle est de créer le signal de sortie, à partir du traitement de données réalisée dans la fonction FP3.

✗ FP5 : Amplification : Cette fonction amplifie le signal en sortie de l'ATMEGA afin d'augmenter la puissance du son sortant de l'enceinte.

✗ FA : Alimentation : Elle a pour but d'alimenter l'ensemble des élément du montage en 5V à partir d'une alimentation continue comprise entre 0 et 30V.

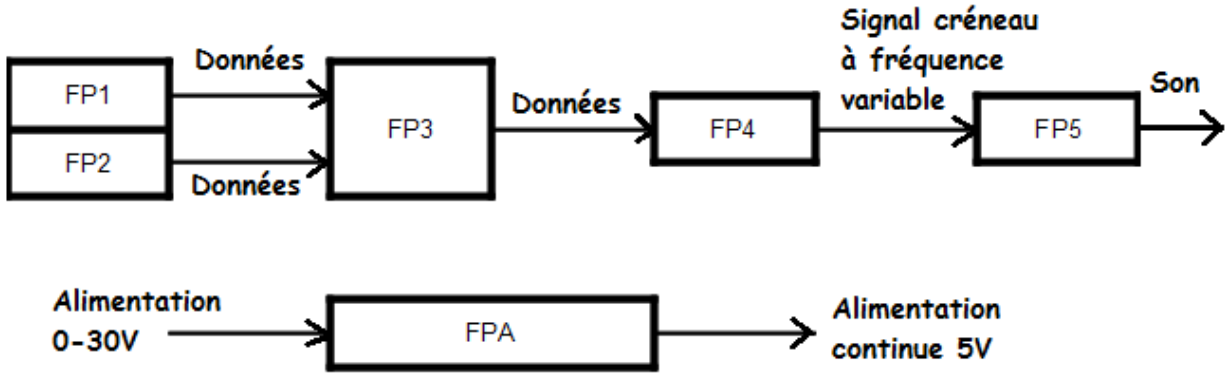


Illustration 2: Schéma fonctionnel du système

1.4. Planification

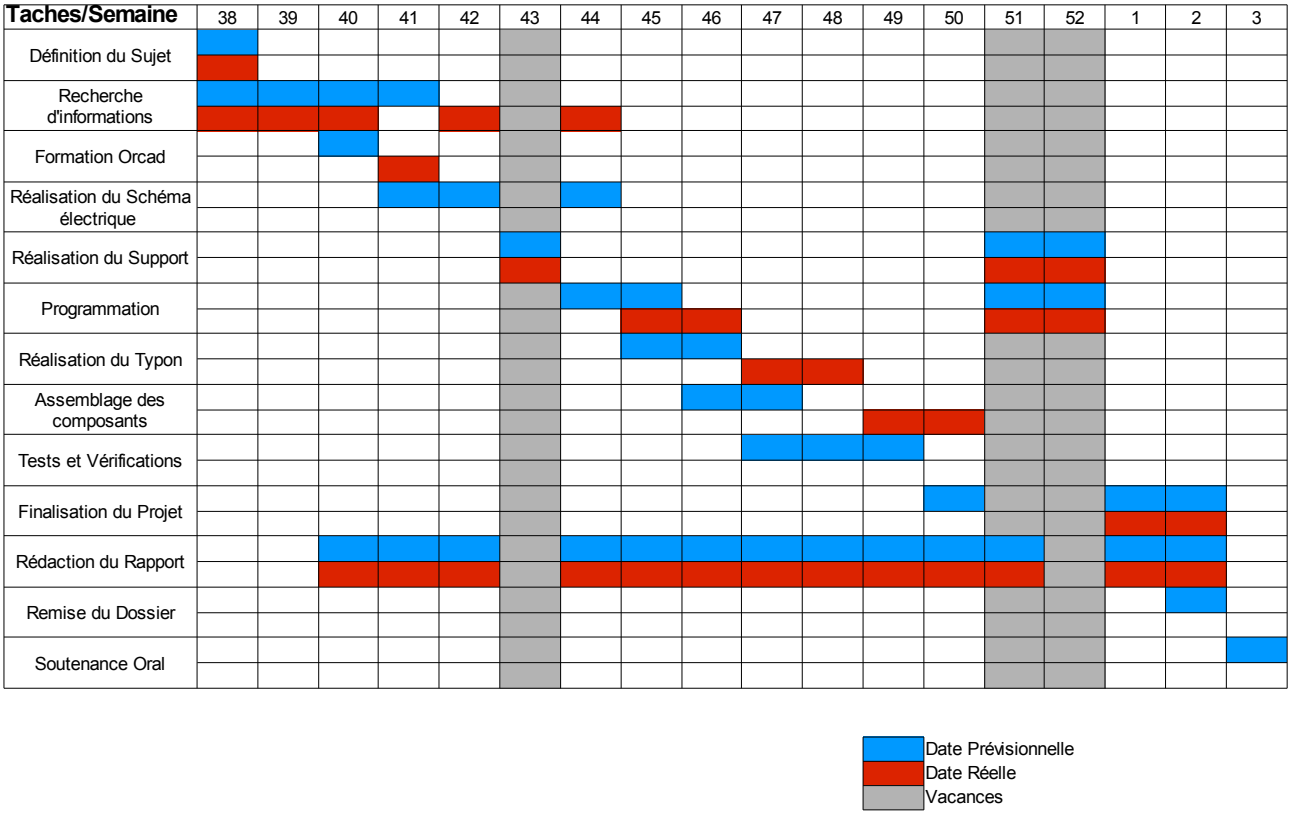


Illustration 3: Planning prévisionnel et réel du projet

2. Solutions techniques

2.1. FP1 : La scrutation colonne

Le principe de la scrutation colonne est de créer une matrice qui va permettre de situer où l'utilisateur a appuyé. Le schéma de câblage est le suivant :

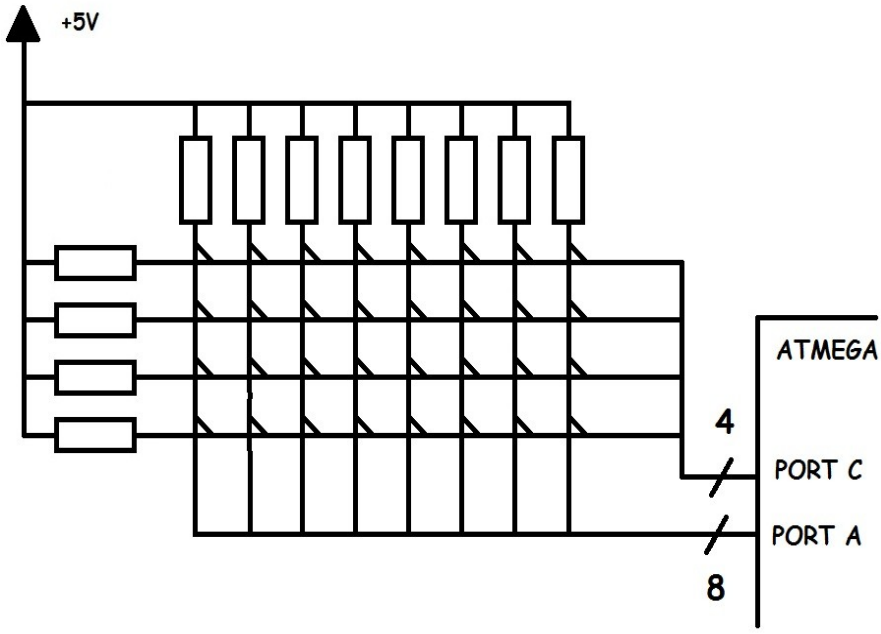


Illustration 4: Matrice cordes/frètes de la scrutation colonne

D'un côté, chaque corde est reliée au 5V par le biais d'une résistance de tirage et de l'autre à une broche de l'ATMEGA. Nous avons décidé de relier les cordes au PORT C et les frètes au PORT A. Ensuite, à l'aide l'ATMEGA, et grâce à un protocole de scrutation préalablement défini, nous allons être en mesure d'obtenir un code reflétant la position souhaitée par l'utilisateur. Ce protocole est le suivant :

- ✗ Imposer sur le port des cordes un niveau logique 0.
- ✗ Lire la valeur du bus des frètes et la garder en mémoire.
- ✗ Imposer cette valeur sur ce même bus.
- ✗ Lire la valeur du bus des cordes et la garder en mémoire.

Les deux valeurs ainsi stockées forment un code, image de la note souhaitée par l'utilisateur. La meilleure façon de se rendre compte du fonctionnement est de l'illustrer à l'aide d'un exemple. Dans l'exemple qui suit, nous utiliserons une matrice de dimension 4x4 pour simplifier la compréhension sachant que le principe pour une matrice 4x8 comme celle que nous utilisons reste le même :

Notre matrice est créée de la même façon que précédemment.

La première étape est de déclarer le bus de frètes en entrée et le bus de corde en sortie sur lequel on va imposer un niveau logique 0. Ensuite, supposons que l'appui se fasse sur la place entourée sur le schéma ci-dessous alors la frète de droite va entrer en contact avec la 3ème corde et ainsi passer au niveau logique 0 à son tour. Le premier code obtenu est donc '1110 '.

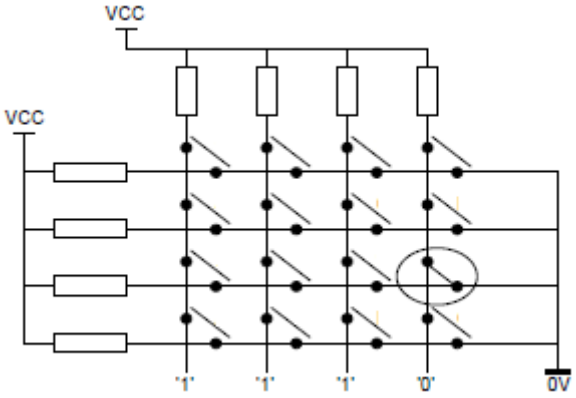


Illustration 5: Exemple matrice 4x4 n°1

Ensuite, il faut inverser les rôles : déclarer le bus des cordes en entrée et celui des frètes en sortie sur lequel on va inscrire le code précédemment obtenu : '1110'.

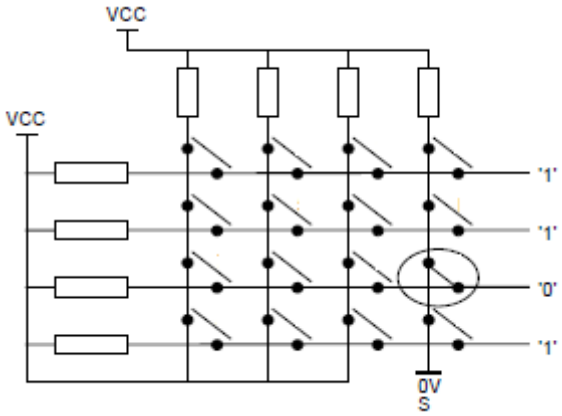


Illustration 6: Exemple matrice 4x4 n°2

Le code lu sur le bus de frète est donc '1101'. La combinaison des codes va permettre au microcontrôleur de situé où l'appui a donc été fait et en conséquence va être capable de connaître le note qu'il va devoir jouer. Tout cette analyse est donc appelée scrutation colonne. Nous allons voir juste après la façon de réaliser le protocole du point de vue du microcontrôleur donc du point de vue informatique.

Voici donc la partie du programme que nous avons réalisé et implanté dans notre système et qui concerne la partie scrutation colonne :

```
void Scrutation(int temp1, int temp2)      //Définition de la fonction "Scrutation"
{                                          //Début de la fonction
        DDRC = 0xFF;                       //Définition du PORTC en en Sortie
        PORTC = 0x00;                      //Mise a 0 du PORTC

        DDRA = 0x00;                       //Définition du PORTA en Entrée
        PORTA = 0x00;                      //Sans résistance de Tirage

        temp1 = PINA;                      //Stocker La Valeur du PortA dans la variable temp1

        DDRA = 0xFF;                       //Définition du PORTA en Sortie
        PORTA = temp1;                     //Mettre la valeur de la variable temp1  dans le PortA

        DDRC = 0x00;                       //Définition du PORTC en Entrée
        PORTC = 0x00;                      //Sans résistance de Tirage

        temp2 = PINC;                      //Stocker La Valeur du PortC dans la variable temp2
}                                          //Fin de la fonction
```

2.2. FP2 : Détection de corde

La matrice de cordes et de frètes n'est pas le seul composant dont le microcontrôleur aura besoin afin de générer un signal permettant d'obtenir une note. Il y a aussi une deuxième zone contenant des cordes dont le but est de lancer l'envoi du signal de sortie, donc c'est cette partie là qui va permettre de « lancer le départ » : lorsque l'une des cordes est « grattée », on obtient un son en sortie.

Pour réaliser cela, il nous faut utiliser les entrées d'interruption de l'ATMEGA. Le problème est que nous n'avons à notre disposition que 3 interruptions alors que notre maquette possède 4 cordes. Pour cela, nous avons du avoir recourt à un système comprenant une porte logique ET reliée à une entrée d'interruption (INT0). Le schéma de réalisation est le suivant :

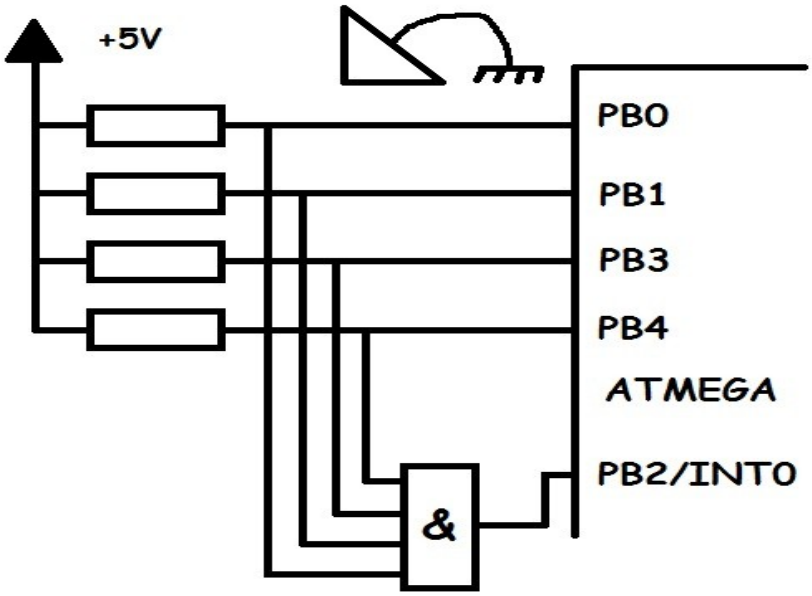


Illustration 7: Schéma électrique des cordes d'interruption

Au repos, toutes les cordes sont au niveau logique 1. Lorsque que l'on gratte l'une des cordes avec le fil relié à la masse (visible sur le haut du schéma électrique), cette corde passe à 0. Donc la sortie de la porte logique ET va passer à 0 à son tour ce qui va déclencher l'interruption. Une fois l'interruption déclenchée, l'ATMEGA va lire l'état des bits PB0, PB1, PB3 et PB4. Le bit qui sera au niveau logique 0 permettra donc de connaître avec exactitude quelle corde a été utilisée.

Le programme correspond à cette partie est le suivant :

```
interrupt[EXT_INT0] void Detection(void) //Définition de la fonction interruption externe 0 "Detection"
{                                        //Début de la fonction
        if(PINB.0==0)                    //Si la Broche 0 du Port B est egale a 0
                Corde = 1;               //Corde est agale a 1
        if(PINB.1==0)                    //Si la Broche 1 du Port B est egale a 0
                Corde = 2;               //Corde est agale a 2
        if(PINB.3==0)                    //Si la Broche 3 du Port B est egale a 0
                Corde = 3;               //Corde est agale a 3
        if(PINB.4==0)                    //Si la Broche 4 du Port B est egale a 0
                Corde = 4;               //Corde est agale a 4
}                                        //Fin de la fonction
```

2.3. FP3 : Traitement des données

Après avoir effectué des systèmes permettant de situer l'appui de l'utilisateur sur la matrice cordes/frètes et de savoir quelle corde à été grattée, il va nous falloir gérer ces informations. Pour cela, tout se passe cette fois au niveau informatique :

```
void main(void)                         //Début de la fonction "Main"
{
        Init_reg();                     //Utilisation de la Fonction "Init_reg"
        TCCR2 = 0x00;                   //Mise à 0 du registre TCCR2
        while(1)                        //Tant que 1
        {                               //Alors

                Scrutation(Frete_Scr, Corde_Scr);       //Utilisation de la fonction Scrutation, avec pour
paramètre Frete_Scr et Corde_Scr
                if(Corde == 1 && PINB.0 == 1)           //Si la variable Corde est égale a 1 et si La
broche 0 du port B est egale a 1
                {                                       //Alors
                        Corde = 0;                      //Corde est égale a 0
                        Corde_1(Frete_Scr, Corde_Scr);  //Utilisation de la fonction Corde_1
                }
                else if(Corde == 2 && PINB.1 == 1)      //Sinon si la variable Corde est égale a 2 et si La
broche 1 du port B est egale a 1
                {                                       //Alors
                        Corde = 0;                      //Corde est égale a 0
                        Corde_2(Frete_Scr, Corde_Scr);  //Utilisation de la fonction Corde_2
                }


                else if(Corde == 3 && PINB.3 == 1)      //Sinon si la variable Corde est égale a 3 et si La
broche 3 du port B est egale a 1
                {                                       //Alors
                        Corde = 0;                      //Corde est égale a 0
                        Corde_3(Frete_Scr, Corde_Scr);  //Utilisation de la fonction Corde_3
                }
                else if(Corde == 4 && PINB.4 == 1)      //Sinon si la variable Corde est égale a 2 et si La
broche 4 du port B est egale a 1
                {                                       //alors
                        Corde = 0;                      //Corde est égale a 0
                        Corde_4(Frete_Scr, Corde_Scr);  //Utilisation de la fonction Corde_4
                }
                else                                    //Sinon
                        TCCR2 = 0x00;                   //Mise a 0 du Registre TCCR2

        }                                               //Fin de tant que 1
}                                                       //Fin de la Fonction Main
```

Le programme principal, en fonction de la corde grattée va nous renvoyer dans une sous fonction qui va permettre de créer une note en fonction des résultats de la scrutation colonne :

```
void Corde_1(int temp1, int temp2)                  //Définition de la fonction interruption externe 0 "Detection"
{                                                   //Début de la fonction
        if(temp2 == (0x80 || temp2))                //Si temps2 est egale 0x80
        {                                           //Alors
                if(temp1 == (0x01 || temp1))        //Si temps1 est egale 0x01
                        Proto_Note(Do1,noire);      //Alors lancer la fonction Proto_Note de la durée d'une noire et de
la note Do1
                else if(temp1 == (0x02 || temp1))   //Sinon si temps1 est egale 0x02
                        Proto_Note(Si1,noire);      //Alors lancer la fonction Proto_Note de la durée d'une noire et de
la note Si1
                else if(temp1 == (0x04 || temp1))   //Sinon si temps1 est egale 0x04
                        Proto_Note(Lad1,noire);     //Alors lancer la fonction Proto_Note de la durée d'une noire et de
la note Lad1
            else if(temp1 == (0x08 || temp1))       //Sinon si temps1 est egale 0x08
                        Proto_Note(La1,noire);      //Alors lancer la fonction Proto_Note de la durée d'une noire et de
la note La1
                else if(temp1 == (0x10 || temp1))   //Sinon si temps1 est egale 0x10
                        Proto_Note(Sold1,noire);    //Alors lancer la fonction Proto_Note de la durée d'une noire et de
la note Sold1
                else if(temp1 == (0x20 || temp1))   //Sinon si temps1 est egale 0x20
                        Proto_Note(Sol1,noire);     //Alors lancer la fonction Proto_Note de la durée d'une noire et de
la note Sol1
                else if(temp1 == (0x40 || temp1))   //Sinon si temps1 est egale 0x40
                        Proto_Note(Fad1,noire);     //Alors lancer la fonction Proto_Note de la durée d'une noire et de
la note Fad1
                else if(temp1 == (0x80 || temp1))   //Sinon si temps1 est egale 0x80
                        Proto_Note(Fa1,noire);      //Alors lancer la fonction Proto_Note de la durée d'une noire et de
la note Fa1
        }
        else                                        //Sinon
        {
                Proto_Note(Mi1,noire);              //Alors lancer la fonction Proto_Note de la durée d'une noire et de
la note Mi1
        }
}                                                   //Fin de la fonction
```

On aperçoit dans le programme précédent, une fonction Proto_Note : c'est cette fonction qui va permettre de sortir le signal permettant d'obtenir un son. Son fonctionnement est expliqué dans la partie suivante.

2.4. FP4 : Mise en forme du signal

Afin de créer le signal qui permettra d'obtenir la bonne note, nous allons utiliser la fonction compteur2 de l'ATMEGA. Cette fonction permet d'obtenir en sortie (OC2) un signal de type créneau à fréquence variable. Le principe de fonctionnement est donc basé sur une variable (TCNT2) qui s'incrémente (d'où le nom compteur) jusqu'à un certain seuil (OCR2) que l'utilisateur a défini. Lorsque la valeur de cette variable atteint la tension de seuil pré-définie, la sortie est complémentée : c'est-à-dire que si celle-ci était à un niveau logique 1, elle passe à 0 et vice versa. En réglant la valeur du seuil, on va donc pouvoir agir sur la période du signal donc sur sa fréquence. L'illustration suivante permet de bien se rendre compte du fonctionnement :

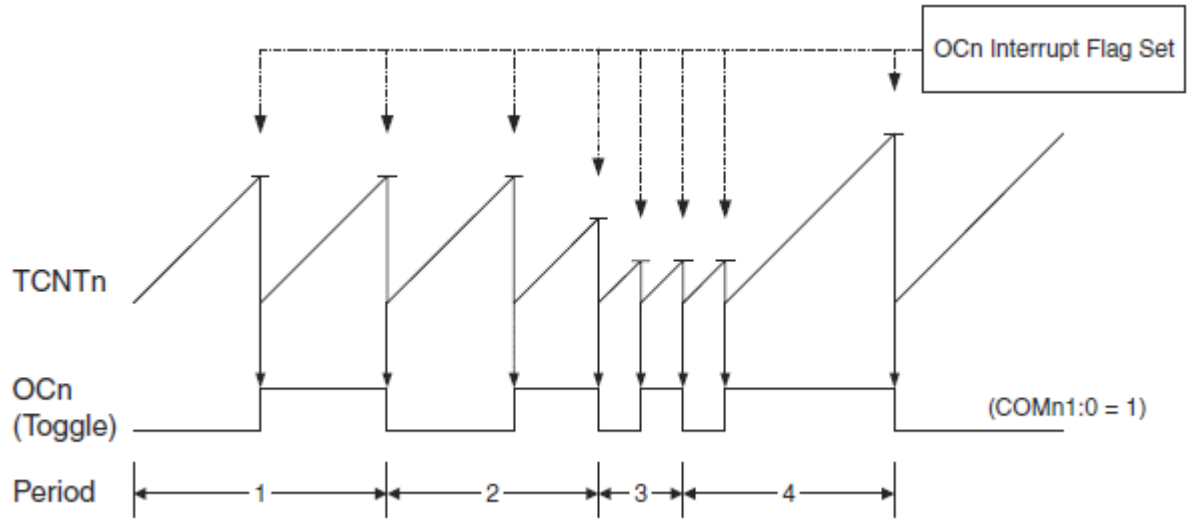


Illustration 8: Illustration du fonctionnement du compteur TCNT2

Donc pour chacune de nos notes, il va nous falloir définir la tension de seuil à utiliser afin d'obtenir la bonne fréquence du signal de sortie donc le bon son. Pour cela, nous allons nous servir de la formule fournie par la documentation de l'ATMEGA :

$$F = \frac{fclk}{2.N.(1+OCR2)}$$

donc $\boxed{OCR2 = \frac{fclk}{2.N.F} - 1}$

Voici donc le premier tableau contenant les fréquences de nos notes :

	Frète 0	Frète 1	Frète 2	Frète 3	Frète 4	Frète 5	Frète 6	Frète 7
Corde 1	82,4	87,3	92,5	98	103,8	110	116,5	123,5
Corde 2	110	116,5	123,4	130,8	138,6	146,8	155,6	164,8
Corde 3	146,8	155,6	164,8	174,6	185	196	207,6	220
Corde 4	196	207,6	220	233,1	247	261,7	277,2	293,7

Illustration 9: Tableau des différentes fréquences utilisées

Et un second contenant les valeurs de seuil à déclarer, afin d'obtenir un signal de sortie à la fréquence correspondante à la note jouée :

	Frète 0	Frète 1	Frète 2	Frète 3	Frète 4	Frète 5	Frète 6	Frète 7
Corde 1	189	178	168	158	150	141	133	126
Corde 2	141	133	126	118	112	105	99	94
Corde 3	105	99	94	88	83	79	74	70
Corde 4	79	74	70	66	62	59	55	52

Illustration 10: Tableau des valeurs de seuil correspondants aux fréquences

Afin de faciliter la programmation, il est utile de définir les notes et de leurs associer leur valeur de seuil :

```
#define Mi1 189          //189 est défini comme la note Mi1
#define Fa1 178          //178 est défini comme la note Fa1
#define Fad1 168         //168 est défini comme la note Fad1
#define Sol1 158          //158 est défini comme la note Sol1
#define Sold1 150        //150 est défini comme la note Sold1
#define La1 141          //141 est défini comme la note La1
#define Lad1 133         //133 est défini comme la note Lad1
#define Si1 126          //126 est défini comme la note Si1
#define Do1 118          //118 est défini comme la note Do1
#define Dod1 112         //112 est défini comme la note Dod1
#define Re1 105          //105 est défini comme la note Re1
#define Red1 99          //99 est défini comme la note Red1
#define Mi2 94           //94 est défini comme la note Mi2
#define Fa2 88           //88 est défini comme la note Fa2
#define Fad2 83          //83 est défini comme la note Fad2
#define Sol2 79          //79 est défini comme la note Sol2
#define Sold2 74         //74 est défini comme la note Sold2
#define La2 70           //70 est défini comme la note La2
#define Lad2 66          //66 est défini comme la note Lad2
#define Si2 62           //62 est défini comme la note Si2
#define Do2 59           //59 est défini comme la note Do2
#define Dod2 55          //55 est défini comme la note Dod2
#define Re2 52           //52 est défini comme la note Re2
#define Red2 49          //49 est défini comme la note Red2
```

1. Une fois nos valeurs de seuil définies, il ne nous reste plus qu'à créer la fonction qui va produire le signal de sortie :

```
void Proto_Note(int note_a_faire, int attente)      //Définition de la fonction "Proto_Note"
{                                                   //Début de la fonction
        TCCR2 = 0x9E;                               //Mise a 0x9E du registre TCCR2
        Note = note_a_faire;                        //La variable Note est égale a la variable note_a_faire
        delay_ms(attente);                          //Attente
        Pause(10);                                  //Utilisation de la Fonction Pause
}                                                   //Fin de la fonction
```

Le seul problème qui existe dans cette solution est que le signal de sortie n'est pas centré sur le 0 : il faut donc supprimer la composante continue afin d'obtenir le signal souhaité. Pour cela, il suffit de relier directement à la sortie un condensateur de 100nF :

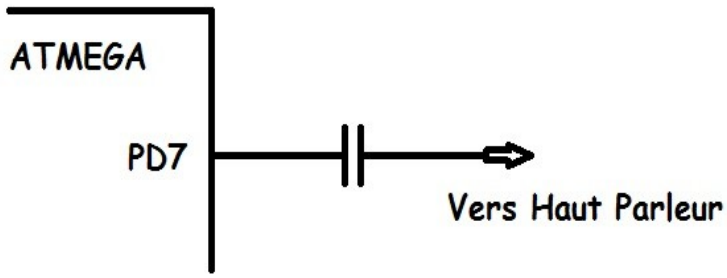


Illustration 11: Schéma du circuit de suppression de la composante continue

2.5. FP5 : Amplification

La fonction amplification a été envisagée mais, faute de temps, n'a pas pu être réalisée. Nous nous pencherons dessus lors du semestre 4 et en attendant nous utiliserons l'amplification interne contenu dans les enceintes qui nous servent à sortir le son.

2.6. FA : Fonction alimentation

Pour l'alimentation, nous allons utiliser un circuit à régulateur à découplage permettant d'obtenir une source de tension continue 5V à partir d'une tension continue comprise entre 0 et 30V .

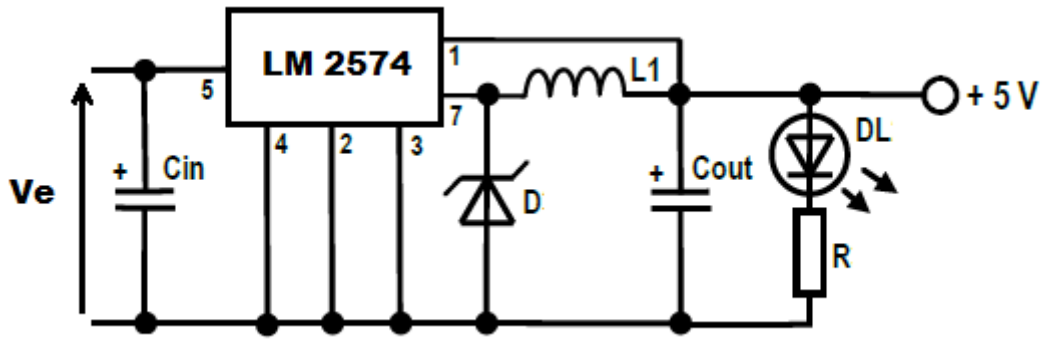


Illustration 12: Circuit d'alimentation +5V

Pour réaliser la tension d'entrée, nous utiliserons une pile 9V. Cette tension 5V va nous permettre par la suite d'alimenter tout notre circuit : notre matrice cordes/frètes, nos cordes de détection, notre ATMEGA8535 ainsi que notre porte logique ET.

3. La maquette

La maquette a été réalisée en bois pour avoir une robustesse suffisante afin de la bouger assez régulièrement de supporter la tension qu'il y a sur les cordes. De plus, le bois permet d'être travaillé sans trop de difficultés. Voici des photos de la maquette une fois terminée :

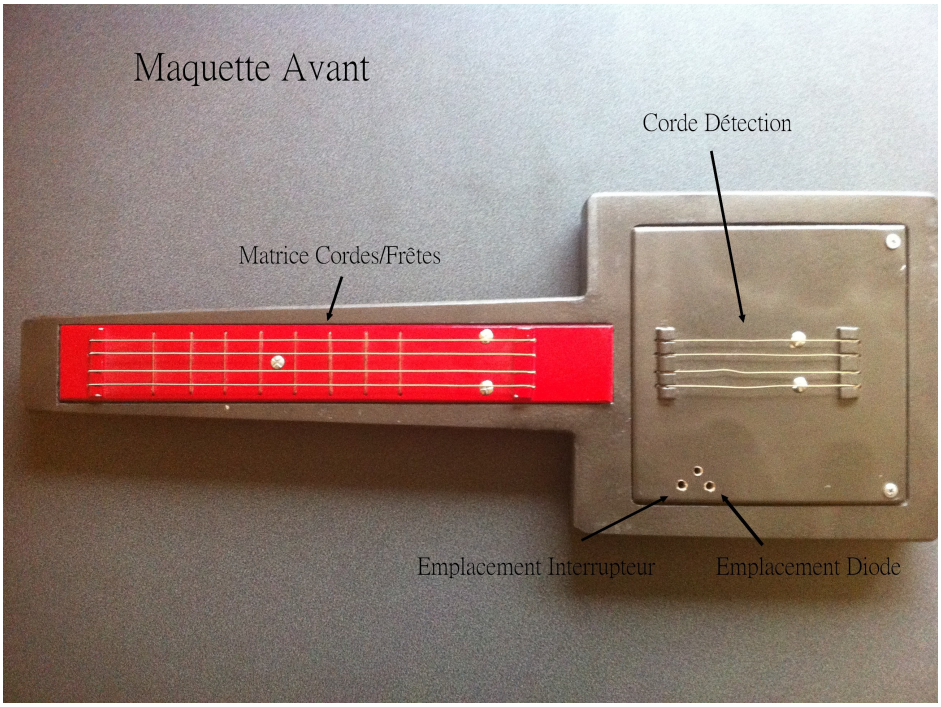


Illustration 13: Face avant de la maquette

Cette vue de la face avant de la guitare permet de bien situer les deux éléments contenant les cordes : sur la manche il y a la matrice créée par les cordes et les frettes et sur le « corps », on trouve les 4 cordes de détections aussi dîtes cordes de grattage.

On aperçoit bien que les différentes parties sont démontables donc l'intérieur est facilement accessible en cas de panne.

Mais ce n'est pas la seule voie d'accès : le panneau arrière est aussi démontable et c'est d'ailleurs par là que l'on va pouvoir introduire la carte électrique.

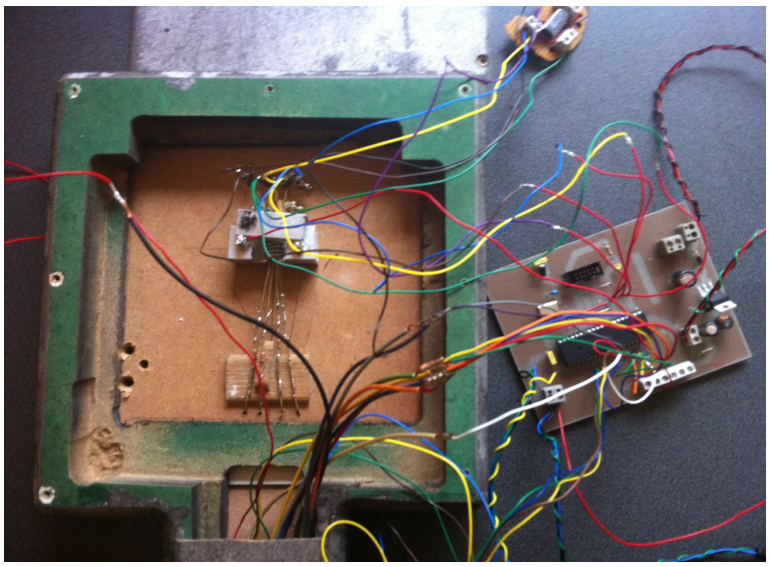


Illustration 14: Vue arrière de la maquette avec panneau démonté

4. Problèmes et tests

4.1. Problèmes rencontrés

Nous avons pour ce projet eu beaucoup de problèmes. Nous avons déjà dû trouver des moyens techniques pour pouvoir être conforme au cahier des charges, et nous avons dû abandonner certaines choses prévues au départ, comme le fait de vouloir intégrer un haut parleur nécessitant une amplification interne. Nous avons dû abandonner cette idée car nous ne pouvions pas gérer une amplification faute de temps.

Nous avons aussi fait un oubli dans la réalisation du circuit imprimé, et nous avons dû intégrer une carte annexe pour pouvoir réaliser la fonction de détection de cordes.

Nous voulions aussi créer le son, en forme de sinusoïde, mais nous avons finalement opté pour un son à forme « carré », créé plus facilement par le microcontrôleur.

Des problèmes ont été rencontrés au moment de souder les composants et plus particulièrement les borniers, car nous avions prévu un espace trop petit pour pouvoir les souder correctement. Nous avons donc dû souder les fils reliés aux cordes et aux frettes directement sur la carte.

4.2. Tests et validation

Nous avons réalisé plusieurs tests avant de se lancer dans la conception de la carte électronique, dont la façon de créer un signal « carré » avec le microcontrôleur. Nous avons finalement trouver une solution et réussi.

Un fois la carte électronique imprimée et soudée, nous avons procédé aux vrais tests et à la validation de la carte. Nous avons déjà testé la partie alimentation pour vérifier que le système nous fournissait bien du 5V continu.

Une fois ceci vérifié, nous avons testé la bonne mise en œuvre du système de programmation. Nous avons donc pu introduire le programme dans le microcontrôleur et vérifier son bon fonctionnement.

Nous avons aussi dû vérifier que la carte annexe comportant la porte logique 'ET' fonctionnait, ce qui était le cas, en modifiant une ligne e programme défectueuse.

Enfin, après mainte petites modifications du programme, nous avons réussi à faire marcher notre guitare.

Conclusion

Nous avons donc réussi à créer une guitare telle que nous le souhaitions, excepté quelques détails qui ne nous ont pas empêché de terminer le projet. Au final, nous obtenons une guitare quasi-autonome dans le sens où seule la partie création du son (amplification et enceintes) n'est pas présente. Pour remédier à cela, nous utiliserons une enceinte externe mais comptons bien incorporer cette partie manquante lors du prochain semestre. Sinon, le fonctionnement est bien validé, il reflète plutôt bien le fonctionnement d'une guitare avec toutefois un son plus électronique ce qui est normal compte tenu de la nature du système utilisé.

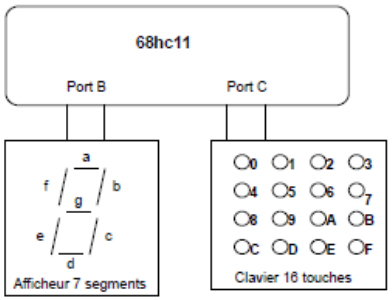
Table des illustrations

TP n°2 : LECTURE DE CLAVIER MATRICIEL EN SCRUTATION

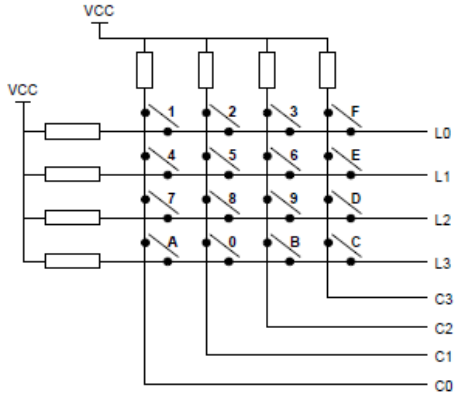
Introduction

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.

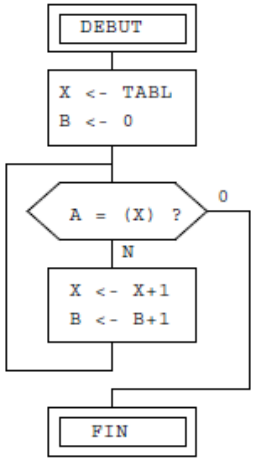
Rajouter juste avant la réinitialisation du port C un test pour attendre que le clavier soit revenu à l'état initial (aucune touche enfoncée=>toutes les colonnes à '1')

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
```

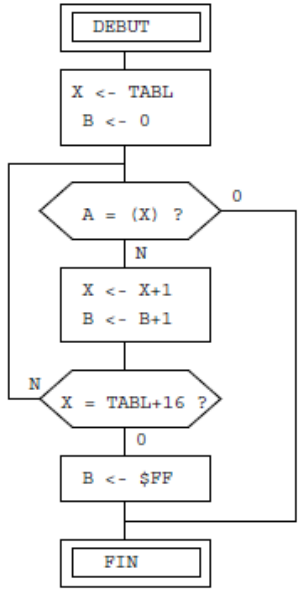
Remarque: le sous-programme AFFICH est celui permettant d'envoyer sur l'afficheur 7 segments le chiffre contenu dans B (voir TP 1ère année). Il faudra bien sûr saisir dans la zone de données la table "7 segments" associée.

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)

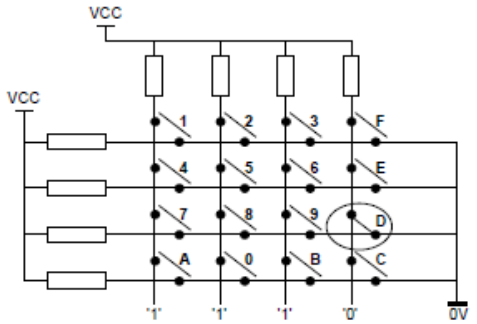
Initialisation

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

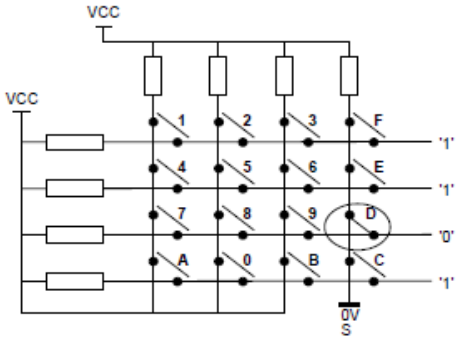
On suppose qu'on appuie sur la touche 'D'

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)

Annexe 2 : Documentation du régulateur LM2574

National Semiconductor

June 1999

LM2574/LM2574HV
SIMPLE SWITCHER™ 0.5A Step-Down Voltage Regulator

General Description

The LM2574 series of regulators are monolithic integrated circuits that provide all the active functions for a step-down (buck) switching regulator, capable of driving a 0.5A load with excellent line and load regulation. These devices are available in fixed output voltages of 3.3V, 5V, 12V, 15V, and an adjustable output version.

Requiring a minimum number of external components, these regulators are simple to use and include internal frequency compensation and a fixed frequency oscillator.

The LM2574 series offers a high-efficiency replacement for popular three-terminal linear regulators. Because of its high efficiency, the copper traces on the printed circuit board are normally the only heat sinking needed.

A standard series of inductors optimized for use with the LM2574 are available from several different manufacturers. This feature greatly simplifies the design of switch-mode power supplies.

Other features include a guaranteed ±4% tolerance on output voltage within specified input voltages and output load conditions, and ±10% on the oscillator frequency. External shutdown is included, featuring 50 µA (typical) standby current. The output switch includes cycle-by-cycle current limiting, as well as thermal shutdown for full protection under fault conditions.

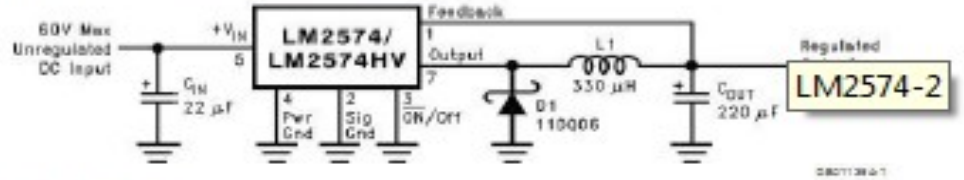
Features

- 3.3V, 5V, 12V, 15V, and adjustable output versions
- Adjustable version output voltage range, 1.23V to 37V (57V for HV version) ±4% max over line and load conditions
- Guaranteed 0.5A output current
- Wide input voltage range, 40V, up to 60V for HV version
- Requires only 4 external components
- 52 kHz fixed frequency internal oscillator
- TTL shutdown capability, low power standby mode
- High efficiency
- Uses readily available standard inductors
- Thermal shutdown and current limit protection

Applications

- Simple high-efficiency step-down (buck) regulator
- Efficient pre-regulator for linear regulators
- On-card switching regulators
- Positive to negative converter (Buck-Boost)

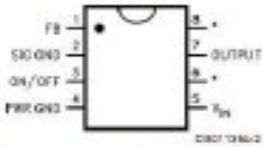
Typical Application (Fixed Output Voltage Versions)



Note: Pin numbers are for 8-pin DIP package.

Connection Diagrams

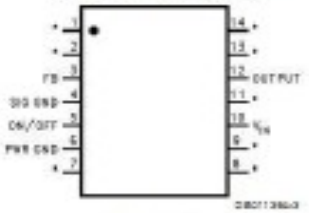
8-Lead DIP



*Nc internal connection, but should be soldered to PC board for best heat transfer.

Top View
Order Number LM2574-3.3HVN, LM2574HVN-5.0, LM2574HVN-12, LM2574HVN-15, LM2574HVN-ADJ, LM2574N-3.3, LM2574N-5.0, LM2574N-12, LM2574N-15 or LM2574N-ADJ
See NS Package Number N08A

14-Lead Wide Surface Mount (WM)



Top View
Order Number LM2574HVM-3.3, LM2574HVM-5.0, LM2574HVM-12, LM2574HVM-15, LM2574HVM-ADJ, LM2574M-3.3 LM2574M-5.0, LM2574M-12, LM2574M-15 or LM2574M-ADJ
See NS Package Number M14B

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Maximum Supply Voltage	
LM2574	45V
LM2574HV	63V
$\overline{ON}$ /OFF Pin Input Voltage	$-0.3V \le V \le +V_{IN}$
Output Voltage to Ground (Steady State)	−1V
Minimum ESD Rating (C = 100 pF, R = 1.5 kΩ)	2 kV
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	260°C
Maximum Junction Temperature	150°C
Power Dissipation	Internally Limited

Operating Ratings

Temperature Range	
LM2574/LM2574HV	$-40°C \le T_J \le +125°C$
Supply Voltage	
LM2574	40V
LM2574HV	60V

LM2574-3.3, LM2574HV-3.3 Electrical Characteristics

Specifications with standard type face are for T_J = 25°C, and those with **boldface type** apply over **full Operating Temperature Range.**

Symbol	Parameter	Conditions	LM2574-3.3 LM2574HV-3.3 Typ	LM2574-3.3 LM2574HV-3.3 Limit (Note 2)	Units (Limits)
SYSTEM PARAMETERS (Note 3) Test Circuit *Figure 2*					
V_{OUT}	Output Voltage	V_{IN} = 12V, I_{LOAD} = 100 mA	3.3	3.234 3.366	V V(Min) V(Max)
V_{OUT}	Output Voltage LM2574	$4.75V \le V_{IN} \le 40V$, $0.1A \le I_{LOAD} \le 0.5A$	3.3	3.168/**3.135** 3.432/**3.465**	V V(Min) V(Max)
V_{OUT}	Output Voltage LM2574HV	$4.75V \le V_{IN} \le 60V$, $0.1A \le I_{LOAD} \le 0.5A$	3.3	3.168/**3.135** 3.450/**3.482**	V V(Min) V(Max)
η	Efficiency	V_{IN} = 12V, I_{LOAD} = 0.5A	72		%

LM2574-5.0, LM2574HV-5.0 Electrical Characteristics

Specifications with standard type face are for T_J = 25°C, and those with **boldface type** apply over **full Operating Temperature Range.**

Symbol	Parameter	Conditions	LM2574-5.0 LM2574HV-5.0 Typ	LM2574-5.0 LM2574HV-5.0 Limit (Note 2)	Units (Limits)
SYSTEM PARAMETERS (Note 3) Test Circuit *Figure 2*					
V_{OUT}	Output Voltage	V_{IN} = 12V, I_{LOAD} = 100 mA	5	4.900 5.100	V V(Min) V(Max)
V_{OUT}	Output Voltage LM2574	$7V \le V_{IN} \le 40V$, $0.1A \le I_{LOAD} \le 0.5A$	5	4.800/**4.750** 5.200/**5.250**	V V(Min) V(Max)
V_{OUT}	Output Voltage LM2574HV	$7V \le V_{IN} \le 60V$, $0.1A \le I_{LOAD} \le 0.5A$	5	4.800/**4.750** 5.225/**5.275**	V V(Min) V(Max)
η	Efficiency	V_{IN} = 12V, I_{LOAD} = 0.5A	77		%

LM2574-12, LM2574HV-12 Electrical Characteristics

Specifications with standard type face are for T_J = 25°C, and those with **boldface type** apply over **full Operating Temperature Range.**

Symbol	Parameter	Conditions	LM2574-12 LM2574HV-12 Typ	LM2574-12 LM2574HV-12 Limit (Note 2)	Units (Limits)
SYSTEM PARAMETERS (Note 3) Test Circuit *Figure 2*					
V_{OUT}	Output Voltage	V_{IN} = 25V, I_{LOAD} = 100 mA	12	11.76 12.24	V V(Min) V(Max)
V_{OUT}	Output Voltage LM2574	$15V \le V_{IN} \le 40V$, $0.1A \le I_{LOAD} \le 0.5A$	12	11.52/**11.40** 12.48/**12.60**	V V(Min) V(Max)
V_{OUT}	Output Voltage LM2574HV	$15V \le V_{IN} \le 60V$, $0.1A \le I_{LOAD} \le 0.5A$	12	11.52/**11.40** 12.54/**12.66**	V V(Min) V(Max)
η	Efficiency	V_{IN} = 15V, I_{LOAD} = 0.5A	88		%

LM2574-15, LM2574HV-15 Electrical Characteristics

Specifications with standard type face are for T_J = 25°C, and those with **boldface type** apply over **full Operating Temperature Range**.

Symbol	Parameter	Conditions	LM2574-15 LM2574HV-15		Units (Limits)
			Typ	Limit (Note 2)	
SYSTEM PARAMETERS (Note 3) Test Circuit *Figure 2*					
V_{OUT}	Output Voltage	V_{IN} = 30V, I_{LOAD} = 100 mA	15		V
				14.70	V(Min)
				15.30	V(Max)
V_{OUT}	Output Voltage LM2574	18V ≤ V_{IN} ≤ 40V, 0.1A ≤ I_{LOAD} ≤ 0.5A	15		V
				14.40/**14.25**	V(Min)
				15.60/**15.75**	V(Max)
V_{OUT}	Output Voltage LM2574HV	18V ≤ V_{IN} ≤ 60V, 0.1A ≤ I_{LOAD} ≤ 0.5A	15		
				14.40/**14.25**	V(Min)
				15.68/**15.83**	V(Max)
η	Efficiency	V_{IN} = 18V, I_{LOAD} = 0.5A	88		%

All Output Voltage Versions Electrical Characteristics

Specifications with standard type face are for T_J = 25°C, and those with **boldface type** apply over **full Operating Temperature Range**. Unless otherwise specified, V_{IN} = 12V for the 3.3V, 5V, and Adjustable version, V_{IN} = 25V for the 12V version, and V_{IN} = 30V for the 15V version. I_{LOAD} = 100 mA.

Symbol	Parameter	Conditions	LM2574-XX LM2574HV-XX		Units (Limits)
			Typ	Limit (Note 2)	
DEVICE PARAMETERS					
I_b	Feedback Bias Current	Adjustable Version Only, V_{OUT} = 5V	50	100/**500**	nA
f_O	Oscillator Frequency	(see Note 10)	52		kHz
				47/**42**	kHz(Min)
				58/**63**	kHz(Max)
V_{SAT}	Saturation Voltage	I_{OUT} = 0.5A (Note 4)	0.9		V
				1.2/**1.4**	V(max)
DC	Max Duty Cycle (ON)	(Note 5)	98		%
				93	%(Min)
I_{CL}	Current Limit	Peak Current, (Notes 4, 10)	1.0		A
				0.7/**0.65**	A(Min)
				1.6/**1.8**	A(Max)
I_L	Output Leakage Current	(Notes 6, 7) Output = 0V		2	mA(Max)
		Output = −1V	7.5		mA
		Output = −1V		30	mA(Max)
I_Q	Quiescent Current	(Note 6)	5		mA
				10	mA(Max)
I_{STBY}	Standby Quiescent Current	$\overline{ON}$ /OFF Pin= 5V (OFF)	50		µA
				200	µA(Max)
θ_{JA}	Thermal Resistance	N Package, Junction to Ambient (Note 8)	92		°C/W
θ_{JA}		N Package, Junction to Ambient (Note 9)	72		
θ_{JA}		M Package, Junction to Ambient (Note 8)	102		
θ_{JA}		M Package, Junction to Ambient (Note 9)	78		

All Output Voltage Versions
Electrical Characteristics (Continued)

Specifications with standard type face are for T_J = 25°C, and those with **boldface type** apply over **full Operating Temperature Range**. Unless otherwise specified, V_{IN} = 12V for the 3.3V, 5V, and Adjustable version, V_{IN} = 25V for the 12V version, and V_{IN} = 30V for the 15V version. I_{LOAD} = 100 mA.

Symbol	Parameter	Conditions	LM2574-XX LM2574HV-XX		Units (Limits)
			Typ	Limit (Note 2)	
ON /OFF CONTROL Test Circuit *Figure 2*					
	ON /OFF Pin Logic Input Level	V_{OUT} = 0V	1.4	**2.2**/2.4	V(Min)
		V_{OUT} = Nominal Output Voltage	1.2	1.0/**0.8**	V(Max)
I_H	ON /OFF Pin Input Current	ON /OFF Pin = 5V (OFF)	12	30	µA µA(Max)
I_{IL}		ON /OFF Pin = 0V (ON)	0	10	µA µA(Max)

LM2574-10

Note 1: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. Operating Ratings indicate conditions for which the device is intended to be functional, but do not guarantee specific performance limits. For guaranteed specifications and test conditions, see the Electrical Characteristics.

Note 2: All limits guaranteed at room temperature (Standard type face) and at temperature extremes (bold type face). All room temperature limits are 100% production tested. All limits at temperature extremes are guaranteed via correlation using standard Statistical Quality Control (SQC) methods. All limits are used to calculate Average Outgoing Quality Level.

Note 3: External components such as the catch diode, inductor, input and output capacitors can affect switching regulator system performance. When the LM2574 is used as shown in the *Figure 2* test circuit, system performance will be as shown in system parameters section of Electrical Characteristics.

Note 4: Output pin sourcing current. No diode, inductor or capacitor connected to output pin.

Note 5: Feedback pin removed from output and connected to 0V.

Note 6: Feedback pin removed from output and connected to +12V for the Adjustable, 3.3V, and 5V versions, and +25V for the 12V and 15V versions, to force the output transistor OFF.

Note 7: V_{IN} = 40V (60V for high voltage version).

Note 8: Junction to ambient thermal resistance with approximately 1 square inch of printed circuit board copper surrounding the leads. Additional copper area will lower thermal resistance further. See application hints in this data sheet and the thermal model in Switcher Made Simple software.

Note 9: Junction to ambient thermal resistance with approximately 4 square inches of 1 oz. (0.0014 in. thick) printed circuit board copper surrounding the leads. Additional copper area will lower thermal resistance further. (See Note 8.)

Note 10: The oscillator frequency reduces to approximately 19 kHz in the event of an output short or an overload which causes the regulated output voltage to drop approximately 40% from the nominal output voltage. This self protection feature lowers the average power dissipation of the IC by lowering the minimum duty cycle from 5% down to approximately 2%.