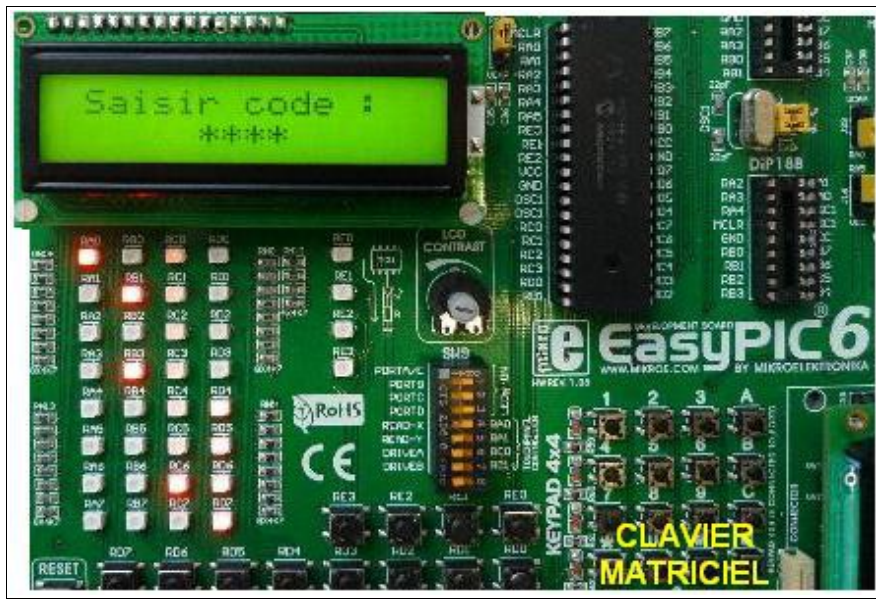




*Cet exercice est un travail de synthèse sur toutes les compétences que vous avez acquises depuis le début de cet enseignement. Vous allez devoir manipuler des ports d'entrée/sortie, configurer le timer0 et l'interruption associée puis utiliser des APIs de votre librairie assurant le pilotage l'afficheur LCD. Votre travail consistera à implémenter un DIGICODE (cf. figure 1), semblable à celui rencontré devant chez vous et à gérer le clavier matriciel associé.*

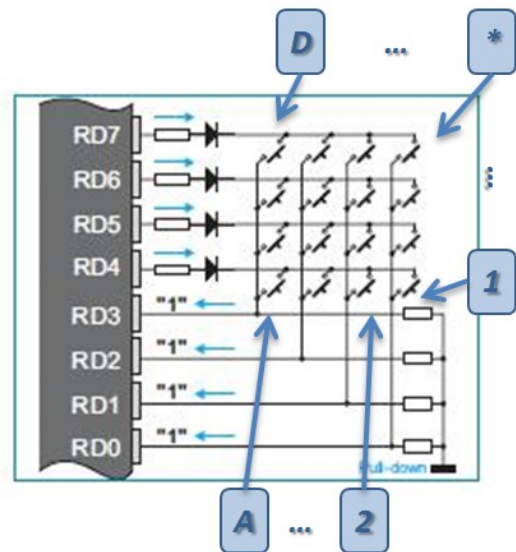


**figure 1** : exemple de réalisation du digicode

## 1. Qu'est-ce qu'un clavier matriciel ?

**Le rôle d'un clavier matriciel est de permettre une économie non négligeable de broches**, pour mieux comprendre prenons un exemple. Supposons que nous souhaitons utiliser un clavier à 12 touches (0,1, 2 ... 8,9,#,\* ), une solution qui fonctionnerait très bien pourrait-être d'utiliser 12 boutons poussoirs et donc **12 broches**. Seulement si nous disposons d'un MCU de 18 broches, nous allons rapidement nous retrouver à court de broches. Rappelez-vous qu'il faut également câbler l'afficheur LCD (6 broches).

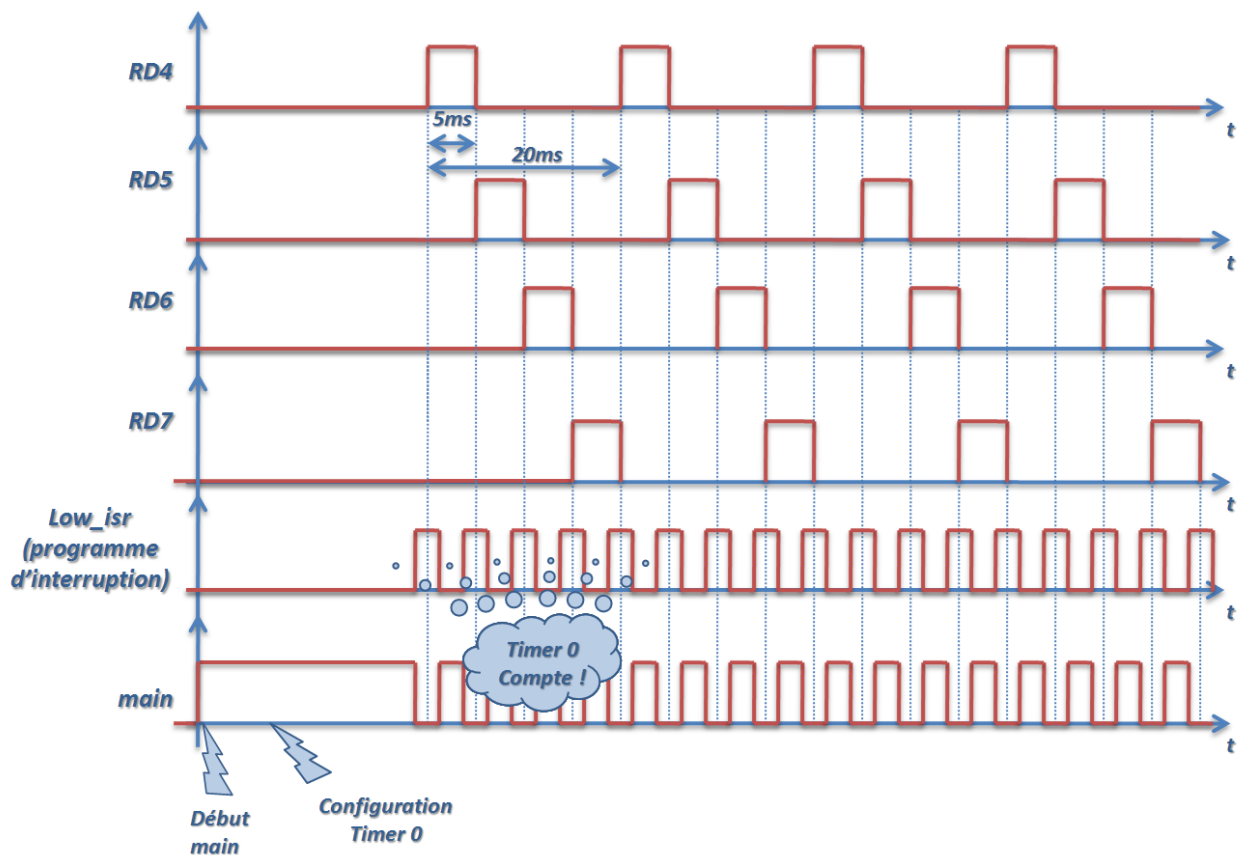
La meilleure solution est d'utiliser un clavier matriciel (cf. **figure 2**). L'idée étant d'utiliser une broche par ligne et par colonne. **Nous allons devoir gérer une matrice 4x3 (4 lignes et 3 colonnes) et nous n'avons donc plus besoin que de 7 broches !**



**figure 2 :** câblage du clavier matriciel sur la maquette EASYPIC6



En observant la **figure 2**, nous constatons que les lignes du clavier sont connectées aux sorties RD4 à RD7 et les colonnes aux entrées RD0 à RD2 (nous ne nous intéressons pas à la colonne A-B-C-D, connectée à RD3).

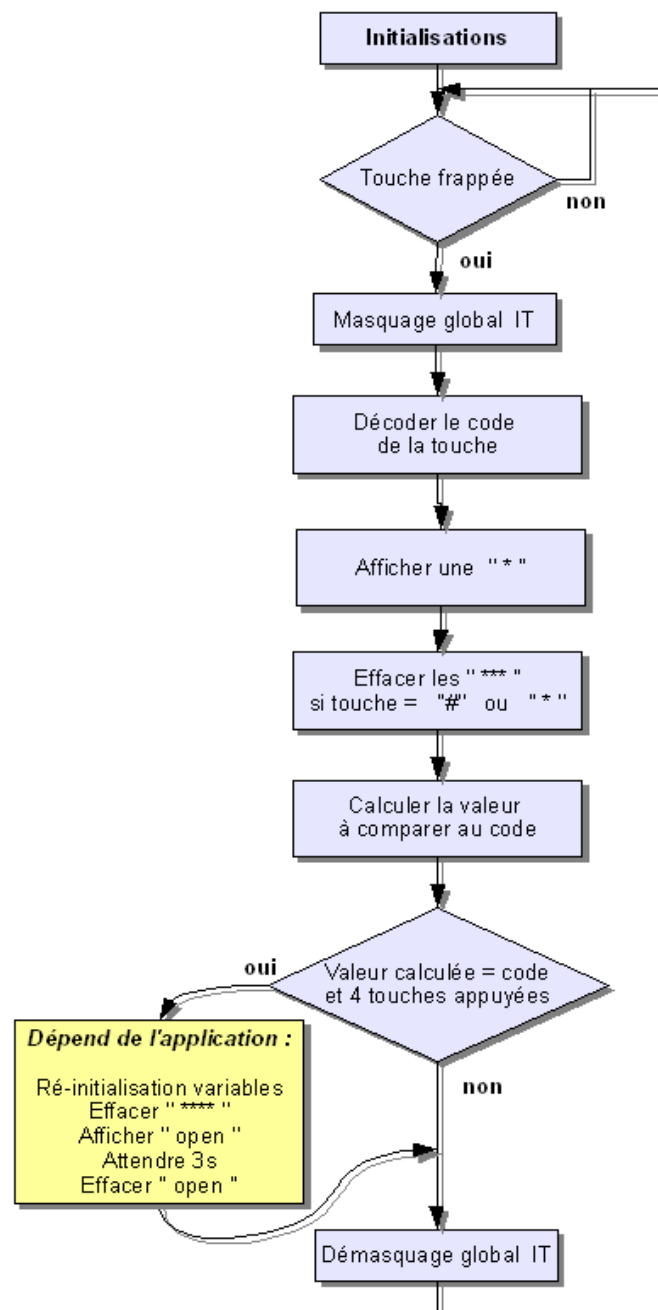


**figure 3 :** signaux à appliquer aux lignes du clavier matriciel

Vous allez devoir configurer le Timer0 de façon à réveiller le programme d'interruption associé toutes les 5ms. Vous devrez, une fois dans le programme d'IT, mettre **une seule des quatre broches** de sortie (RD4 à RD7) à "1" comme présenté en **figure 3**. Nous allons donc effectuer un balayage des lignes. Le programme principal quant à lui scrutera les colonnes en testant la valeur sur les broches RD0-RD2 jusqu'à ce que l'une d'entre elles soit différente de "0", signifiant que l'utilisateur vient d'appuyer sur une touche. On ira alors lire la valeur sur le port D (RD0-RD7) puis on décodera le caractère de la touche correspondante.

## 2. Digicode :

Vous êtes maintenant prêts à attaquer le développement du digicode dans son ensemble. Cependant, ce travail est relativement long. Pour vous permettre de gagner un peu de temps, nous vous donnons l'organigramme de la boucle principale du main().





**Bien, vous êtes maintenant prêt à attaquer le TP ! ... vous pouvez d'ailleurs dès à présent commencer à compléter les fichiers sources présents juste après ces questions.**

**Cependant, dans un soucis pédagogique, il vous est demandé un petit travail préparatoire. Il sera vérifié en début de séance et vous permettra d'anticiper une perte de temps non négligeable dans l'avancement des TP (nb ★ = difficulté).**

1. (★) Quel est l'avantage d'un clavier matriciel ?
2. (★) Proposez une configuration du Timer0 afin de générer un demande d'IT toutes les 5ms ?
3. (★) Proposez une configuration (démasquage) pour l'IT du Timer0 ? Nous souhaitons travailler en priority mode et utiliser le vecteur de priorité basse.
4. (★★) Complétez la table de décodage ci-dessous afin de retrouver le code de la touche frappée à partir du signal sur les broches RD7 à RD0. Par exemple **la touche "1" est reliée aux broches RD0 et RD4, la touche "2" à RD1 et RD4, la touche "3" à RD2 et RD4 ... la touche "#" à RD2 et RD7.**

Valeur sur le port D de RD7à RD0	Touche correspondante
0x11	1
	2
	3
	4
	5
	6
	7
	8
	9
	*
	0
	#



## Listing du programme *ex4.c*

```

/*****
@file : ex4.c
@brief : gestion d'un clavier matriciel – PROJET DIGICODE
@author :
last modification :
*****/

/** Configuration bits */
#pragma config PLLDIV=2, CPUDIV=OSC1_PLL2, FOSC=HSPLL_HS, BOR=OFF, WDT=OFF, MCLRE=ON

/** Includes files */
#include <p18f4550.h>
#include <delays.h>
#include "LCDUser.h"

/** Définition des Macros */
#define Delay_200ms()      Delay10KTCYx(240)    /** 240.10000.TCY ~ 200ms */

#define Delay_1s()          \
                          Delay10KTCYx(240); \
                          Delay10KTCYx(240); \
                          Delay10KTCYx(240); \
                          Delay10KTCYx(240); \
                          Delay10KTCYx(240)

#define CODE_VALUE          1492
#define TMR0LVALUE         /** à compléter ! */
#define TMR0HVALUE         /** à compléter ! */

/** Déclaration des variables globales */
char * accueilMess = "Saisir code :";
char * effaceMess = " ";
char * openMess = "Open";
int CODE = CODE_VALUE;

#pragma code
/*****
/** ISR : routine d'interruption de priorité basse */
*****/
#pragma interruptlow low_isr
void low_isr(void)
{static unsigned int Decode = 1;

    if(INTCONbits.TMR0IF)
    {
        /** Clear Timer0 overflow flag */

        /** à compléter ! */

        /** pré-chargement de TMR0L + chargement automatique de TMR0H */

        /** à compléter ! */

        /** BALAYAGE : Mise à "1" de l'une des broches RD4-RD7 */

        /** à compléter ! */
    }
}

```

```
/** Configuration du vecteur d'interruption de priorité basse **/  
#pragma code low_vector = 0x18  
void interrupt_at_low_vector(void)  
{  
    _asm goto low_isr _endasm  
}  
  
#pragma code  
/*****/  
/** FONCTION : initialisation du timer0 et de l'IT associée **/  
/*****/  
void Timer_Config(void)  
{  
    /** T0CON : Timer0=OFF, mode=16bits, synchro=Tcy, prescaler=ON, prescale value=1:256 **/  
        /** à compléter ! **/  
  
    /** TMR0 : Initialisation des registres de pré-chargement **/  
        /** à compléter ! **/  
  
    /** Gestion des IT : priority mode, vecteur d'IT de priorité basse, démasquage IT TMR0, TMR0IF=0 **/  
        /** à compléter ! **/  
  
    /** Démarrage du Timer0 **/  
        /** à compléter ! **/  
}  
  
/*****/  
/** FONCTION : initialisation des ports B et D **/  
/*****/  
void Port_Config(void)  
{  
    /** LCD config : Broches RB0 à RB5 en sortie **/  
        /** à compléter ! **/  
  
    /** KEYBOARD config : Broches RD4 à RD7 en sortie et RD0 à RD2 en entrée **/  
        /** à compléter ! **/  
}  
  
/*****/  
*** PROGRAMME PRINCIPAL ***  
/*****/  
void main() {  
    unsigned char    TouchNum , CodeCount = 0, latchPORTD;  
    unsigned int     CodeCompare = 0;  
  
    /** Appel fonction de configuration des port B et D **/  
        /** à compléter ! **/  
}
```



```

    /*** Appel fonction de configuration du Timer0 et des IT ***/
    /*** à compléter ! ***/

    /*** Initialisation LCD et affichage message d'accueil ***/
    /*** à compléter ! ***/

    /*** Démasquage global des IT ***/
    /*** à compléter ! ***/
while(1){

    /*** Test si l'utilisateur a appuyé sur une touche ***/
    if( ???? ){
        /*** Masquage global des IT ***/

        /*** à compléter ! ***/

        /*** Sauvegarde de la valeur sur le PORT D ***/
        latchPORTD = PORTD;

        /*** n°1 : Détection de la touche frappée ***/
        switch(latchPORTD){
            case 0x11      :      TouchNum = 0x1;      break;
            case ?????    :      TouchNum = 0x2;      break;
            case ?????    :      TouchNum = 0x3;      break;
            case ?????    :      TouchNum = 0x4;      break;
            case ?????    :      TouchNum = 0x5;      break;
            case ?????    :      TouchNum = 0x6;      break;
            case ?????    :      TouchNum = 0x7;      break;
            case ?????    :      TouchNum = 0x8;      break;
            case ?????    :      TouchNum = 0x9;      break;
            case ?????    :      TouchNum = 0xF;      break;
            case ?????    :      TouchNum = 0x0;      break;
            case ?????    :      TouchNum = 0xF;      break;
            default        :      TouchNum = 0xF;

        }

        /*** n°2 : Affichage des caractères "*" et gestion du compteur de touches frappées ***/

        /*** à compléter ! ***/

        /*** n°3 : Effacement des "****" si touche "#" ou "*" frappée ***/

        /*** à compléter ! ***/

        /*** n°4 : Calcul de la valeur à comparer au CODE ***/

        /*** à compléter ! ***/

        /*** n°5 : Si code calculer = CODE et si on a appuyé sur 4 touches alors ... ***/

        /*** à compléter ! ***/

        /*** tant que l'on reste appuyé sur la touche, on ne relance pas le balayage des lignes ***/
        while(PORTD == latchPORTD);

        /*** Démasquage global des IT ***/
        /*** à compléter ! ***/
    }
}

```



**Notez vos remarques :**



## **1. Travail pratique (ex4 : partie 1) :** Générer le signal de balayage

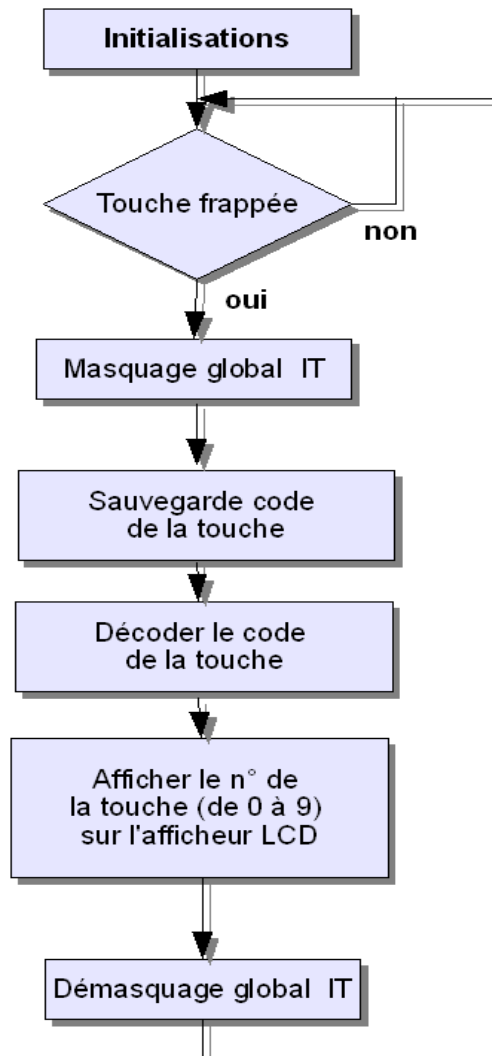
Durant cette première partie de l'exercice votre cahier des charges est de générer le signal de balayage des lignes. Votre travail va se découper en trois parties :

1. Dans un premier temps il vous faut configurer les ports B et D. Les broches n°0 à 5 du port B sont reliées à l'afficheur et les broches n°0 à 7 du port D sont reliées au clavier matriciel.
2. Dans un second temps, il vous faudra configurer le Timer0 et démasquer l'interruption associée. Nous utiliserons le vecteur d'IT de priorité basse.
3. Pour conclure, vous devrez écrire le programme d'interruption de façon à respecter le chronogramme donné en **figure 3**. Le programme d'IT est appelé une fois toutes les 5ms si la configuration du Timer0 et le démasquage des IT ont été bien effectués.

- ➡  **Copiez et complétez le listing du programme ex4.c.**
- ➡  **Créer un projet ex4 dans votre répertoire de travail (cf. ANNEXE 1)**
- ➡ Configurez les port B et D
- ➡ Configurez le Timer0 et démasquez l'interruption associée
- ➡ Complétez le programme d'interruption de façon à respecter le chronogramme donné en **figure 3**
- ➡ Initialisez l'afficheur et affichez un message d'accueil (dans le main())
- ➡ Vérifiez et relevez expérimentalement en utilisant l'oscilloscope ou l'analyseur logique que les délais donnés en **figure 3** soient bien respectés.

## **2.Travail pratique (ex4 : partie 2) :** Affichage du n° de la touche sur l'afficheur LCD

En vous aidant de l'organigramme du main donné ci-dessous, réalisez un programme qui récupère le numéro de la touche appuyée (entre 0 et 9) puis qui l'affiche sur l'afficheur LCD. Utilisez par exemple la librairie développée durant l'exercice précédent.



## **3.Travail pratique (ex4 : partie 3) :** digicode

Il ne vous reste plus qu'à implémenter le digicode. A ce niveau du développement, il ne s'agit plus que d'un simple problème de C. Aidez-vous de l'organigramme donné dans la présentation.



*A vous de jouer maintenant !*