TP 1 Prise en main

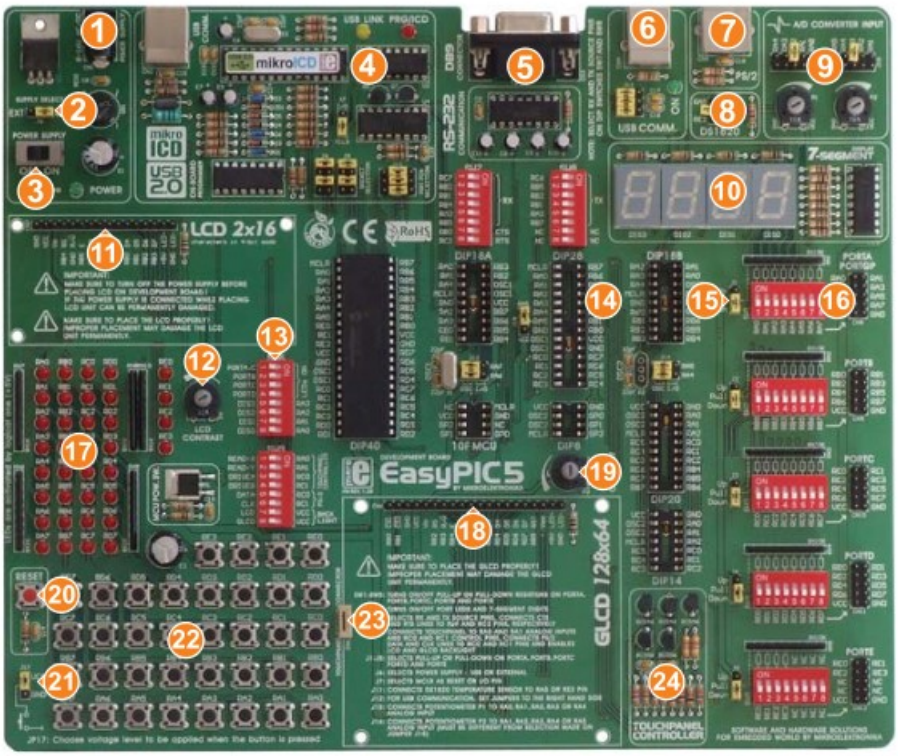
I. Préambule

– une documentation est disponible dans C:\doc_mikroc for PIC\
– sur le site du constructeur(http://www.mikroe.com/), vous pourrez consulter divers éléments de la documentation et télécharger une version de démonstration de l'IDE et du compilateur.

Sur votre poste de travail, vous devez obligatoirement vous connecter en utilisant comme login "ge1" et sans mot de passe.

II.Observation de la maquette

Voici une maquette comportant des numéros



1. Manipulation

Identifier les éléments suivants sur cette reproduction (donner les numéros) :

- switch d'alimentation
- programmateur USB (mikroICD)
- micro-contrôleur PIC,
- bouton de reset
-diodes de contrôle des PORTs A à E,
- afficheurs sept segments multiplexés

III.Création d'un premier projet

Vous disposez d'un petit fascicule en anglais intitulé *Creating the first project in mikroC for PIC*. Ce texte vous indique la marche à suivre pour créer et compiler un projet.

1. Manipulation

Suivre les instructions (jusqu'à l'exécution - *run*) avec les recommandations suivantes :
- nom du projet : tp1a
- Chemin du projet d:\rep_perso\tp1. Ce dossier doit être créé en remplaçant rep_perso par un nom qui vous est propre
- Le type (*device*) est à lire sur la puce elle-même. En principe 16F887.
- Choisir les fusibles par défaut.
- Le programme suivant est à saisir :

```
1         void main (){
2         //ANSEL = ANSELH = 0; // Toutes le ports E/S sont configurés comme numériques
3           PORTC = 0;
4           TRISC = 0;
5           while (1) {
6             PORTC = ~PORTC; // toggle PORTC
7             Delay_ms(1000);
8           }
9         }
```

Si vous voulez un fonctionnement correct de Delay_ms, il faut configurer aussi la fréquence du quartz (8 MHz) correctement.

2. Manipulation

Aller dans le dossier de votre projet et examiner les fichiers .hex, .mcl, .asm, .lst. Que contiennent ces fichiers, quelle est leur utilité ?

Par quelles instructions assembleur sont traduites les lignes suivantes ?

- PORTC = 0;

- TRISC = 0;

IV.Exécution pas-à-pas, déboguage

Créer un nouveau projet toujours dans votre dossier tp1 nommé tp1b. Le programme à saisir est le suivant :

```
1         void main (){
2         //ANSEL = ANSELH = 0; // Toutes le ports E/S sont configurés comme numériques
3         int k;
4           PORTC = 0;
5           TRISC = 0;
6           for (k=0;k<256;k++) {
7             PORTC = k; // toggle PORTC
8           }
9         }
```

Ce programme est tellement rapide qu'il y a peu de chance que vous voyez ce qu'il fait. Pour le voir on va utiliser un débogueur qui permet de l'exécuter pas à pas.

3. Manipulation

Pour utiliser le débogueur reportez-vous à la photocopie ci-après.

Name	Description	Function key
Debug	Starts Debugger.	[F9]
Run/Pause debugger	Runs or pauses program execution and debugging.	[F6]
Toggle Breakpoints	During debugging, program is executed until a breakpoint is reached. At that point, the *Toggle Breakpoints* option sets new breakpoints or removes those already set at the current cursor position. To view all the breakpoints, select *Run >View Breakpoints* from the drop-down menu. Double click on an item from the list locates the breakpoint.	[F5]
Run to cursor	Program is executed until the cursor position is reached.	[F4]
Step Into	Executes the current C/Pascal/Basic (single- or multi-cycle) program line, then halts. If the program line is a routine call, steps into the routine and halts at the first instruction within it.	[F7]
Step Over	Executes the current C/Pascal/Basic (single- or multi-cycle) program line, then halts. If the program line is a routine call, enters the routine and halts at the first instruction following the call.	[F8]
Flush RAM	Flushes current PIC microcontroller RAM. All RAM memory values will be changed according to values in the *Watch* window.	-
Stop Debugger	Stops Debugger.	[Ctrl+F2]
Step Out	Executes all remaining program lines within the subroutine. It halts immediately upon exit from the subroutine.	[Ctrl+F8]
Disassembly View	Instead of program written in high-level program language (*Basic* in this very case), the assembly version of the same program will appear. Each program line written in Basic is divided in assembly instructions executed by the microcontroller.	[Alt+D]

- Modifier les options du projet pour activer le débogueur ICD (dans *project setup Window*, sur la gauche) :

** cocher mikroICD Debug sous Build type

** vérifier que mikroICD Debugger est activé sous Debugger.

- Compiler le projet (*Build* - CTRL+F9)

- Programmer la puce (*Program* -F11)

- Lancer le debogueur (*Start Debugger* - F9)

- Suivre en pas à pas l'exécution du programme (*Step Info* par exemple). Vérifier l'allumage correct des diodes du PORTC.

Prenez l'habitude à partir de maintenant de réaliser vos programme en version *runtime*, c'est à dire sans débogueur. Celui-ci ne sera utilisé que si nécessaire, c'est à dire pour retrouver une erreur subtile dans un programme.

TP2 Des leds, des leds

I. Rappels

On rappelle qu'en C le OU booléen se fait par ||, le ET booléen par &&. Nous aurons besoin du OU bit à bit | et du ET bit à bit &. Soit le contenu d'un registre B sur 8 bits,

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
1	1	1	0	0	0	1	1

1. Préparation

- Vous désirez mettre le bit b_2 à 1 sans changer les autres bits, comment faites-vous ?

- Vous désirez mettre le bit b_6 à 0 sans changer les autres bits, comment faites-vous ?

II.Exemple

On vous donne un programme C qui fait clignoter une led (poids faible) sur le port C.

```
1          void main (){
2            ANSEL = ANSELH = 0; // Toutes le ports E/S sont configurés comme numériques
3            TRISC = 0; // tous les bits en sortie pour PORTC
4            PORTC = 0;
5            while(1) {
6              PORTC = 0x01;
7              Delay_ms(1000);
8              PORTC = 0x00;
9              Delay_ms(1000);
10           }
11         }
```

1. Exercice 2.1

Écrire ce programme, le charger et l'exécuter. Modifiez-le pour faire clignoter RC1.

III.Exercices

1. Exercice 2.2

Écrire un chenillard simple : une led se déplaçant sur le PORTC (de haut en bas) et en utilisant le même type de temporisation que dans le programme exemple. Utilisez l'un des opératuers >> ou <<.

2. Exercice 2.3

Écrire un chenillard double : un chenillard de haut en bas et simultanément de bas en haut qui se croisent. Utilis

3. Exercice 2.4

Écrire un chenillard à entassement. Une led se déplaçant et s'accumulant vers le bas.

TP3 Ports en entrée/sortie

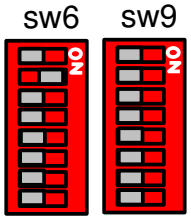
Les ports A, B, C, D et E sont des ports d'entrée/sortie dont chaque bit peut être utilisé soit en entrée soit en sortie, de façon indépendante. Ainsi chaque bit possède un satellite : TRISA, TRISB, TRISC, TRISD et TRISE qui permet de déterminer le sens de chaque bit (0: *Output*, 1 : *input*). Par exemple :

```
1          void main () {
2            ANSEL = ANSELH = 0; // Toutes le ports E/S sont configurés comme numériques
3            TRISCA = 0b00000100; // tous les bits en sortie pour PORTA sauf RA2
4            PORTA = 0;
5          }
```

est un programme qui positionne des entrées et sorties sur le PORTA.

1. Exercice 3.1

Écrire un programme qui positionne en entrée les quatre bits de poids faible du PORTB et en sortie les quatre autres. Le programme doit alors en permanence copier les 4 bits de poids faibles vers les quatre bits de poids fort.



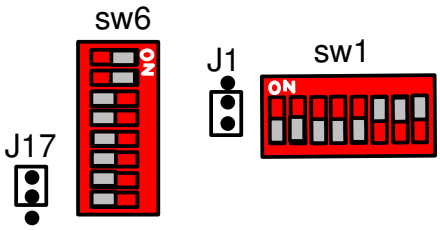
2. Exercice 3.2

Écrire un programme qui comporte les éléments suivants :

- un compteur binaire sur le PORTB (256 états). Placer une temporisation de 100 ms entre chaque état.

- L'appui sur RA2 (bit b_2 du PORTA) doit remettre à zéro le compteur.

- Les accès individuels aux bits seront effectués à l'aide de masques.



- Modifier ensuite le programme pour que le comptage n'ait lieu que lorsque RA1 est à 0.

3. Exercice 3.3 Changement d'état d'une diode

Réaliser un programme avec le cahier des charges suivant :

- En début de programme le bit RB0 (bit b_0 du PORTB) doit être allumé.

- Ensuite un front montant sur RA0 (bit b_0 du PORTA) provoquera un changement d'état de la diode.

- Les accès aux bits seront effectués par des accès directs (en écrivant PORTB.F2 pour accéder au bit b_2 du PORTB). Puis dans un second temps avec des masques.

4. Exercice 3.4 Compteur

Reprendre le compteur de l'exercice 3.2 avec les modifications suivantes :

- un front descendant sur RA2 (bit b_2 du PORTA) provoquera le RAZ du compteur

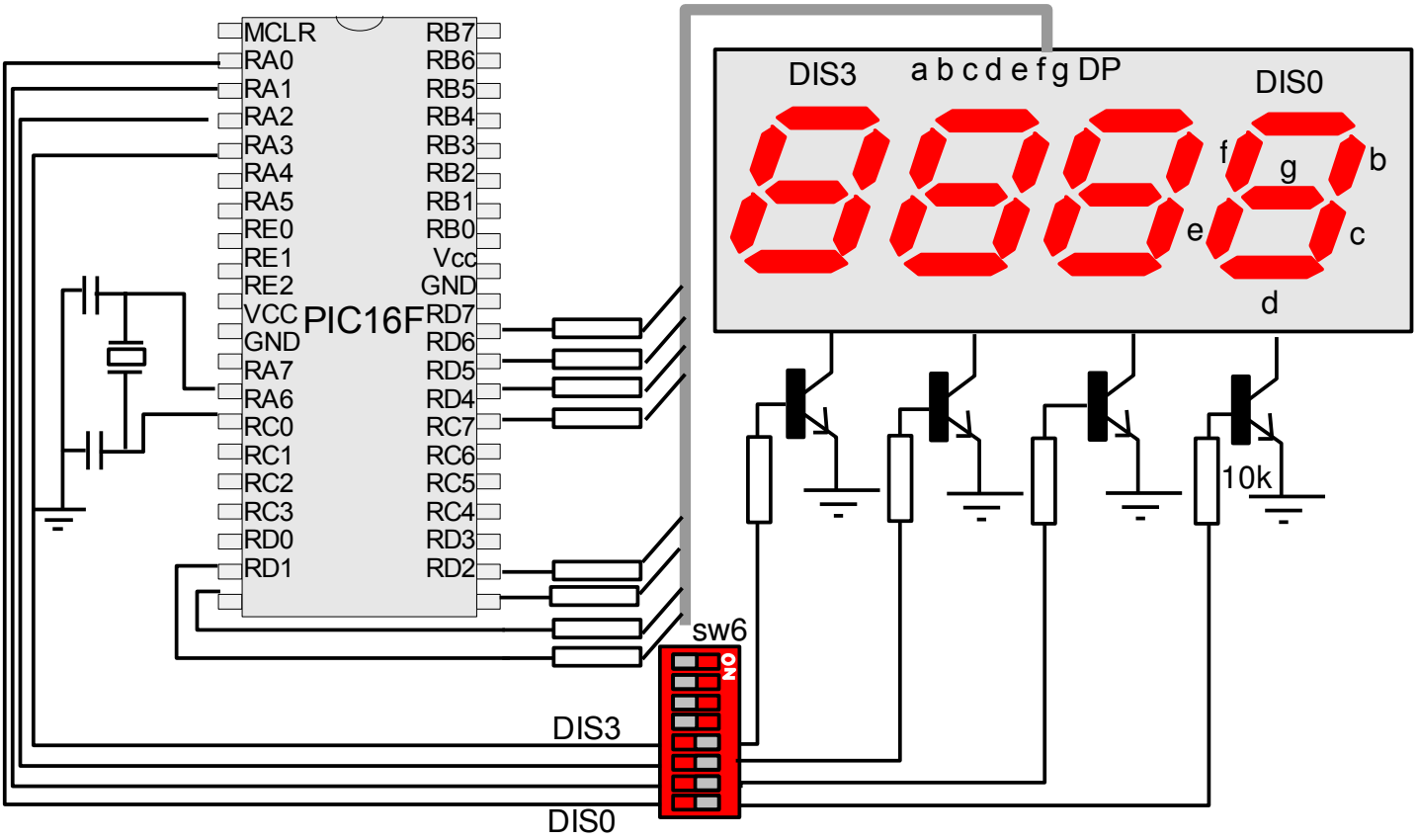
- un front descendant sur RA1 (bit b_1 du PORTA) incrémentera le compteur.

Que pensez-vous de l'attende de plusieurs front ?

TP 4 Multiplexage - afficheurs sept segments

I. Le matériel

On vous donne le schéma de principe de la maquette de TP avec la figure ci-dessous. Ce qui n'est pas apparent est que le segment "a" est poids faible (RD0) et "g" est relié en RD6 et que les afficheurs sont cathode commune c'est à dire qu'ils s'allument avec un un logique.



1. Quel port permet de sélectionner l'afficheur actif ?
2. Quelles valeurs faut-il envoyer pour sélectionner l'afficheur DIS0 ? L'afficheur DIS1 ? L'afficheur DIS2 ? L'afficheur DIS3 ?
3. Quel port permet d'activer les segments ?
4. Est-il possible d'activer deux afficheurs en même temps ?
5. Est-il possible d'afficher deux symboles différents en même temps ?
6. Quelle est l'utilité du multiplexage ?

II.Logiciel - test

1. Exercice 4.1

Voici un premier projet exploitant un afficheur.

```
1        unsigned short mask( unsigned short num);
```

```
2         void main (){
3         //ANSEL = ANSELH = 0; // Toutes le ports E/S sont configurés comme numériques
4           unsigned short i;
5           INTCON = 0; // Disable GIE , PEIE , INTE , RBIE , TOIE
6           TRISA = 0;
7           PORTA = 0;
8           TRISD = 0;
9           PORTD = 0;
10          while (1) {
11            for (i = 0; i <= 9u; i++){
12              PORTA = 0; // Turn off all 7seg displays
13              PORTD = mask(i); // bring appropriate value to PORTD
14              PORTA = 1; // turn on appropriate 7seg. display
15              Delay_ms (1000);
16            }
17          }
18        }
19
20        unsigned short mask( unsigned short num) {
21          switch (num) {
22          case 0 : return 0x3F;
23          case 1 : return 0x06;
24          case 2 : return 0x5B;
25          case 3 : return 0x4F;
26          case 4 : return 0x66;
27          case 5 : return 0x6D;
28          case 6 : return 0x7D;
29          case 7 : return 0x07;
30          case 8 : return 0x7F;
31          case 9 : return 0x6F;
32          }
33        }
```

Compiler et exécuter le programme. Que fait ce programme ?
Le modifier pour afficher le compteur sur DIS1, puis sur DIS2, puis sur DIS3.

III.Logiciel - multiplexage

L'objectif est d'afficher un compteur sur DIS1 et DIS0 - donc de 00 à 99. Il sera indispensable de basculer entre les deux afficheurs selon l'algorithme suivant :

```
Faire plusieurs fois:
  - activer DIS0
  - afficher le chiffre de poids faible
  - tempo
  - activer DIS1
  - afficher le chiffre de poids fort
  - tempo
```

2. Exercice 4.2

1. Écrire et tester un programme qui affiche un compteur de 00 à 99 sur les deux afficheurs - en s'appuyant sur l'algorithme donné. On utilisera les opérateurs modulo (%) et division (/).
2. Modifier les valeurs des temporisation - la méthode est-elle robuste ? simple ?

3. Exercice 4.3

Les opérateurs modulo et division sont très couteux sur un PIC 16F. On vous demande donc de les éviter en gérant une incrémentation BCD. Astuce à trouver.

IV.Affichage de 4 digits

1. Exercice 4.4

Réaliser un compteur et son affichage sur 4 digits en vous basant sur un algorithme similaire à celui de la section 3.

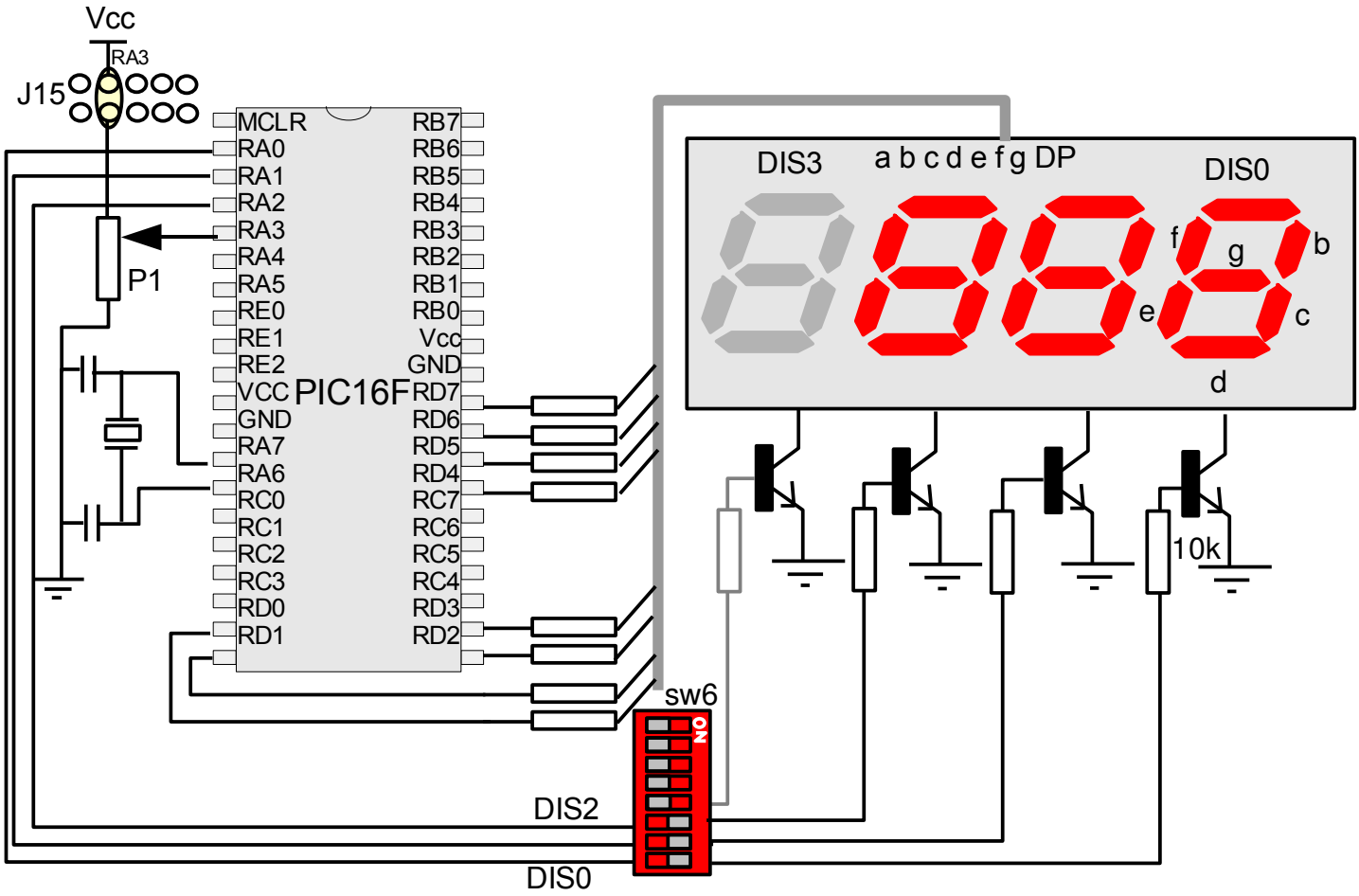
V.Affichage de la valeur d'un convertisseur analogique numérique

1. Exercice 4.5

On désire afficher la valeur en provenance d'un convertisseur analogique numérique sur trois digits. La valeur lue est sur 10 bits et nécessite donc en principe quatre digits. Mais pour des raisons électriques, il nous est impossible d'utiliser l'afficheur de poids le plus fort. La figure ci-dessous explique pourquoi. Un programme d'exemple se trouve parmi les démonstrations fournies avec la carte.

```
1         unsigned int temp_res;
2
3         void main() {
4           ANSEL  = 0x04;              // Configure AN2 pin as analog
5           ANSELH = 0;                 // Configure other AN pins as digital I/O
6           C1ON_bit = 0;               // Disable comparators
7           C2ON_bit = 0;
8
9           TRISA  = 0xFF;              // PORTA is input
10          TRISC  = 0;                 // PORTC is output
11          TRISD  = 0;                 // PORTD is output
12
13          do {
14            temp_res = ADC_Read(2);   // Get 10-bit results of AD conversion
15            PORTD = temp_res;         // Send lower 8 bits to PORTD
16            PORTC = temp_res >> 8;    // Send 2 most significant bits to RC1, RC0
17          } while(1);
18        }
```

Réaliser un programme qui lit le convertisseur et affiche le résultat sur trois digits.
- adapter pour que votre programme lise RA3 (contre RA2 dans le programme d'exemple)
- repérer le connecteur J15 sur votre carte et brancher le comme ci-dessous.
- repérer les interrupteurs sw6 et éteindre l'afficheur DIS3 comme indiqué ci-dessous.
- adapter votre aéfficheur 4 digits sur 3 digits. Un bon programme devra gérer le cas du digit des centaines dépassant neuf, ce qui peut arriver.
- après utilisation remettre J15 comme à votre arrivée.


Vcc
RA3
J15
P1
MCLR
RA0
RA1
RA2
RA3
RA4
RA5
RE0
RE1
RE2
VCC
GND
RA7
RA6
RC0
RC1
RC2
RC3
RD0
RD1
PIC16F
RB7
RB6
RB5
RB4
RB3
RB2
RB1
RB0
Vcc
GND
RD7
RD6
RD5
RD4
RC7
RC6
RC5
RC4
RD3
RD2
DIS3
a b c d e f g DP
DIS0
f
g
b
e
c
d
10k
sw6
DIS2
DIS0

TP5 Afficheur GLCD - capteur de température DS1820 one-wire

I. Affichage sur le GLCD

La résolution de l'écran est de 128 pixels horizontaux et 64 pixels verticaux. Elle sera notée 128x64 par la suite.

Les coordonnées suivant l'axe x vont donc de 0 à 127 et de 0 à 63 pour y.

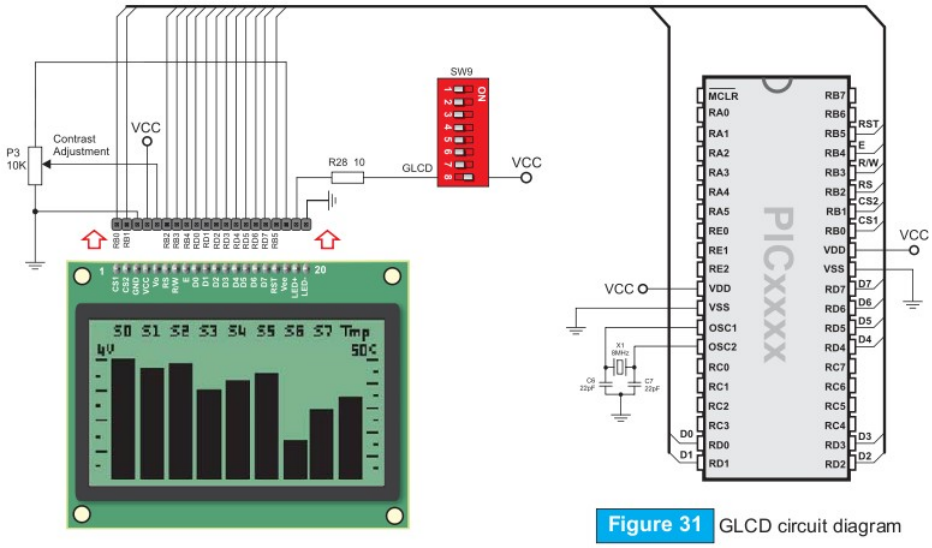


Figure 31 GLCD circuit diagram

Tester le programme suivant :

```
1         void my_glcd_init (){
2           ANSEL = ANSELH = 0; // AN pins as digital
3           Glcd_Init (&PORTB , 0, 1, 2, 3, 5, 4, &PORTD);
4           Glcd_Set_Font(FontSystem5x8, 5, 8, 0x20); // Sélection police de caractères
5           Glcd_Fill (0 x00 );
6         }
7         void main (){
8           my_glcd_init ();
9           Glcd_Write_Text (" Hello world !", 0, 0, 1);
10        }
```

1. Écrire ”Hello world!” en noir sur fond blanc. Pour cela consulter la documentation des fonctions Glcd_Fill et Glcd_Write dans l’aide intégrée (QHelp).

2. Modifier le programme pour placer (approximativement) la phrase au milieu de l’écran.

3. La fonction sprinti est fréquemment utilisée pour formater un affichage. Elle s’utilise comme suit (voir l’aide) :

```
1         sprinti (& chaine , format , arg1 , arg2 , ...)
```

où chaine est une chaine de caractères (i.e. un tableau de char) qui sera modifiée, format est une chaine de caractères contenant des caractères ordinaires et des spécifications de format du type %0[taille][type], [taille] étant le nombre de chiffres utilisés pour l’affichage et [type] étant d pour des entiers signés et u pour des entiers non-signés. La fonction sprinti ne fonctionne qu’avec des int.

Compléter le programme suivant pour afficher ”t = 20,5”

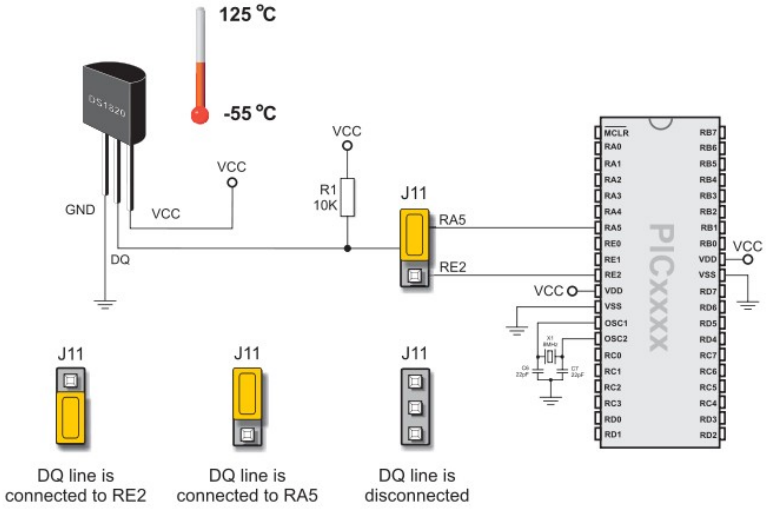
```
1         _____ text[10];
2         void main (){
```

```
3           unsigned int val = 20;
4           unsigned int dec = 5;
5           my_glcd_init ();
6           sprinti (text , ____ , val , dec );
7           Glcd_Write_Text (text , 0, 0, 1);
8         }
```

II.Communication one-wire avec le capteur DS1820

DS1820 digital thermometer is convenient for environmental temperature measurement. It can measure temperature in the range between -55°C and 125°C with 0.5°C accuracy. It must be properly placed in the 3-pin socket provided on the EasyPIC5, with its rounded side directed to the right, as marked on the board (refer to the Figure 41 below). Otherwise, the DS1820 could be permanently damaged. The DS1820 data pin can be connected to either RA5 or RE2 pin, which is determined by the jumper J11.

Figure 40 DS1820 connector



Le capteur de température DS1820 s'appuie sur le protocole one-wire pour communiquer sa mesure. Comme son nom l'indique, un seul fil est nécessaire (même si plusieurs périphériques sont utilisés).

1. Mise en place

1. Vérifier que le commutateur (switch) J11 est placé en position RE2, le déplacer si nécessaire. La ligne DQ est ainsi connectée à la broche 2 de PORTE.

2. En consultant la documentation de la librairie (QHelp, OneWire Library), donner les caractéristiques principales du protocole one-wire.

2. Communication

La librairie est composée de trois fonctions : Ow_reset(), 0w_Write() et Ow_Read. Pour lire et afficher une température, il faut suivre les étapes suivantes :

a) Envoi de la commande CONVERT_T au capteur (mesure de la température)

b) Envoi de la commande READ_SCRATCHPAD au capteur (placement de la temérature dans le buffer du capteur)

c) Lecture du buffer

d) Affichage.

Exercice 5-1 : Compléter le programme suivant (en vous aidant de la documentation) :

```
1         void main (){
2           unsigned int temp;
3           my_glcd_init ();
4           while (1) {
5             // Step a)
6             Ow_Reset (____ , ____ );
7             Ow_Write (____ , ____ , 0xCC ); //on s'adresse à tous les périphériques
          one-wire
8             Ow_Write (____ , ____ , 0x44 ); // Envoi de la commande CONVERT_T
9             Delay_us (120); // attente mesure
10            // Step b)
11            Ow_Reset (____ , ____ );
12            Ow_Write (____ , ____ , 0xCC ); // on s'adresse à tous les périphériques
          one-wire
13            Ow_Write (____ , ____ , 0xBE ); // Envoi de la commande READ_SCRATCHPAD
14            // Step c)
15            temp = Ow_Read (____ , ____ );
16            // Step d)
17            // a écrire
18            delay_ms (100);
19          }
20        }
```

Quel affichage obtenez-vous ?

3. Décodage de la température

La mesure envoyée par le capteur est codée comme indiqué par l'extrait de la datasheet du DS1820 :

TEMPERATURE REGISTER FORMAT Figure 2

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
LS Byte	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}
	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
MS Byte	S	S	S	S	S	S	S	S

TEMPERATURE/DATA RELATIONSHIP Table 2

TEMPERATURE	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+85.0°C*	0000 0000 1010 1010	00AAh
+25.0°C	0000 0000 0011 0010	0032h
+0.5°C	0000 0000 0000 0001	0001h
0°C	0000 0000 0000 0000	0000h
-0.5°C	1111 1111 1111 1111	FFFFh
-25.0°C	1111 1111 1100 1110	FFCEh
-55.0°C	1111 1111 1001 0010	FF92h

*The power-on reset value of the temperature register is +85°C

Il est donc indispensable de la décoder avant de l'afficher.

Exercice 5-2 :

1. Quels bits contiennent la partie entière de la température ?

2. Quels bits contiennent la partie décimale de la température ?

3. Quels bits contiennent le signe de la mesure ?

4. Quelle est la précision de la mesure ?

5. Créer deux variables temp_int et temp_dec déclarée en unsigned int destinées à contenir respectivement la partie entière et la partie décimale. On négligera de gérer le signe.

6. Affecter ces deux variables avec les parties enti`re et d´cimale, d´termin´e ` partir de temp. Vous pourrez utiliser des opérateurs de masque (&) et de décalage (>>, <<).

7. Corriger l'affichage en utilisant la fonction sprinti utilisée comme suit :

III.Supplément

On rappelle que la résolution graphique de l'écran est de 128 pixels horizontaux et 64 pixels verticaux. Elle sera notée 128x64 par la suite. Les coordonnées suivant l'axe x vont donc de 0 à 127 et de 0 à 63 pour y.

Exercice 5-3 :

Au choix :

1. Utiliser les fonctions de la librairie du GLCD pour réaliser un affichage graphique de la température en fonction du temps.

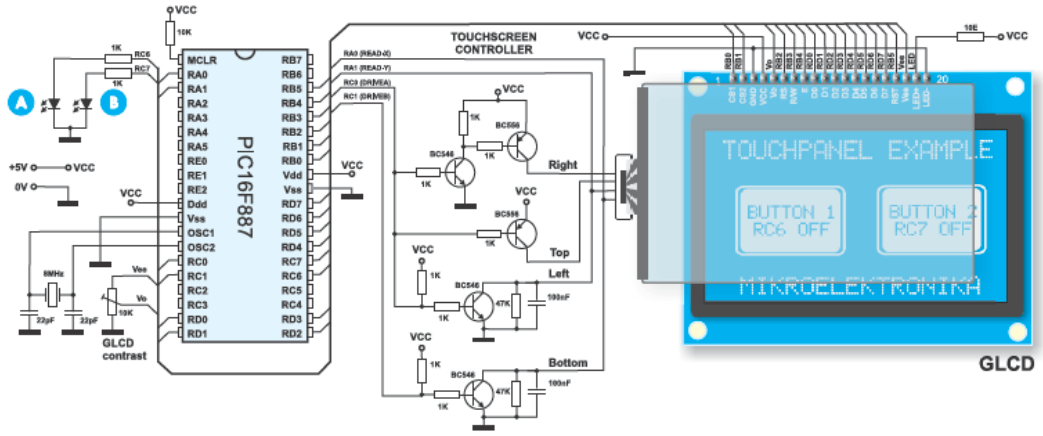
2. Lire en détail la documentation du DS1820 pour réaliser une lecture de température avec une résolution supérieure à 9 bits (voir p. 3 de la datasheet et le projet OneWire dans les examples).

Exercice 5-4 :

Réaliser un jeu de pong à deux raquettes commandées par des boutons poussoirs. L'affichage du score se fera sur les afficheurs sept segments (deux afficheurs par joueurs)

TP 6 Interfaçage d'un écran tactile

L'objectif est d'interfacer l'écran tactile pour commander l'allumage et l'extinction d'une LED. Voici le schéma détaillant la connexion de l'écran au micro-contrôleur :



I. Ressources

Diverses ressources sont récupérable à l'adresse

http://pixel-shaker.fr/fr/enseignements/geii-programmation-pic-en-c-easypic5-mikroc :

– un article (en français) sur le fonctionnement et l'utilisation de l'écran tactile ;

– un premier programme (à terminer) d'interfaçage : touchscreen1.c (reproduit ci-dessous)

– un second programme à tester (en fin de TP) : touchscreen2.c

1. Programme touchescreen1.c

```
1         unsigned int GetX() {
2         //reading X
3           PORTC.F0 = 1;   // DRIVEA = 1 (LEFT drive on, RIGHT drive on, TOP drive off)
4           PORTC.F1 = 0;   // DRIVEB = 0 (BOTTOM drive off )
5           Delay_ms(5);
6           return ADC_read(0);   // reading X value from RA0 (BOTTOM)
7         }
8         unsigned int GetY() {
9           //reading Y
10          PORTC.F0 = 0; // DRIVEA = 0 (LEFT drive off, RIGHT drive off, TOP drive on)
11          PORTC.F1 = 1; // DRIVEB = 1 (BOTTOM drive on)
12          Delay_ms(5);
13          return ADC_read(1);   // reading Y value from RA1 (from LEFT)
14        }
15
16        void main() {
17          PORTA = 0x00;
18          TRISA = 0x03;         // RA0 i RA1 are analog inputs
19          ANSEL = 0x03;
20          ANSELH = 0;           // Configure other AN pins as digital I/O
21          PORTC = 0 ;
22          TRISC = 0 ;
23         // PORTC is output
24          ...
25          while (1) {
```

```
26            // read X-Y and display it
27            x_coord = GetX();
28            y_coord = GetY();
29            ...
30            Delay_ms(100);
31          }
32        }
```

II.Travail de réalisation

1. Préparation

1. Lire le document, en particulier le paragraphe “principe de fonctionnement” et expliquer pourquoi seuls quatre fils sont nécessaires pour interfacer l’écran tactile.

2. Expliquer alors le fonctionnement des fonctions GetX() et GetY() dans le programme touchscreen1.c

2. Réalisation

Exercice 6-1 :

Placer le fichier touchscreen1.c dans un projet et compléter le programme pour, dans une boucle infinie, afficher (sur le GLCD) les coordonnées x et y acquises via les fonctions GetX() et GetY().

– Quelles sont les valeurs minimales et maximales que peuvent prendre ces coordonnées ?

– Quelle est l’orientation des axes ?

Exercice 6-2 :

1. A partir des coordonnées x et y, calculer les coordonnées x_screen et y_screen correspondant à la position courante dans le GLCD.

2. Ajouter le code suivant en début de la fonction main :

```
1         Glcd_Fill (0);                          // Clear GLCD
2         Glcd_Write_Text (" TOUCHPANEL EXAMPLE " ,10 ,0 ,1);
3         // Display Buttons on GLCD:
4         Glcd_Rectangle (8 ,16 ,60 ,48 ,1);
5         Glcd_Rectangle (68 ,16 ,120 ,48 ,1);
6         Glcd_Box (10 ,18 ,58 ,46 ,1);
7         Glcd_Box (70 ,18 ,118 ,46 ,1);
8         Glcd_Write_Text (" BUTTON1 " ,14 ,3 ,0);
9         Glcd_Write_Text ("RC6 OFF" ,14 ,4 ,0);
10        Glcd_Write_Text (" BUTTON2 " ,74 ,3 ,0);
11        Glcd_Write_Text ("RC7 OFF" ,74 ,4 ,0);
```

3. Vérifier que deux “boutons” sont dessinés. Ajouter alors le code nécessaire dans la boucle infinie pour qu’une pression sur le “bouton 1” provoque l’extinction de la LED numéro 6 du PORTC, et qu’un appui sur le “bouton 2” commande sont allumage.

Exercice 6-3 : Afficher un bargraph horizontal qui s'allonge et se rétrécit avec les deux boutons.

Exercice 6-4 :

Vérifier le programme touchscreen2.c téléchargeable à partir du lien au début de ce TP.

TP 7 Timer et interruption - mesure de temps et production de signaux périodiques

I. Mesure de durée d'exécution d'un morceau de code

Voici une fonction que l'on souhaite tester :

```
1         unsigned int div10 ( unsigned int A){
2           unsigned int Q; /* the quotient */
3           Q = ((A >> 1) + A) >> 1; /* Q = A*0.11 */
4           Q = ((Q >> 4) + Q) ; /* Q = A *0.110011 */
5           Q = ((Q >> 8) + Q) >> 3; /* Q = A *0.00011001100110011 */
6           /* either Q = A/10 or Q+1 = A/10 for all A < 534 ,890 */
7           return Q;
8         }
```

Exercice 7-1 :

1. Écrire un programme utilisant cette fonction pour afficher le résultat (sur le GLCD) de la division par 10 du nombre 171.

2. On souhaite maintenant connaitre la durée d'exécution de cette fonction en utilisant le timer TMR0. Écrire un programme qui mesure cette durée, avec l'algorithme suivant :

   (a) Initialisation du timer

   (b) Appel de la fonction

   (c) Lecture du timer

   (d) Calcul de la durée (en fonction de la fréquence d'horloge et du prescaler.

   (e) Affichage de la durée mesurée.

II.Production d'un signal périodique

Exercice 7-2 :

Générer un signal de fréquence 1 KHz sur PB0. Pour cela :

   (a) calculer la valeur de prédivision,

   (b) calculer la valeur de comptage,

   (c) écrire le programme.

Exercice 7-3 :

Générer un signal de fréquence 1 KHz de rapport cyclique 1/4.

III.Test d'interruption

1. Interruption simple

Exercice 7-4 :

Saisir et tester le programme suivant :

```
unsigned int cnt;
void interrupt () {
  cnt ++;                 // Increment value of cnt on every interrupt
  TMR0 = 96;
  INTCON = 0x20;                // Set T0IE , clear T0IF
}
void main () {
  OPTION_REG = 0x84;            // Assign prescaler to TMR0
  ANSEL = 0;                   // Configure AN pins as digital I/O
  ANSELH = 0;
  TRISB = 0;                    // PORTB is output
  PORTB = 0xFF;                 // Initialize PORTB
  TMR0 = 96;                    // Timer0 initial value
  INTCON = 0xA0;                // Enable TMRO interrupt
  cnt = 0;                      // Initialize cnt
  while (1)  {
    if (cnt  == 400) {
       PORTB = ~ PORTB ;        // Toggle PORTB LEDs
       cnt = 0;                 // Reset cnt
    }
  }
}
```

2. Réalisation de PWM (avec librairie Mikroelektronika)

On vous donne le programme suivant tiré de la documentation mikroelektronika :

```
unsigned short current_duty, old_duty, current_duty1, old_duty1;

void InitMain() {
  ANSEL  = 0;                           // Configure AN pins as digital
  ANSELH = 0;
  C1ON_bit = 0;                         // Disable comparators
  C2ON_bit = 0;

  PORTA = 255;
  TRISA = 255;                          // configure PORTA pins as input
  PORTB = 0;                            // set PORTB to 0
  TRISB = 0;                            // designate PORTB pins as output
  PORTC = 0;                            // set PORTC to 0
  TRISC = 0;                            // designate PORTC pins as output
  PWM1_Init(5000);                      // Initialize PWM1 module at 5KHz
  PWM2_Init(5000);                      // Initialize PWM2 module at 5KHz
}

void main() {
  InitMain();
  current_duty  = 16;                   // initial value for current_duty
  current_duty1 = 16;                   // initial value for current_duty1

  PWM1_Start();                         // start PWM1
  PWM2_Start();                         // start PWM2
  PWM1_Set_Duty(current_duty);          // Set current duty for PWM1
  PWM2_Set_Duty(current_duty1);         // Set current duty for PWM2

  while (1) {                           // endless loop
    if (RA0_bit) {                      // button on RA0 pressed
      Delay_ms(40);
      current_duty++;                   // increment current_duty
```

```
33              PWM1_Set_Duty(current_duty);
34             }
35
36             if (RA1_bit) {                    // button on RA1 pressed
37               Delay_ms(40);
38               current_duty--;                 // decrement current_duty
39               PWM1_Set_Duty(current_duty);
40             }
41
42             if (RA2_bit) {                    // button on RA2 pressed
43               Delay_ms(40);
44               current_duty1++;                // increment current_duty1
45               PWM2_Set_Duty(current_duty1);
46             }
47
48             if (RA3_bit) {                    // button on RA3 pressed
49               Delay_ms(40);
50               current_duty1--;                // decrement current_duty1
51               PWM2_Set_Duty(current_duty1);
52             }
53
54             Delay_ms(5);                      // slow down change pace a little
55           }
56         }
```

Exercice 7-5 :

Modifier l'exemple pour réaliser un éclairage sur une LED dont l'intensité dépend de la valeur d'un potentiomètre. Il faut donc supprimer une gestion PWM et la remplacer par la lecture d'un potentiomètre déjà vu auparavant.