

UTILISATION DU SIMULATEUR PIC : PICSIMLAB¹

1. Introduction.....	2
2. Protocole d'utilisation.....	2
3. Board McLab2 (PIC16F877A).....	2
4. Exemple de code.....	4
4.1 Test.....	5
4.2 Port UART.....	6
4.3 Oscilloscope.....	7
5. Création d'un board personnel.....	7
5.1 Le board.....	7
5.2 Test moteur pas à pas.....	8
5.2.1 Schéma.....	8
5.2.2 Explication.....	8
5.2.3 Code.....	8
5.2.4 Conclusion.....	12

¹By S.B. v29/03/20-09:43:08

1. Introduction

Ce simulateur permet de tester des programmes compilés pour différents PIC.

Le programme HEX est chargé et utilisé sur différents boards.

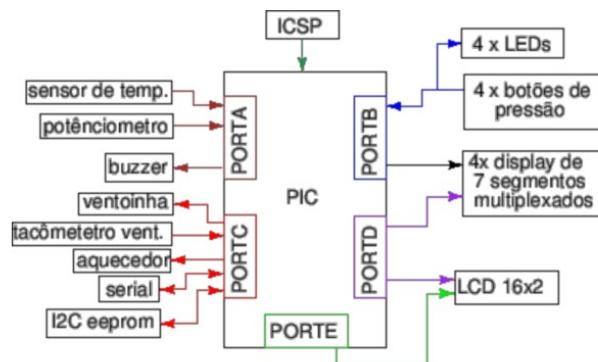
ATTENTION : la version 64 bits plante. La version 32 bits fonctionne bien.

2. Protocole d'utilisation

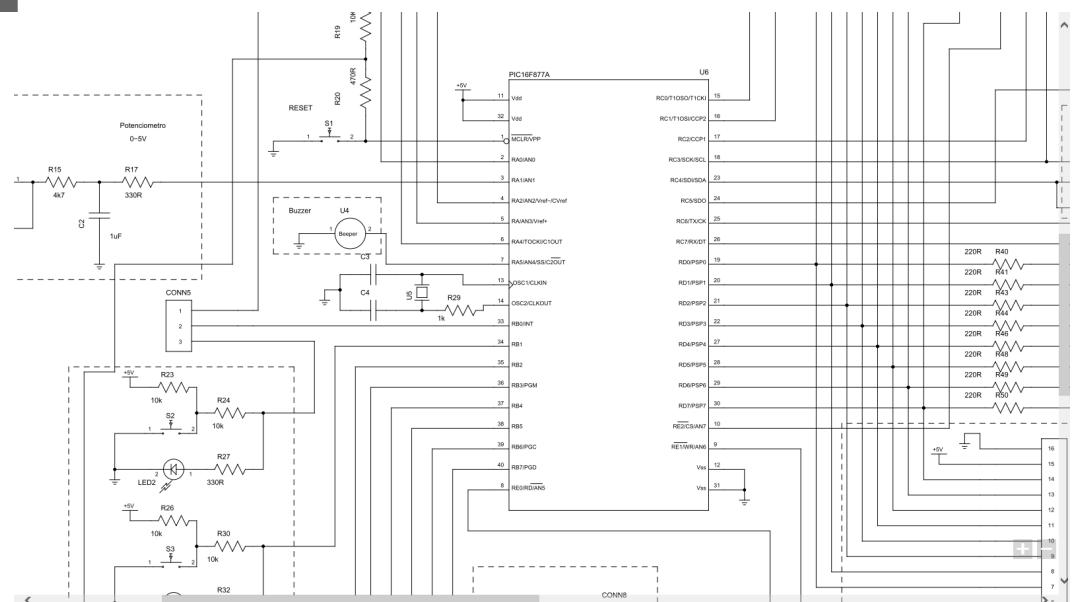
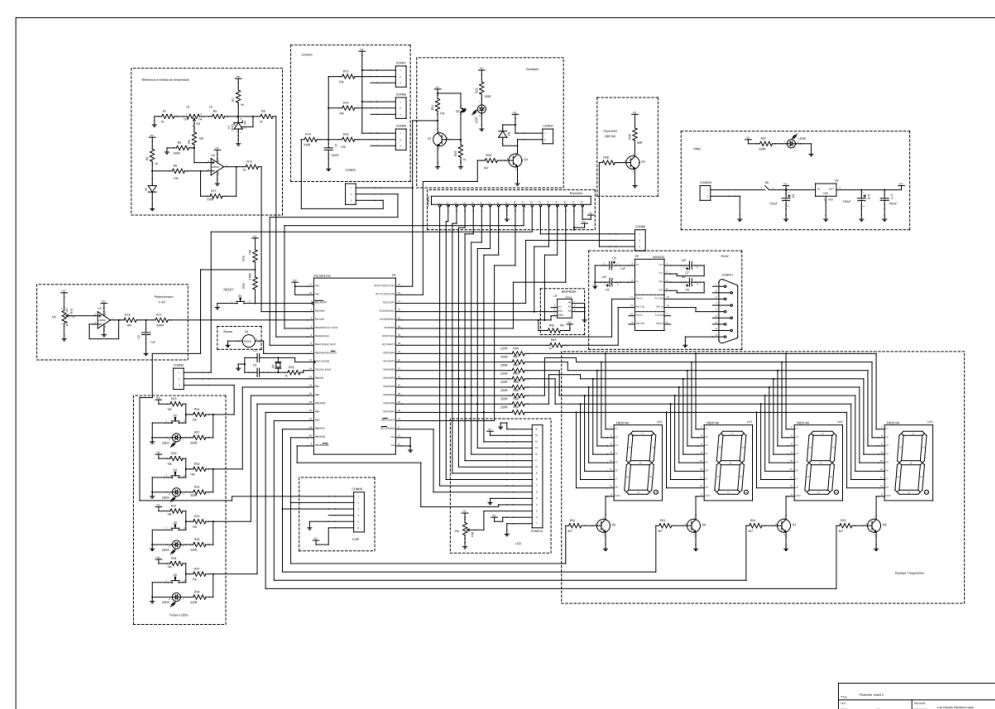
1. Écrire le programme en C en suivant les broches du schéma de la board
2. Compiler le programme en utilisant PICC ou MPLABX
3. Charger l'HEX dans la board adaptée.
4. Tester le bon fonctionnement de votre code.

La board utilisée dans ce tuto est la McLab2 en utilisant un 16F877A.

3. Board McLab2 (PIC16F877A)



Utilisation du simulateur PIC : PicSimLab



La board contient de nombreux éléments :

LED, switch, un potentiomètre P2, 4 afficheurs 7 segments multiplexé commandés par transistors, afficheur LCD parallèle.

Le schéma donne toutes les informations pour créer les programmes.

4. Exemple de code

```
#include <16F877A.h>
#device adc=8

#FUSES NOWDT          //No Watch Dog Timer
#FUSES HS              //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)
#FUSES NOPUT           //No Power Up Timer
#FUSES NOPROTECT       //Code not protected from reading
#FUSES NODEBUG          //No Debug mode for ICD
#FUSES NOBROWNOUT      //No brownout reset
#FUSES NOLVP            //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOCPD             //No EE protection
#FUSES NOWRT           //Program memory not write protected

#use delay(clock=20000000)
#define BP0_B0    PIN_B0

#define BP_L1 PIN_B1
#define BP_L2 PIN_B2
#define L3 PIN_B3
#define AFFQ8 PIN_B4
#define AFFQ7 PIN_B5
#define AFFQ6 PIN_B6
#define AFFQ5 PIN_B7
#define 7SEGA PIN_D0
#define 7SEGEB PIN_D1
#define 7SEGC PIN_D2
#define 7SEGD PIN_D3
#define 7SEGE PIN_D4
#define 7SEGF PIN_D5
#define 7SEGG PIN_D6
#define 7SEGDP PIN_D7
#define P2 PIN_A1

#define TX PIN_C6
#define RX PIN_C7
#use rs232(baud=9600,parity=N,xmit=TX,rcv=RX,bits=8)
```

Utilisation du simulateur PIC : PicSimLab

```
void main()
{
int16 valeurA0=0;

    setup_adc_ports(AN0_AN1_AN2_AN3_AN4);
    setup_adc(ADC_CLOCK_INTERNAL);
    setup_psp(PSP_DISABLED);
    setup_spi(SPI_SS_DISABLED);
    setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
    setup_timer_1(T1_DISABLED);
    setup_timer_2(T2_DISABLED,0,1);
    setup_comparator(NC_NC_NC_NC);
    setup_vref(FALSE);

    printf("Debut...");delay_ms(1000);
    // TODO: USER CODE!!

while(1){
    //Test LED
    output_high(BP_L1);delay_ms(1000);output_low(BP_L1);
    output_toggle(BP_L2);
    //Test 7SEG
    output_low(AFFQ5);output_low(AFFQ6);output_low(AFFQ7);output_low(AFFQ8);
    output_high(AFFQ8);
    output_d(0b00000110);
    //Test INTER
    if (input(BP_L1)) printf("A");
    //Test Analogique A1
    set_adc_channel(1);
    valeurA0=read_adc();
    printf("valA0= %lu",valeurA0);

} //while

}//main
```

4.1 Test

Ce programme affiche 1 sur un des afficheurs 7 segments:ok

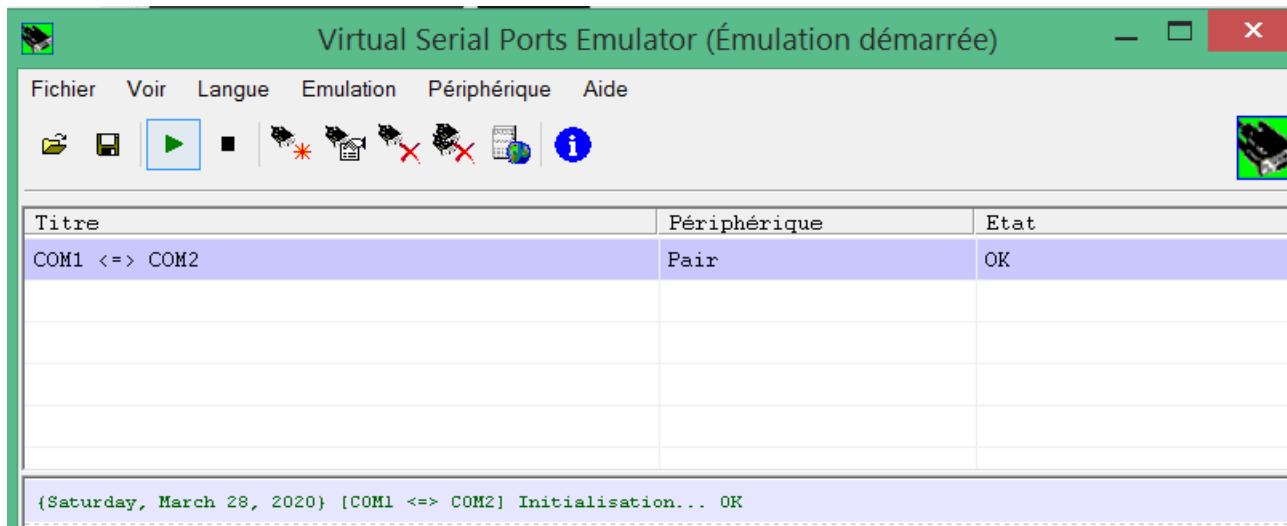
Mesure la valeur sur AN1 et l'envoie sur le port série: ok

La LED L2 clignote : ok

Un appui sur BP_L1 envoie 'A' sur le port série. : ok

4.2 Port UART

Le test de la liaison série nécessite un émulateur de port série afin de diriger les données du simulateur de pic vers un terminal série.



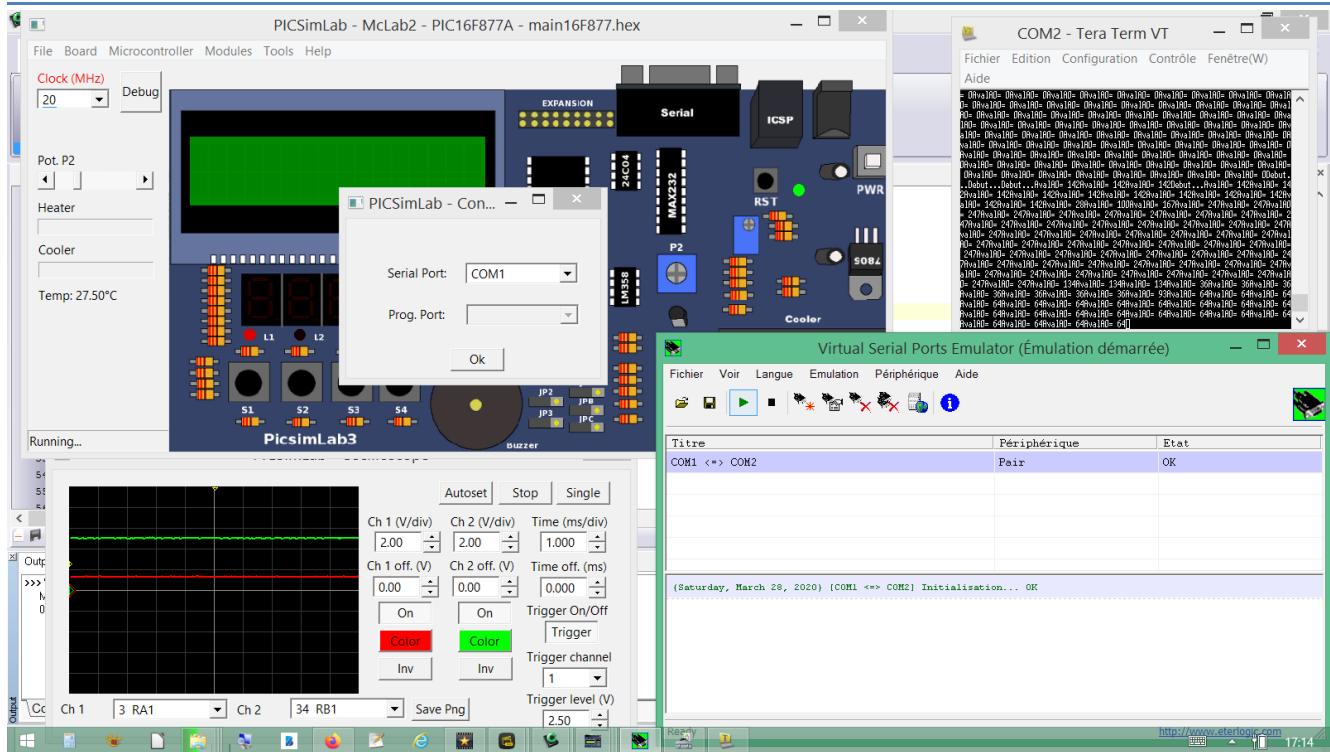
On installe 'virtual serial ports emulator' (gratuit) avec lequel on crée 2 ports virtuels : COM1 et COM2.

Sur PicSimLab : on configure l'UART sur COM1 (9600 – 8N1)

Sur un terminal (ici Teraterm) : on configure le COM2 (9600-8N1)

Les printf du code C sont alors envoyés vers Teraterm :

Utilisation du simulateur PIC : PicSimLab



4.3 Oscilloscope

Un oscilloscope permet de visualiser les broches du Pic :

Modules + Oscilloscope

Faire les réglages classiques : Calibre des tension, base de temps et déclenchement

Choisir les broches à visualiser.

5. Création d'un board personnel

**ATTENTION : Enregistrer les boards avec un nom différent à chaque sauvegarde.
Utiliser un même nom en écrasant le précédent fait PLANTER le logiciel !!!**

Un outil permet de créer son propre board :

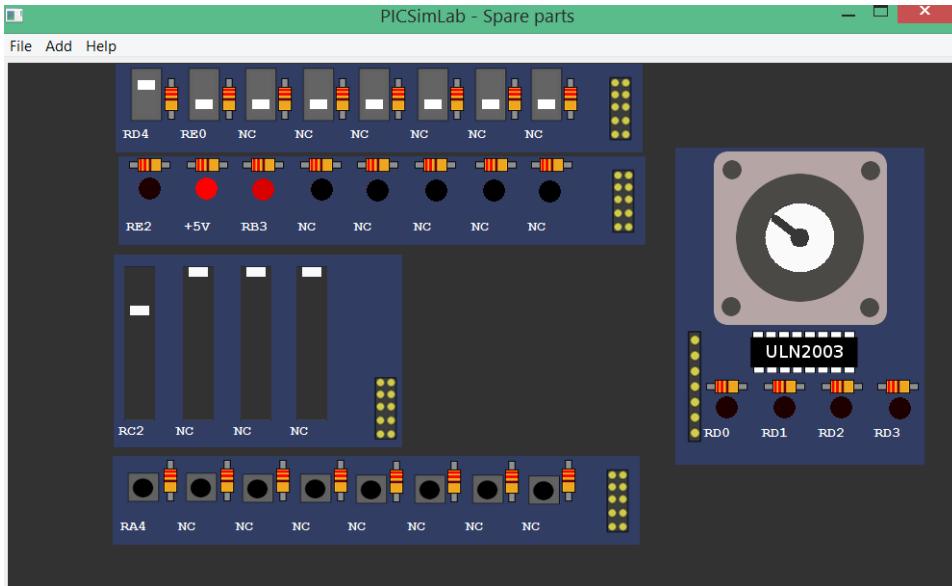
Modules + Spare parts

Un doc. En anglais est disponible dans : Help + Contents

5.1 Le board

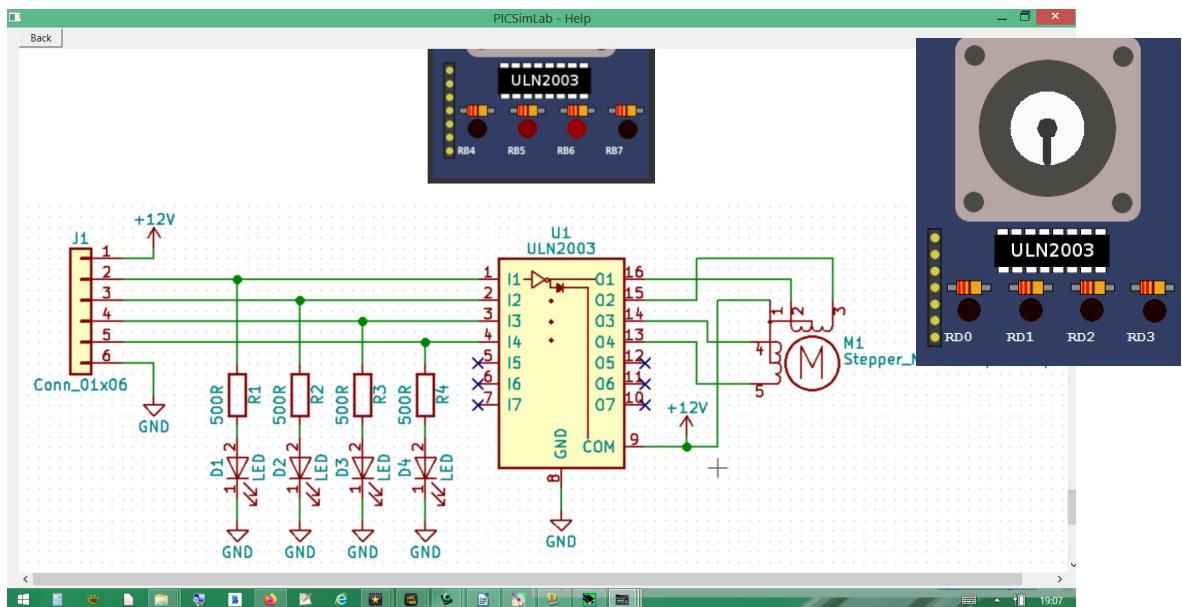
Pour faire fonctionner le board 'perso' il faut lancer aussi le board McLab2 (qui gère le PIC)

Le board 'perso' permet d'ajouter des périphériques adaptés à votre projet.



5.2 Test moteur pas à pas

5.2.1 Schéma



5.2.2 Explication

On dispose d'un MPP 4 bobines.

Les bobines sont respectivement contrôlées par RD0 RD1 RD2 RD3.

5.2.3 Code

Cette exemple est développé avec PICC et fournit un .HEX (Vous pouvez aussi utiliser MPLABX : PicSimLab s'intègre dans le logiciel – voir l'aide)

Utilisation du simulateur PIC : PicSimLab

```
#include <16F877A.h>
#device adc=8

#FUSES NOWDT          //No Watch Dog Timer
#FUSES HS              //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for
PCD)
#FUSES NOPUT           //No Power Up Timer
#FUSES NOPROTECT        //Code not protected from reading
#FUSES NODEBUG          //No Debug mode for ICD
#FUSES NOBROWNOUT       //No brownout reset
#FUSES NOLVP            //No low voltage prgming, B3(PIC16) or B5(PIC18)
used for I/O
#FUSES NOCPD           //No EE protection
#FUSES NOWRT            //Program memory not write protected

#use delay(clock=20000000)
#define BP0_B0    PIN_B0

#define BP_L1 PIN_B1
#define BP_L2 PIN_B2
#define L3 PIN_B3
#define AFFQ8 PIN_B4
#define AFFQ7 PIN_B5
#define AFFQ6 PIN_B6
#define AFFQ5 PIN_B7
#define 7SEGA PIN_D0
#define 7SEGEB PIN_D1
#define 7SEGC PIN_D2
#define 7SEGD PIN_D3
#define 7SEGE PIN_D4
#define 7SEGF PIN_D5
#define 7SEGG PIN_D6
#define 7SEGDP PIN_D7
#define P2 PIN_A1
//defin epour le moteur pas a pas MPP
#define D0a1 output_high(PIN_D0)
#define D0a0 output_low(PIN_D0)
#define D1a1 output_high(PIN_D1)
#define D1a0 output_low(PIN_D1)
```

Utilisation du simulateur PIC : PicSimLab

```
#define D2a1 output_high(PIN_D2)
#define D2a0 output_low(PIN_D2)
#define D3a1 output_high(PIN_D3)
#define D3a0 output_low(PIN_D3)

#define TX PIN_C6
#define RX PIN_C7
#use rs232(baud=9600,parity=N,xmit=TX,rcv=RX,bits=8)

void main()
{
int16 valeurA0=0;
    setup_adc_ports(AN0_AN1_AN2_AN3_AN4);
    setup_adc(ADC_CLOCK_INTERNAL);
    setup_psp(PSP_DISABLED);
    setup_spi(SPI_SS_DISABLED);
    setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
    setup_timer_1(T1_DISABLED);
    setup_timer_2(T2_DISABLED,0,1);
    setup_comparator(NC_NC_NC_NC);
    setup_vref(FALSE);

    printf("Debut...");delay_ms(1000);

while(1){
    //Test LED
    output_high(BP_L1);delay_ms(1000);output_low(BP_L1);
    output_toggle(BP_L2);
    //Test 7SEG
    output_low(AFFQ5);output_low(AFFQ6);output_low(AFFQ7);output_low(AFFQ8);
    output_high(AFFQ8);
    output_d(0b00000110);
    //Test INTER
    if (input(BP_L1)) printf("A");
    //Test Analogique A1
    set_adc_channel(1);
    valeurA0=read_adc();
    printf("valA0= %lu",valeurA0);
```

Utilisation du simulateur PIC : PicSimLab

```
//Test Spareparts board avec inter (RD4RD5) pasapas(RD0RD1RD2RD3) LED(RE5)
//Test INTER RD4 et LED RE2
printf("\r\nTest LEDRE2 INTER RD4");
if (input(PIN_D4)) output_high(PIN_E2);
//delay_ms(1000);
//Test PASAPAS RD0RD1RD2RD3 (ULN003)
printf("\r\nTest PAS A PAS\r\n");

printf("\r\nMPP ... \r\n");
//D0a1;D1a0;D2a0;D3a0;delay_ms(50);
D0a0;D1a1;D2a0;D3a0;delay_ms(50);
D0a0;D1a0;D2a1;D3a0;delay_ms(50);
D0a0;D1a0;D2a0;D3a1;delay_ms(50);

//D0a1;D1a0;D2a0;D3a0;delay_ms(50);
D0a0;D1a1;D2a0;D3a0;delay_ms(50);
D0a0;D1a0;D2a1;D3a0;delay_ms(50);
D0a0;D1a0;D2a0;D3a1;delay_ms(50);

D0a1;D1a0;D2a0;D3a0;delay_ms(50);
D0a0;D1a1;D2a0;D3a0;delay_ms(50);
D0a0;D1a0;D2a1;D3a0;delay_ms(50);
D0a0;D1a0;D2a0;D3a1;delay_ms(50);

D0a1;D1a0;D2a0;D3a0;delay_ms(50);
D0a0;D1a1;D2a0;D3a0;delay_ms(50);
D0a0;D1a0;D2a1;D3a0;delay_ms(50);
D0a0;D1a0;D2a0;D3a1;delay_ms(50);

D0a1;D1a0;D2a0;D3a0;delay_ms(50);
D0a0;D1a1;D2a0;D3a0;delay_ms(50);
D0a0;D1a0;D2a1;D3a0;delay_ms(50);
D0a0;D1a0;D2a0;D3a1;delay_ms(50);

} //while

}//main
```

5.2.4 Conclusion

L'enclenchement de l'interrupteur RB0 affiche un segment sur l'afficheur 7 segment de gauche : ok

Le moteur tourne : ok (un pas en arrière et 3 en avant : revoir la séquence des pas).