AFEKA המכללה האקדמית להנדסה בתל־אביב אפקה
TEL-AVIV ACADEMIC COLLEGE OF ENGINEERING

הנדסת תוכנה

Software Engineering

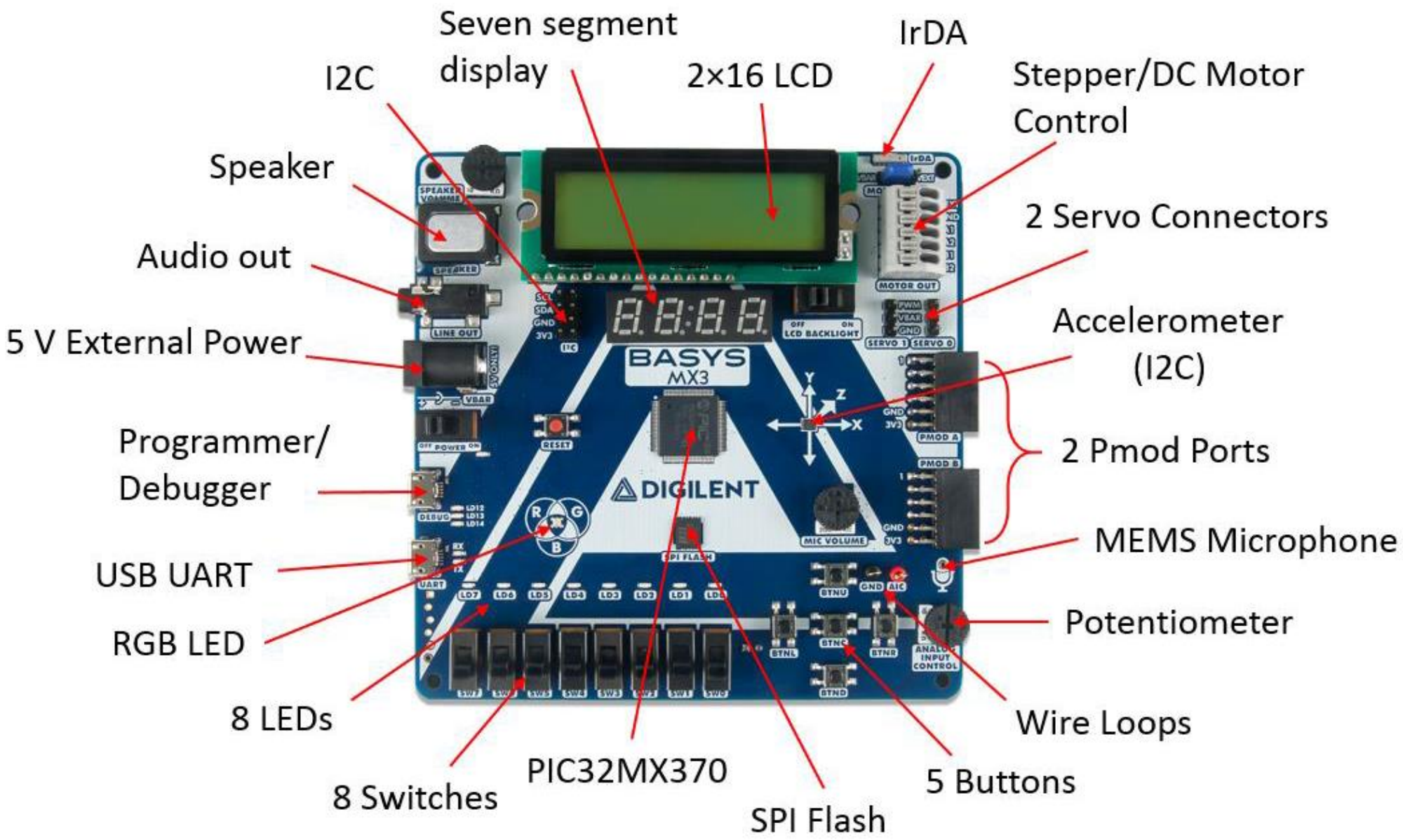
מערכות משובצות מחשב (קורס מס' 10110)

הרצאה מספר 2 – ממשק תצוגת LCD

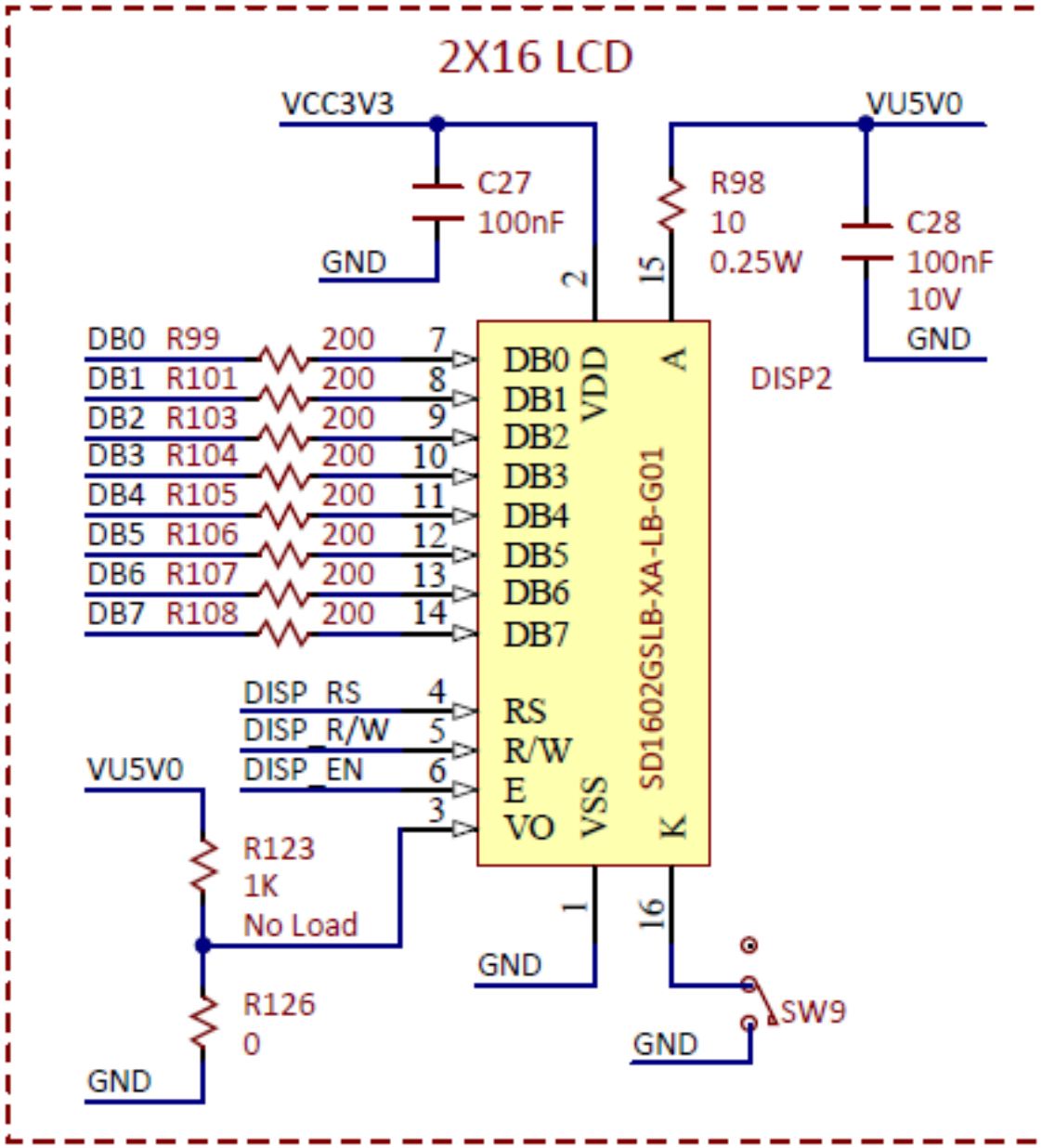
32PIC

כתב: ד"ר מנחם אפשטיין

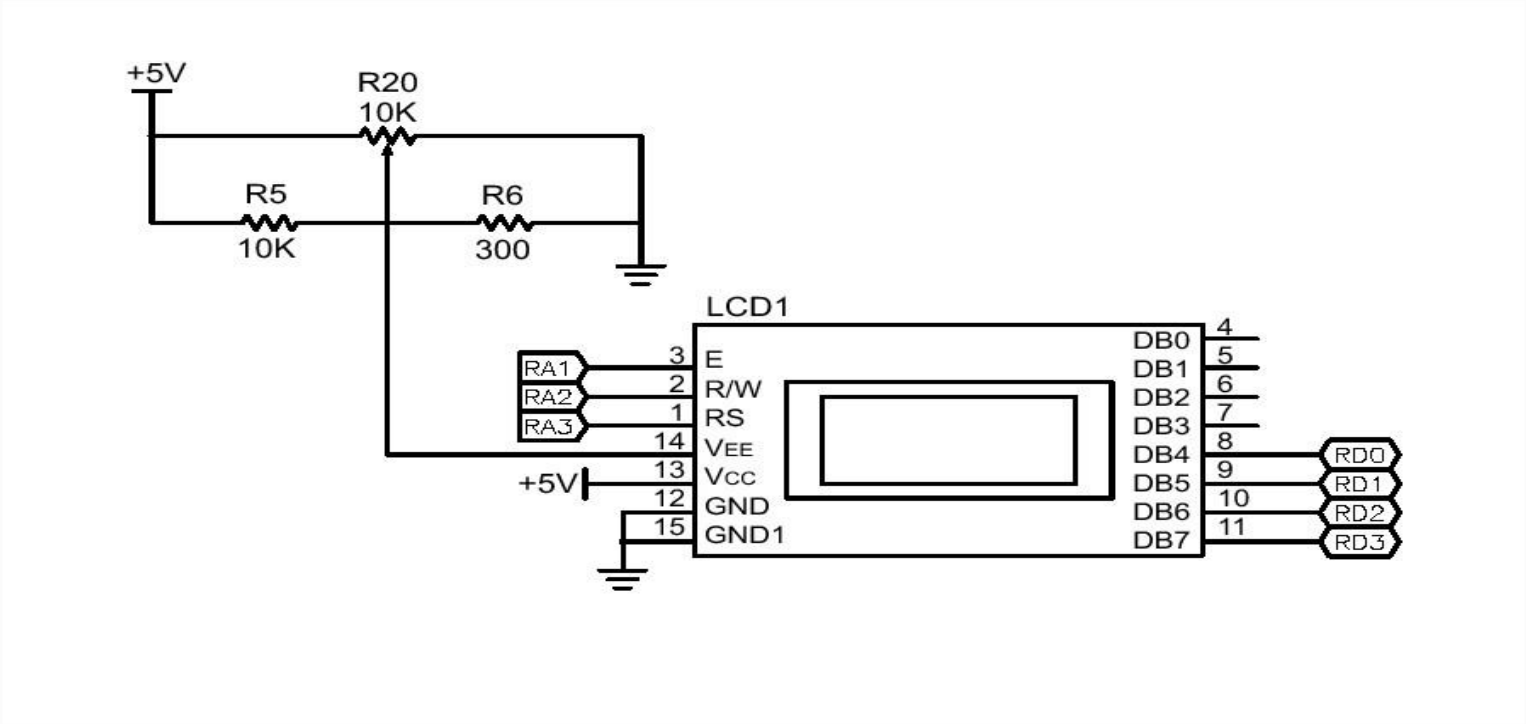
בקורס מערכות משובצות מחשב


Seven segment display
IrDA
I2C
2×16 LCD
Stepper/DC Motor Control
Speaker
2 Servo Connectors
Audio out
Accelerometer (I2C)
5 V External Power
2 Pmod Ports
Programmer/ Debugger
MEMS Microphone
USB UART
Potentiometer
RGB LED
Wire Loops
8 LEDs
PIC32MX370
5 Buttons
8 Switches
SPI Flash
BASYS MX3
DIGILENT

חיבור 8bit LCD



חיבור LCD – 4 bit



Instruction	Code RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description	Execution Time (max) (when f_{cp} or f_{OSC} is 270 kHz)
Clear display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DDRAM address 0 in address counter.	
Return home	0	0	0	0	0	0	0	0	1	—	Sets DDRAM address 0 in address counter. Also returns display from being shifted to original position. DDRAM contents remain unchanged.	1.52 ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	37 µs
Display on/off control	0	0	0	0	0	0	1	D	C	B	Sets entire display (D) on/off, cursor on/off (C), and blinking of cursor position character (B).	37 µs
Cursor or display shift	0	0	0	0	0	1	S/C	R/L	—	—	Moves cursor and shifts display without changing DDRAM contents.	37 µs
Function set	0	0	0	0	1	DL	N	F	—	—	Sets interface data length (DL), number of display lines (N), and character font (F).	37 µs
Set CGRAM address	0	0	0	1	ACG	ACG	ACG	ACG	ACG	ACG	Sets CGRAM address. CGRAM data is sent and received after this setting.	37 µs
Set DDRAM address	0	0	1	ADD	ADD	ADD	ADD	ADD	ADD	ADD	Sets DDRAM address. DDRAM data is sent and received after this setting.	37 µs
Read busy flag & address	0	1	BF	AC	AC	AC	AC	AC	AC	AC	Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.	0 µs

Instruction	Code RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description	Execution Time (max) (when f_{cp} or f_{OSC} is 270 kHz)
Write data to CG or DDRAM	1	0	Write data								Writes data into DDRAM or CGRAM.	37 µs t_{ADD} = 4 µs*
Read data from CG or DDRAM	1	1	Read data								Reads data from DDRAM or CGRAM.	37 µs t_{ADD} = 4 µs*
	I/D = 1: Increment I/D = 0: Decrement S = 1: Accompanies display shift S/C = 1: Display shift S/C = 0: Cursor move R/L = 1: Shift to the right R/L = 0: Shift to the left DL = 1: 8 bits, DL = 0: 4 bits N – 1: 2 lines, N – 0: 1 line F = 1: 5 × 10 dots, F = 0: 5 × 8 dots BF = 1: Internally operating BF = 0: Instructions acceptable										DDRAM: Display data RAM CGRAM: Character generator RAM ACG: CGRAM address ADD: DDRAM address (corresponds to cursor address) AC: Address counter used for both DD and CGRAM addresses	Execution time changes when frequency changes Example: When f_{cp} or f_{OSC} is 250 kHz, $37\ \mu s \times \frac{270}{250} = 40\ \mu s$

Name	PIC32 Pin	Description
DISP_RS	AN15/RPB15/OCFB/CTED6/PMA0/RB15	Register Select: High for Data Transfer, Low for Instruction Transfer.
DISP_RW	RPD5/PMRD/RD5	Read/Write signal: High for Read mode, Low for Write mode.
DISP_EN	RPD4/PMWR/RD4	Read/Write Enable: High for Read, falling edge writes data
DB0	PMD0/RE0	Data bits 0 -7.
DB1	PMD1/RE1	
DB2	AN20/PMD2/RE2	
DB3	RPE3/CTPLS/PMD3/RE3	
DB4	AN21/PMD4/RE4	
DB5	AN22/RPE5/PMD5/RE5	
DB6	AN23/PMD6/RE6	
DB7	AN27/PMD7/RE7	

```
#include <xc.h>
#pragma config JTAGEN = OFF
#pragma config FWDTEN = OFF
#pragma config FNOSC =        FRCPLL
#pragma config FSOSCEN =     OFF
#pragma config POSCMOD =  EC
#pragma config OSCIOFNC =   ON
#pragma config FPBDIV =     DIV_1
#pragma config FPLLIDIV =      DIV_2
#pragma config FPLLMUL =     MUL_20
#pragma config FPLLODIV =    DIV_1
#include <stdio.h>
#include <stdlib.h>
void busy(void);
void main (void)
{int j,i;
char string[]="Menachem Epstein";
char control[]={0x38,0x38,0x38,0xe,0x6,0x1};
TRISBbits.TRISB15 = 0; // RB15 (DISP_RS) set as an output
ANSELBbits.ANSB15 = 0; // disable analog functionality on RB15 (DISP_RS)
TRISDbits.TRISD5 = 0; // RD5 (DISP_RW) set as an output
TRISDbits.TRISD4 = 0; // RD4 (DISP_EN) set as an output
//TRISEbits.TRISE0 = 1; // RE0 (DB0) set as input (change 1 to 0 for
TRISE&=0xff00;
ANSELEbits.ANSE2 = 0;
ANSELEbits.ANSE4 = 0;
ANSELEbits.ANSE5 = 0;
ANSELEbits.ANSE6 = 0;
PORTBbits.RB15=0;//rs=0
PORTDbits.RD5=0;//w=0
ANSELEbits.ANSE7 = 0;
```

```
for(i=0;i<6;i++)
{
   PORTE=control[i];
   PORTDbits.RD4=1;
   PORTDbits.RD4=0;
   //for(j=0;j<32000;j++);
   busy();
}
PORTBbits.RB15=1;//rs=0
PORTDbits.RD5=0;//w=0
for(i=0;i<16;i++)
{
   PORTE=string[i];
   PORTDbits.RD4=1;
   PORTDbits.RD4=0;
   //for(j=0;j<32000;j++);
   busy();
}
}
```

```
void busy(void)
     {
       char RD,RS;
       int STATUS_TRISE;
       int portMap;
   RD=PORTDbits.RD5;
   RS=PORTBbits.RB15;
   STATUS_TRISE=TRISE;
                PORTDbits.RD5 = 1;//w/r
                PORTBbits.RB15 = 0;//rs
              portMap = TRISE;
                portMap |= 0x80;
                TRISE = portMap;
do
    {
    PORTDbits.RD4=1;//enable=1
    PORTDbits.RD4=0;//enable=0
    }
  while(PORTEbits.RE7); // BF רגסטר
       PORTDbits.RD5=RD;
       PORTBbits.RB15=RS;
       TRISE=STATUS_TRISE;
       }
```

```
/*
 * File:   lcd_fader.c
 * Author: MenachemE
 *
 * Created on July 11, 2019, 5:49 PM
 */

#include <xc.h>
#pragma config JTAGEN = OFF
#pragma config FWDTEN = OFF
#pragma config FNOSC =  FRCPLL
#pragma config FSOSCEN =                OFF
#pragma config POSCMOD =                EC
#pragma config OSCIOFNC =               ON
#pragma config FPBDIV =     DIV_1
#pragma config FPLLIDIV =               DIV_2
#pragma config FPLLMUL =                MUL_20
#pragma config FPLLODIV =               DIV_1
#include <stdio.h>
#include <stdlib.h>
void busy(void);
void main (void)
{int j,i;
char  CG_father[16]={0x09,0x0a,0x0c,0x0c,0x0c,0x0a,0x09,0x09,//?
                     0x26,0x29,0x31,0x21,0x29,0x2b,0x36,0x34};//?
char father[3]={0,1,0};// אבא
char control[7]={0x38,0x38,0x38,0xe,0x6,0x1,0x40};
// set CGRAM=0x40
char string[]="Menachem Epstein";
```

```
TRISBbits.TRISB15 = 0; // RB15 (DISP_RS) set as an output
ANSELBbits.ANSB15 = 0; // disable analog functionality on RB15 (DISP_RS)
TRISDbits.TRISD5 = 0; // RD5 (DISP_RW) set as an output
TRISDbits.TRISD4 = 0; // RD4 (DISP_EN) set as an output
TRISE&=0xff00;
ANSELEbits.ANSE2 = 0;
ANSELEbits.ANSE4 = 0;
ANSELEbits.ANSE5 = 0;
ANSELEbits.ANSE6 = 0;
ANSELEbits.ANSE7 = 0;
PORTBbits.RB15=0;//rs=0
PORTDbits.RD5=0;//w=0
for(i=0;i<7;i++)
{
   PORTE=control[i];
   PORTDbits.RD4=1;
   PORTDbits.RD4=0;
busy();
}
  PORTBbits.RB15 = 1;//rs
      for(i = 0;i < 16;i++)
        {
      PORTE=CG_father[i];
      PORTDbits.RD4=1;//enable=1
      PORTDbits.RD4=0;//enable=0
busy();
      }
```

```
PORTBbits.RB15 = 0;//rs control
  PORTE=0x80;//DDRAM
  PORTDbits.RD4=1;//enable=1
  PORTDbits.RD4=0;//enable=0
  //for(j=0;j<32000;j++);
  busy();
  PORTBbits.RB15 = 1;//rs

for(i = 0;i < 3;i++)
    {
       PORTE=father[i];
       PORTDbits.RD4=1;//enable=1
       PORTDbits.RD4=0;//enable=0
busy();
     }
}
```

```
void busy(void)
     {
       char RD,RS;
       int STATUS_TRISE;
       int portMap;
   RD=PORTDbits.RD5;
   RS=PORTBbits.RB15;
   STATUS_TRISE=TRISE;
                PORTDbits.RD5 = 1;//w/r
                PORTBbits.RB15 = 0;//rs
              portMap = TRISE;
                portMap |= 0x80;
                TRISE = portMap;
do
    {
    PORTDbits.RD4=1;//enable=1
    PORTDbits.RD4=0;//enable=0
    }
  while(PORTEbits.RE7); // BF רגסטר
       PORTDbits.RD5=RD;
       PORTBbits.RB15=RS;
       TRISE=STATUS_TRISE;
       }
```

110