```
#include <P18f452.h>
#include <delays.h>

#pragma config WDT = OFF
#pragma config OSC = XT
```

برای شروع پایه هایی را که به LCD متصل می شوند را معرفی می کنیم:

```
#define RS  PORTDbits.RD0
#define RW  PORTDbits.RD1
#define EN  PORTDbits.RD2
#define CS1 PORTDbits.RD3
#define CS2 PORTDbits.RD4
#define RST PORTDbits.RD5

#define DirRS  TRISDbits.TRISD0
#define DirRW  TRISDbits.TRISD1
#define DirEN  TRISDbits.TRISD2
#define DirCS1 TRISDbits.TRISD3
#define DirCS2 TRISDbits.TRISD4
#define DirRST TRISDbits.TRISD5

#define GlcdDataBus    PORTB
#define DirGlcdDataBus TRISB
```

برای نمایش فونت بر روی LCD باید دیتابیسی نوشته شود که اطلاعات گرافیکی مربوط به هر کاراکتر در آن موجود باشد که برای این منظور از دیتابیس از قبل نوشته شده ی کامپایلر بسکام استفاده شده:

```
const rom unsigned char Bascom8x8Font[760] = {
 0,0,0,0,0,0,0,0
,0,0,6,95,6,0,0,0
,0,7,3,0,7,3,0,0
,0,36,126,36,126,36,0,0
,0,36,43,106,18,0,0,0
,0,99,19,8,100,99,0,0
,0,54,73,86,32,80,0,0
,0,0,7,3,0,0,0,0
,0,0,62,65,0,0,0,0
,0,0,65,62,0,0,0,0
,0,8,62,28,62,8,0,0
,0,8,8,62,8,8,0,0
,0,0,224,96,0,0,0,0
,0,8,8,8,8,8,0,0
,0,0,96,96,0,0,0,0
```

,0,32,16,8,4,2,0,0
,0,62,81,73,69,62,0,0
,0,0,66,127,64,0,0,0
,0,98,81,73,73,70,0,0
,0,34,73,73,73,54,0,0
,0,24,20,18,127,16,0,0
,0,47,73,73,73,49,0,0
,0,60,74,73,73,48,0,0
,0,1,113,9,5,3,0,0
,0,54,73,73,73,54,0,0
,0,6,73,73,41,30,0,0
,0,0,108,108,0,0,0,0
,0,0,236,108,0,0,0,0
,0,8,20,34,65,0,0,0
,0,36,36,36,36,36,0,0
,0,0,65,34,20,8,0,0
,0,2,1,89,9,6,0,0
,0,62,65,93,85,30,0,0
,0,126,17,17,17,126,0,0
,0,127,73,73,73,54,0,0
,0,62,65,65,65,34,0,0
,0,127,65,65,65,62,0,0
,0,127,73,73,73,65,0,0
,0,127,9,9,9,1,0,0
,0,62,65,73,73,122,0,0
,0,127,8,8,8,127,0,0
,0,0,65,127,65,0,0,0
,0,48,64,64,64,63,0,0
,0,127,8,20,34,65,0,0
,0,127,64,64,64,64,0,0
,0,127,2,4,2,127,0,0
,0,127,2,4,8,127,0,0
,0,62,65,65,65,62,0,0
,0,127,9,9,9,6,0,0
,0,62,65,81,33,94,0,0
,0,127,9,9,25,102,0,0
,0,38,73,73,73,50,0,0
,0,1,1,127,1,1,0,0
,0,63,64,64,64,63,0,0
,0,31,32,64,32,31,0,0
,0,63,64,60,64,63,0,0
,0,99,20,8,20,99,0,0
,0,7,8,112,8,7,0,0

```
,0,113,73,69,67,0,0,0
,0,0,127,65,65,0,0,0
,0,2,4,8,16,32,0,0
,0,0,65,65,127,0,0,0
,0,4,2,1,2,4,0,0
,128,128,128,128,128,128,128,128
,0,0,3,7,0,0,0,0
,0,32,84,84,84,120,0,0
,0,127,68,68,68,56,0,0
,0,56,68,68,68,40,0,0
,0,56,68,68,68,127,0,0
,0,56,84,84,84,8,0,0
,0,8,126,9,9,0,0,0
,0,24,164,164,164,124,0,0
,0,127,4,4,120,0,0,0
,0,0,0,125,64,0,0,0
,0,64,128,132,125,0,0,0
,0,127,16,40,68,0,0,0
,0,0,0,127,64,0,0,0
,0,124,4,24,4,120,0,0
,0,124,4,4,120,0,0,0
,0,56,68,68,68,56,0,0
,0,252,68,68,68,56,0,0
,0,56,68,68,68,252,0,0
,0,68,120,68,4,8,0,0
,0,8,84,84,84,32,0,0
,0,4,62,68,36,0,0,0
,0,60,64,32,124,0,0,0
,0,28,32,64,32,28,0,0
,0,60,96,48,96,60,0,0
,0,108,16,16,108,0,0,0
,0,156,160,96,60,0,0,0
,0,100,84,84,76,0,0,0
,0,8,62,65,65,0,0,0
,0,0,0,119,0,0,0,0
,0,0,65,65,62,8,0,0
,0,2,1,2,1,0,0,0
};
```

عکسی را که باید با استفاده از تابع مربوطه روی LCD چاپ کنیم با استفاده از نرم افزار شرکت MikroElektronika به آرایه تبدیل کرده و ضمیمه ی برنامه می کنیم:

```
const rom unsigned char MikroE_Logo_BMP[1024] = {
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0,192,224,112,176,176,176,176,176,176,176,176,
176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,
176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,
176,176,176,176,176,176,176,176,176,176, 96,224,128, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0,255,255, 0,255,255,255, 3, 35, 49, 49, 49,
49, 49, 49, 49, 51, 3,199,255,127, 31, 31, 15, 7, 7, 3, 3,
3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 7, 7, 15,
15, 31,127,255,255,255,255,255,255, 0,255,255, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0,255,255, 0,255,255,255, 7, 6, 6, 14, 30,
30, 30, 30, 30,254,255,255, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0,252,255,255,255,254, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 7,255,255,255,255, 0,255,255, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0,255,255, 0,255,255,255, 17, 51, 3, 3, 15,
7, 3, 3, 17,255,255,255, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0,129,129,129,129,128,128,128,128,128,128,128,128,
128,128,128,128,128,255,255,255,255, 0,255,255, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0,255,255, 0,255,255,255, 98,127,111, 99, 99,
99, 99, 99, 99,255,255,255, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0,255,255,255,255,255,255, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3,255,255,255,255, 0,255,255, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

```
  0, 0, 0, 0, 0,255,255, 0,255,255,255, 8,140,140,140,140,
 140,140,140,140,141,159,255,240,192, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 7, 31, 63, 31, 31, 7, 0, 0, 0, 0, 0, 0, 0,
  0, 0,128,192,248,255,255,255,255, 0,255,255, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0,127,255,128,127,127,127,112,113,113,113,113,
 113,113,113,113,113,113,113,113,113,115,119,126,124,124,120,120,112,
 112,112,112,112,112,112,112,112,112,112,112,112,112,120,120,124,
 126,127,127,127,127,127,127,127, 63,192,255,127, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 1, 1, 3, 3, 3, 3, 3, 3, 3, 3,
  3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
  3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
  3, 3, 3, 3, 3, 3, 3, 3, 3, 1, 0, 0, 0, 0, 0, 0
};

float sqrt( float x );
char *strcpypgm2ram(char *dest,const rom char *src);
```

برای ریست کردن LCD کافی است که به پایه ی ریست آن با توجه به Low Active بودنش یک پالس صفر بدهیم:(این تابع فقط توسط تابع InitGlcd فراخوانی می شود)

```
//ResetGlcd ------------------------------------------------------------------------------------------------------------------------------------
void ResetGlcd(void){

  RST = 0;
  RST = 1;
}
```

برای ارسال یا دریافت دیتا باید به پایه ی Enable با توجه به High Active بودنش یک پالس یک بدهیم:

```
//LatchData ------------------------------------------------------------------------------------------------------------------------------------
void LatchData(void){

  EN = 1;
  EN = 0;
  Delay10TCYx(5);
}
```

برای ارسال فرمان به LCD پایه های RS و RW طبق جدول صفحه 13 دیتاشیت باید صفر شوند و بعد از قرار دادن دیتای فرمان بر روی باس با فراخوانی تابع LatchData فرمان ارسال شود:

```
//WriteCmdGlcd ----------
void WriteCmdGlcd(unsigned char cmd){

  RS = 0;
  RW = 0;
  GlcdDataBus = cmd;
  LatchData();
}
```

برای ارسال دیتا به LCD پایه های RS و RW طبق جدول صفحه 13 دیتاشیت باید به ترتیب یک و صفر شوند و بعد از قرار دادن دیتای فرمان بر روی باس با فراخوانی تابع LatchData فرمان ارسال شود:

```
//WriteDataGlcd ----------
void WriteDataGlcd(unsigned char data){

  RS = 1;
  RW = 0;
  GlcdDataBus = data;
  LatchData();
}
```

تابع زیر برای خواندن دیتای نوشته شده ی از آدرس دهی شده به کار می رود:

```
//ReadDataGlcd ----------
unsigned char ReadDataGlcd(void){

   unsigned char data;
```

طبق جدول صفحه ی 13 دیتاشیت برای خواندن دیتا از LCD پایه های RS و RW هر دو باید یک منطقی باشند:

```
  RS = 1;
  RW = 1;
```

چون اینبار قرار است که میکرو از LCD دیتا دریافت کند پس باس دیتا را ورودی می کنیم:

```
  DirGlcdDataBus = 0xFF;
```

برای اینکه LCD فرمان را اجرا کند یک پالس به آن می دهیم:

```
  EN = 1;
  EN = 0;
```

پایه ی Enable را یک منطقی کرده و بعد از یک تاخیر کوتاه دیتا را از باس دریافت نموده و دوباره Enable را صفر می کنیم و پس از آن باس را به حالت عادی خود برگردانده و دیتای دریافت شده را در خروجی تابع قرار می دهیم:

```
 EN = 1;
 Delay10TCYx(1);
 data = GlcdDataBus;
 EN = 0;
 DirGlcdDataBus = 0x00;

 return(data);
}
```

به علت اینکه این LCD از دو بخش جداگانه ی چپ و راست تشکیل شده باید برای خواندن دیتا یا نوشتن در هر سمت آن سمت را انتخواب و سپس نوع عملیات مورد نظر را انجام دهیم:

برای انتخاب سمت چپ باید با توجه به Low Active بودن پایه های انتخاب به CS1 صفر داده و به CS2 یک بدهیم:

```
//LeftSelect ----------
void LeftSelect(void){

 CS1 = 0;
 CS2 = 1;
 Delay10TCYx(10);
}
```

برای انتخاب سمت راست باید با توجه به Low Active بودن پایه های انتخاب به CS1 یک داده و به CS2 صفر بدهیم:

```
//RightSelect ----------
void RightSelect(void){

 CS1 = 1;
 CS2 = 0;
 Delay10TCYx(10);
}
```

فرآیند ست کردن یک پیکسل در این نوع LCD به علت عدم امکان آدرس دهی مستقیم به پیکسل مورد نظر کمی پیچیده و کند می باشد,در هر حال برای این کار تابع زیر نوشته شده:

```
//SetDot ----------Ex
void SetDot(signed char x,signed char y,unsigned char c){

  unsigned char data;
```

در مرحله ی اول به علت اینکه LCD از دو بخش جداگانه ی چپ و راست تشکیل شده و هر بخش به صورت جداگانه آدرس دهی می شود باید تشخیص داده شود که پیکسل خواسته شده در کدام بخش قرار دارد و عملیات انتخاب سمت چپ یا راست صفحه و سپس آدرس دهی ستون

مربوط به آن پیکسل که با OR کردن X با عدد 64 طبق جدول صفحه ی 13 دیتاشیت و ارسال دیتای محاسبه شده انجهم می شود:

```
if(x<64) {
  LeftSelect();
  x = x|0x40;
}else {
  RightSelect();
  x = (x-64)|0x40;
}
WriteCmdGlcd(x);
```

حال به علت اینکه هر سمت از 8 پیج تشکیل شده (هر پیج شامل 64 بایت) برای آدرس دهی پیجی که پیکسل مورد نظر در آن است کافیست که Y را بر 8 تقسیم کرده و طبق جدول صفحه ی 13 دیتاشیت با OR 184 کنیم و سپس به LCD ارسال کنیم:

```
WriteCmdGlcd((y/8)|0xB8);
```

پس تا الآن بایتی را که پیکسل مورد نظر در آن قرار دارد آدرس دهی شد و قبل از ارسال دیتا به این بایت باید دیتایی را که قبلا در آن قرار داشته گرفته شود و بعد از پردازش بر روی آن دیتای جدید ارسال شود,این کار را به این علت انجام می دهیم که دیتای قبل از بین نورد چون که قرار است فقط یک بیت آن دست کاری شود:

```
data = ReadDataGlcd();
```

به علت اینکه بعد از خواند یا نوشتن دیتا,رجیستر LCD Y به صورت خودکار یک واحد اضافه می شود و به سراغ بایت بعدی میرود باید دوباره آدرس دهی شود:

```
WriteCmdGlcd(x);
```

در آخر طی عملیاتی با توجه به ست یا ریست کردن پیکسل باید بیت مورد نظر در دیتای قبلی 1 یا صفر شود:

```
  WriteDataGlcd(c ? ((1<<(y-((y/8)*8)))|data) : ((~(1<<(y-((y/8)*8))))&data));
}
```

رسم خط عمودی:

```
//DrawVerticalLine ----------------------------------------------------------------------------------------------------------------Ex
void DrawVerticalLine(unsigned char Y0,unsigned char Y1,unsigned char X,unsigned char Color) {
```

برای رسم یک خط عمودی کافیست تعدادی نقطه در امتداد یک خط عمودی که مکانش توسط X مشخص شده رسم کنیم پس مختصات نقاط با X ثابت و Y از Y0 تا Y1 می باشند:

```
  for(Y0;Y0<=Y1;Y0++) SetDot(X,Y0,Color);
}
```

رسم خط افقی:

```
//DrawHorizontalLine ----------------------------------------------------------------------------------------------------------------Ex
void DrawHorizontalLine(unsigned char X0,unsigned char X1,unsigned char Y,unsigned char Color) {
```

برای رسم یک خط افقی کافیست تعدادی نقطه در امتداد یک خط افقی که مکانش توسط Y مشخص شده رسم کنیم پس مختصات نقاط با Y ثابت و X از X0 تا X1 می باشند:

```
 for(X0;X0<=X1;X0++) SetDot(X0,Y,Color);
}
```

رسم خط با زاویه ی دلخواه:

```
//DrawLine ----------------------------------------------------------------------------------------------------------------------------------------Ex
void DrawLine(signed char X0,signed char Y0,signed char X1,signed char Y1,unsigned char Color) {

  signed int Dy,Dx,Fraction;
  signed char StepX,StepY;
```

اولین مرحله برای رسم خط تشخیص بیشتر یا کمتر بودن مقدارهای X0 و Y0 از X1 و Y1 با هدف ترسیم صحیح خط می باشد:

```
 Dy=Y1-Y0;
 Dx=X1-X0;
```

بعد از تشخیص مثبت نا منفی بودن رابطه ی بین مختصات شروع و پایان بر این اساس باید پله های رسم خط افزایشی یا کاهشی شوند:

```
 if(Dy<0) {Dy=-Dy;StepY=-1;} else {StepY=1;}
 if(Dx<0) {Dx=-Dx;StepX=-1;} else {StepX=1;}
```

اختلاف های بدست آمده برای محاسبات بعدی در 2 ضرب می شوند:

```
 Dy<<=1;
 Dx<<=1;
```

رسم اولین نقطه,چون در حلقه ی رسم خط رسم نمی شود:

```
 SetDot(X0,Y0,Color);
```

در صورت بزرگتر بودن Dx از Dy خط مایل به افقی است و به روش زیر رسم می شود:

```
 if(Dx>Dy) {
```

با فرمول نوشته شده شیب خط محاسبه می شود:

```
  Fraction=Dy-(Dx>>1);
```

شروع حلقه ی رسم:

```
  while(X0!=X1) {
```

در صورت مثبت بودن شیب Y0 که مختصات عمودی نقطه را در خود دارد باید با StepY که مشخص کننده ی پله ی افزایشی یا کاهشی می باشد جمع شود و از Fraction مقداری برابر با DX کم می شود:

```
   if(Fraction>=0) {

    Y0+=StepY;
    Fraction-=Dx;
   }
```

بعد از مشخص شدن وضعیت مختصات عمودی نقطه,نوبت به مختصات افقی می رسد و بعد از جمع کردن آن با متغییر StepX دوباره نوبت به جمع کردن Fraction با اینبار Dy می شود,با این کار Fraction گاهی بیشتر از صفر و گاهی کمتر از آن است که باعث می شود مختصات عمودی نقطه

با شیب تعیین شده و با سرعتی کمتر از مختصات افقی افزایش یا کاهش یابد:

```
    X0+=StepX;
    Fraction+=Dy;
```

و در نهایت رسم نقطه:

```
    SetDot(X0,Y0,Color);
   }
```

در صورت بزرگتر بودن Dy از Dx خط مایل به عمودی است و به روش زیر رسم می شود:

```
  }else {
```

با فرمول نوشته شده شیب خط محاسبه می شود:

```
   Fraction=Dx-(Dy>>1);
```

شروع حلقه ی رسم:

```
   while (Y0!=Y1) {
```

در صورت مثبت بودن شیب X0 که مختصات افقی نقطه را در خود دارد باید با StepX که مشخص کننده ی پله ی افزایشی یا کاهشی می باشد جمع شود و از Fraction مقداری برابر با Dy کم می شود:

```
    if(Fraction>=0) {

      X0+=StepX;
      Fraction-=Dy;
    }
```

بعد از مشخص شدن وضعیت مختصات عمودی نقطه,نوبت به مختصات افقی می رسد و بعد از جمع کردن آن با متغییر StepY دوباره نوبت به جمع کردن Fraction با اینبار Dx می شود,با این کار Fraction گاهی بیشتر از صفر و گاهی کمتر از آن است که باعث می شود مختصات افقی نقطه با شیب تعیین شده و با سرعتی کمتر از مختصات عمودی افزایش یا کاهش یابد:

```
    Y0+=StepY;
    Fraction+=Dx;
```

و در نهایت رسم نقطه:

```
    SetDot(X0,Y0,Color);
   }
  }
}
```

رسم مربع:

```
//DrawRectangle ----------------------------------------------------------------------------------------------------------------------------Ex
void DrawRectangle(unsigned char X0,unsigned char Y0,unsigned char X1,unsigned char Y1,unsigned char Fill,unsigned char Color) {
```

این تابع برای رسم مربع هم به صورت توپر و تو خالی نوشته شده که با مقدار یکی از آرگومان های ورودی به نام Fill تعیین می شود مربع چگونه رسم شود:

```
  if(Fill) {
```

در صورت یک بودن مقدار Fill می توان با رسم خطوط افقی با طولی برابر با X0 تا X1 به تعداد Y0 تا Y1 یک مربع توپر رسم کرد:

```
  for(Y0;Y0<=Y1;Y0++)
    DrawHorizontalLine(X0,X1,Y0,Color);

 }else {
```

و در صورت صفر بودن مقدار Fill می توان با توجه به مقدار های X0,Y0,X1,Y1 چهار ضلع یک مربع را رسم کرد تا یک مربع توخالی تشکیل شود:

```
  DrawHorizontalLine(X0,X1,Y0,Color);
  DrawVerticalLine(Y0,Y1,X0,Color);
  DrawHorizontalLine(X0,X1,Y1,Color);
  DrawVerticalLine(Y0,Y1,X1,Color);
 }
}
```

رسم دایره:

```
//DrawCircle ----------------------------------------------------------------------------------------------------------------------------------Ex
void DrawCircle(unsigned int X,unsigned int Y,unsigned int R,unsigned char Fill,unsigned char Color) {

  unsigned int Rx,Ry;
```

این تابع برای رسم دایره هم به صورت توپر و تو خالی نوشته شده که با مقدار یکی از آرگومان های ورودی به نام Fill تعیین می شود دایره چگونه رسم شود:

```
 if(Fill) {
```

در صورت یک بودن مقدار Fill می توان با رسم خطوط افقی با طولی که به وسیله ی تابع رسم کمان محاسبه می شود رسم کرد که با رسم بخش محاسبه شده به صورت عادی و قرینه دایره را کامل کرد:

```
  for(Rx=0;Rx<=R;Rx++) {

   Ry = sqrt((R*R)-(Rx*Rx));

   DrawHorizontalLine(X-Ry,X+Ry,Y+Rx,Color);
   DrawHorizontalLine(X-Ry,X+Ry,Y-Rx,Color);
  }
 }else {
```

و در صورت صفر بودن مقدار Fill باید یک دایره ی توخالی رسم شود که بعد از محاسبه ی بخش بخش کمان نقطه ی بدست آمده به 8 صورت در 8 سمت رسم می شود:

```
  for(Rx=0;Rx<=R;Rx++) {

   Ry = sqrt((R*R)-(Rx*Rx));
```

```
    SetDot(Rx+X,Ry+Y,Color);
    SetDot(Ry+X,Rx+Y,Color);
    SetDot((Rx*-1)+X,Ry+Y,Color);
    SetDot(Ry+X,(Rx*-1)+Y,Color);
    SetDot(Rx+X,(Ry*-1)+Y,Color);
    SetDot((Ry*-1)+X,Rx+Y,Color);
    SetDot((Rx*-1)+X,(Ry*-1)+Y,Color);
    SetDot((Ry*-1)+X,(Rx*-1)+Y,Color);
   }
  }
}
```

تابع پاک کردن صفحه ی LCD:

```
//ClsGlcd ----------------------------------------------------------------------------------------------------------------------------------Ex
void ClearGlcd(unsigned char Color){

  unsigned char Page,X,R;
```

برای پاک کردن صفحه ی نمایشگر یک ورودی داریم که یا صفر یا یک باید باشد در صورت 1 بودن بعد از ضرب در 255 یک دیتا ی 8 بیتی داریم که طی یک عملیات ساده به این صورت که اول انتخاب سمت چپ یا راست LCD و سپس انتخاب Page و در نهایت ارسال دیتای 0 یا 255 که کل صفحه را سیاه یا سفید می کنند:

```
 Color *= 255;
```

شروع حلقه ی رسم که فقط دو بار اجرا می شود برای رسم بر روی سمت چپ و سپس سمت راست:

```
 for(R=0;R<2;R++) {
```

انتخاب کردن سمت صفحه بر اساس اجرای اول یا دوم حلقه:

```
  if(R == 0) LeftSelect();
  else RightSelect();
```

شروع حلقه ی آدرس دهی و رسم بر روی Page ها:

```
  for(Page=0;Page<8;Page++){
```

آدرس دهی Page و اولین ستون آن:

```
    WriteCmdGlcd(Page|0xB8);
    WriteCmdGlcd(0x40);
```

ارسال 64 دیتا به Page انتخاب شده:

```
    for(X=0;X<64;X++) WriteDataGlcd(Color);
   }
  }
}
```

تابع نمایش تصویر:

```
//ShowPicture ----------------------------------------------------------------------------------------------------------------------------------Ex
void ShowPicture(const rom unsigned char *pic){

  unsigned char Page,X,R;
```

متغییر P نقش اشاره گر به درایه های آرایه ی عکس را دارد:

```
  unsigned int P;
```

این تابع همانند تابع قبل می باشد با این تفاوت که بجای ارسال دیتای ثابت 0 یا 255 اعداد دیتابیس عکس مورد نظر ارسال می شود:

شروع حلقه ی رسم که فقط دو بار اجرا می شود برای رسم بر روی سمت چپ و سپس سمت راست:

```
 for(R=0;R<2;R++) {
```

در صورت رسم در سمت چپ P برابر با 0 و سمت راست P برابر با 64 می شود تا در مراحل بعد به درستی به درایه های آرایه اشاره شود:

```
  if(R == 0) {
    P = 0;
    LeftSelect();
  }else {
    P = 64;
    RightSelect();
  }
```

شروع حلقه ی آدرس دهی و رسم بر روی Page ها:

```
  for(Page=0;Page<8;Page++){
```

آدرس دهی Page و اولین ستون آن:

```
    WriteCmdGlcd(Page|0xB8);
    WriteCmdGlcd(0x40);
```

اینبار به علت اینکه دیتای ارسالی ثابت نیست و بر اساس دیتابیس عکس می باشد باید در حلقه ی رسم بر Page با استفاده از متغییرP به درایه ی مورد نظر اشاره کرده و به LCD ارسال کنیم:

```
    for(X=0;X<64;X++) {

      WriteDataGlcd(pic[P]);
      P++;
    }
```

بعد از اتمام رسم بر روی Page مشخص شده متغییر P را با 64 جمع می کنیم تا پیج سمت مقابل را رد کرده به به پیج پایینی اشاره کند:

```
    P += 64;
  }
 }
}
```

تابع چاپ متن:

```
//PutString ----------------------------------------------------------------------------------------------------------------------------Ex
void PutString(unsigned char line,unsigned char X,unsigned char *text){

  unsigned int R,P=0;
```

عملیات چاپ متن هم تقریبا شبیه به چاپ عکس می باشد,به این صورت که اول از همه با توجه به مختصات انتخاب شده چیپ کنترلی سمت چپ یا راست انتخاب می شود و بعد از آدرس دهی رجیستر Y و انتخاب Page دیتای مربوط به کاراکتر ارسال می شود:

اینبار یک حلقه داریم که پایان آن به پایان رسیدن متنی است که قرار است رسم شود:

```
 while(text[P] != 0) {
```

این حلقه 8 بار تکرار می شود به این علت که فونت ما 8 در 8 می باشد:

```
 for(R=0;R<8;R++) {
```

چون متنی که قرار است چاپ شود ممکن است بخشی از آن در سمت چپ و بخشی دیگر در سمت راست باشد پس باید با توجه به مختصات فعلی چاپ سمت مورد نظر انتخاب شده و پس از آن ستون مورد نظر از آن سمت را آدرس دهی کنیم:

```
  if(X<64) {
   LeftSelect();
   WriteCmdGlcd(X|0x40);
  }else {
   RightSelect();
   WriteCmdGlcd((X-64)|0x40);
  }
```

با توجه به اینکه متن قرار آست در کدام سطر نوشته شود باید Page مورد نظر را انتخاب کرد:

```
  WriteCmdGlcd(line|0xB8);
```

کد اسکیی که P به آن اشاره دارد باید منهای 32 شود چون شروع دیتابیس فونت ما از کاراکتر Space می باشد و بعد از ضرب آن در 8 (به علت 8 در 8 بودن فونت) و جمع با R که اشاره به ستون مورد نظر از کاراکتر در دیتابیس می باشد در مجموع به درایه ی مورد نظر از دیتابیس می کنند که باید به LCD ارسال شود:

```
  WriteDataGlcd(Bascom8x8Font[(((unsigned int)text[P]-32)*8)+R]);
```

بعد از رسم دیتا اشاره گر LCD باید یک واحد جلو رفته تا دیتای بعدی چاپ شود و در همین راستا متغییر X که وظیفه ی اشاره به ستون مورد نظر را دارد یک واحد اضافه می شود:

```
  X++;
 }
```

بعد از رسم کاراکتر شماره ی انم باید سراغ کاراکتر بعدی برمیم و این کار را با یک واحد اضافه کردن به P که وظیفه ی اشاره به کاراکتر های متن را دارد انجام می دهیم:

```
 P++;
```

```
  }
}
```

تابع شروع به کار LCD وپیکر بندی پورت های مربوطه:

```
//InitGlcd ----------------------------------------------------------------------------------------------------------------------------------------Ex
void InitGlcd(void){
```

قبل از استفاده از LCD و فراخوانی توابع مربوطه این تابع که پورت ها را جهت می دهد و دیسپلی را ریست کرده و آماده به کار می کند باید اول برنامه یک بار فراخوانی شود:

```
  DirRS = 0;
  DirRW = 0;
  DirEN = 0;
  DirCS1 = 0;
  DirCS2 = 0;
  DirRST = 0;
  DirGlcdDataBus = 0x00;

  RS = 0;
  RW = 0;
  EN = 0;
  CS1 = 0;
  CS2 = 0;
  RST = 0;
  GlcdDataBus = 0x00;

  ResetGlcd();
  ClearGlcd(0);
}
```