به نام خدا

LCD کاراکتری

سطح مقدماتی

گردآوری

محمد یارمحمدی

کلمات کلیدی

LCD کاراکتری، *AVR* ، زبان سی، *Code vision AVR*

چکیده

در این مقاله مختصری مورد *LCD*های کاراکتری و نحوه ارتباط با آنها توسط میکرو کنترلر *AVR* آورده شده است.

۱. مقدمه

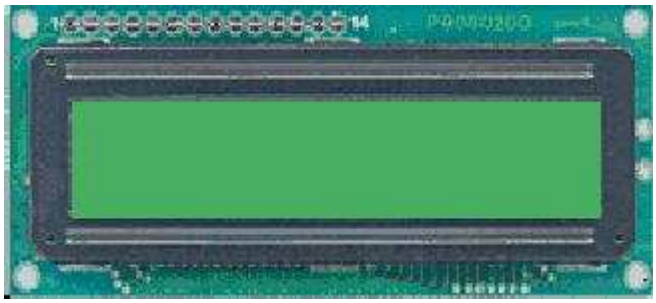
در بیشتر مدارهای میکروکنترلری ما نیاز به نمایش اطلاعات برای کاربر داریم،برای این کار راه های مختلفی وجود دارد از جمله استفاده از چند *LED*،استفاده از *7segment* ،استفاده از *LCD* و ...،اما *LCD* ها به علت داشتن قابلیتهای بیشتر نمایش،ابعاد مختلف و برنامه ریزی آسان اغلب مورد توجه قرار می گیرند.

LCD ها شامل انواع کاراکتری و گرافیکی هستند که در اینجا نوع کاراکتری آن را معرفی و طرز استفاده آن را شرح می دهیم.

LCD های کاراکتری نیز خود از نظر نحوه تبادل اطلاعات به دو دسته سریال و موازی تقسیم می شوند،از مزایای نوع سریال استفاده کمتر از پین های میکروکنترلر است.

از نظر اندازه،این *LCD* ها دارای تنوع مختلفی هستنداز جمله *16x2*،*32x2*،*40x2*،*16x4* و ... که منظور از این اعداد تعداد کاراکترهای قابل نمایش در یک سطر و تعداد سطرهای آن است،به عنوان مثال نوع *16x2* دارای ۲ سطر است و در هر سطر تعداد ۱۶ کاراکتر را نمایش می دهد.

ما در این مقاله در مورد نوع *16x2* که بسیار پرکاربرد است صحبت می کنیم با این توضیح که سایر انواع موازی نیز مشابه همین نوع هستند.



یک نمونه LCD از نوع 16x2

این نوع *LCD* ها بسته به اینکه دارای نور پشت زمینه باشند یا نه دارای ۱۴ یا ۱۶ پایه هستندکه در جدول زیر این پایه ها معرفی شده اند.

پایه	سمبول	*I/O*	توضیح
1	*VSS*	---	زمین منبع تغذیه
2	*VDD*	---	ولتاژ ۵ ولت منبع تغذیه
3	*VEE*	---	ولتاژ کنترل شدت نور صفحه
4	*RS*	*I*	اگر *RS=0* باشد رجیستر دستور انتخاب می شود اگر *RS=1* باشد رجیستر داده انتخاب می شود
5	*R/W*	---	*R/W=0* برای نوشتن اطلاعات *R/W=0* برای نوشتن اطلاعات
6	*E*	*I*	فعال ساز
7	*D0*	*I/O*	بیت ۰ باس داده
8	*D1*	*I/O*	بیت ۱ باس داده
9	*D2*	*I/O*	بیت ۲ باس داده
10	*D3*	*I/O*	بیت ۳ باس داده
11	*D4*	*I/O*	بیت ۴ باس داده

12	D5	I/O	بیت ۵ باس داده
13	D6	I/O	بیت ۶ باس داده
14	D7	I/O	بیت ۷ باس داده
15	BLA	---	آنود *LED* پشت زمینه *LCD*
16	BLK	---	کاتود *LED* پشت زمینه *LCD*

۲.توضیح مختصری راجع به پایه ها:

پایه *RS*: در داخل *LCD* دو رجیستر وجود دارد،که توسط پایه *RS* انتخاب می شوند.اگر *RS=0* شود رجیستر دستور (*Instruction Register*) انتخاب می شود تا فرمانهای مانند پیکره بندی *LCD*،پاک کردن آن،جابجایی مکان نما و ... برای *LCD* ارسال شود.

در صورتی که اگر *RS=1* شود،رجیستر داده (*Data Register*)انتخاب می شود،تا کاربر بتواند اطلاعاتی را که می خواهد روی *LCD* بنویسد برای *LCD* ارسال کند.

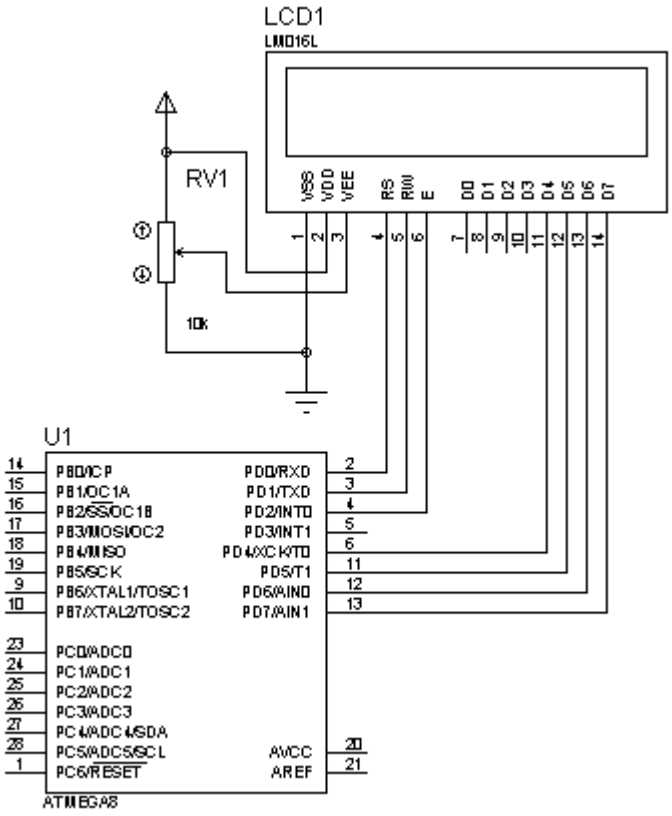
پایه *R/W*: به کمک این پایه کاربر مشخص می کند که می خواهد اطلاعات را روی *LCD* بنویسد یا از روی آن بخواند.اگر این پایه یک شود اطلاعات از روی *LCD* خوانده می شود و در صورتی که صفر شود اطلاعات روی آن نوشته می شود.

پایه *E*: اگر در این پایه پالسی از یک به صفر قرار داده شود،اطلاعاتی که روی پایه های *D0* تا *D7* قرار دارد درون یکی از رجیسترهایی که توسط پایه *RS* (*Register Select*) مشخص می شود،جای می گیرد.حداقل زمانی که این پایه باید صفر باشد ۴۵۰ نانو ثانیه است.

۳.نحوه اتصال LCD به میکروکنترلر:

با اینکه این نوع *LCD* های کاراکتری دارای ۸ پایه برای تبادل اطلاعات هستند(ارتباط ۸ بیتی) اما دارای قابلیت ارتباط با ۴ پایه را نیز دارند(ارتباط ۴ بیتی) که این امر باعث صرفه جویی در استفاده از پایه های میکروکنترلر می شود.

کتابخانه موجود برای *LCD* کاراکتری در نرم افزار *Codevision* نیز برای ارتباط ۴ بیتی نوشته شده است.



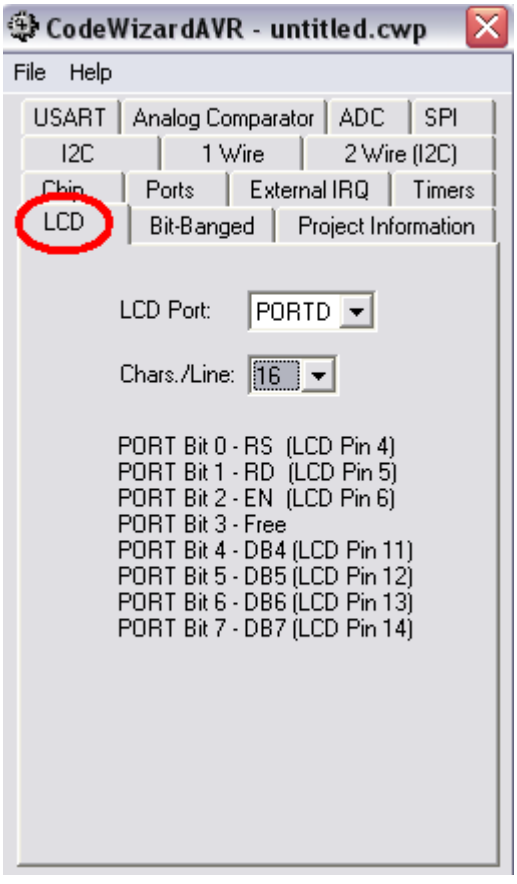
نحوه اتصال LCD به میکروکنترلر

البته بهتر است که پایه های آزاد *LCD* توسط ۴ عدد مقاومت به زمین متصل شوند.

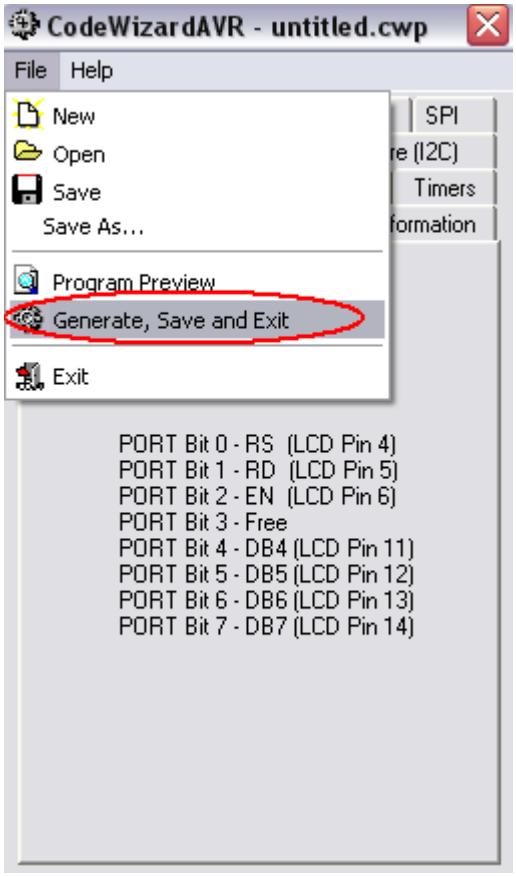
توجه: این نحوه اتصال با توجه به برنامه ای است که درون کتابخانه نرم افزار *Codevision* نوشته شده است.

۴.برنامه نویسی برای LCD:

آسان ترین روش برای پیکره بندی *LCD* در نرم افزار *Codevision* استفاده از *CodeWizard* آن است.بعد از اجرای برنامه *Codevision* با زدن کلیدهای *Shift+F2* منوی *CodeWizard* نمایش داده می شود.همانطور که در شکل زیر نمایش داده شده است،زبانه *LCD* را انتخاب کنید،سپس از قسمت مربوطه پورتی را که *LCD* با آن متصل است را مشخص کنید،همچینین نوع *LCD* (تعداد کاراکترها برای هر سطر) را مشخص کنید.همانطور که مشاهده می کنید نحوه اتصال پایه های *LCD* به میکروکنترلر در زیر این قسمت برای شما نمایش داده می شود.



سپس از منوی *File* گزینه *Generate,Save and Exit* انتخاب کنید(شکل زیر).



بعد از طی مراحلی که برای مشخص مردن محل ذخیره فایلهای برنامه است،محیط برنامه نویسی که شامل چند خط برنامه است ظاهر می شود،این برنامه ها در واقع همان تنظیماتی است که در *CodeWizard* انجام داده اید.

توجه کنید اگر از *CodeWizard* استفاده نمی کنید،باید قسمت های زیر را خودتان به برنامه اضافه کنید.

- مشخص کردن پورتی که *LCD* به آن متصل است و همچنین اضافه کردن کتابخانه *LCD*.این قسمتها باید در ابتدای برنامه اضافه شوند.

خطی از برنامه که بین دو عبارت *#asm* و *#endasm* آمده است یک دستور اسمبلی است و مشخص کننده پورتی است که *LCD* به آن متصل است،به عنوان مثال در شکل زیر *LCD* به پورت *D* متصل شده است و عدد *0x12* مشخص کننده آدرس این پورت است.

توجه: اگر میخواهید این پورت را تغییر بدهید باید حتما این آدرس را عوض کنید.

```
// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x12 ;PORTD
#endasm
#include <lcd.h>
```

نام پورت	آدرس پورت
PORTA	0x1b
PORTB	0x18
PORTC	0x15
PORTD	0x12

اگر از پورت دیگری استفاده میکنید آدرس آن را میتوانید از جدول مربوط به آدرس رجیسترها در دیتاشیت میکروکنترلر مورد نظر بیابید.

- پیکره بندی *LCD*:

برای پیکره بندی *LCD* در تابع *main* برنامه و قبل از عبارت *While(1)* این عبارت را اضافه کنید:

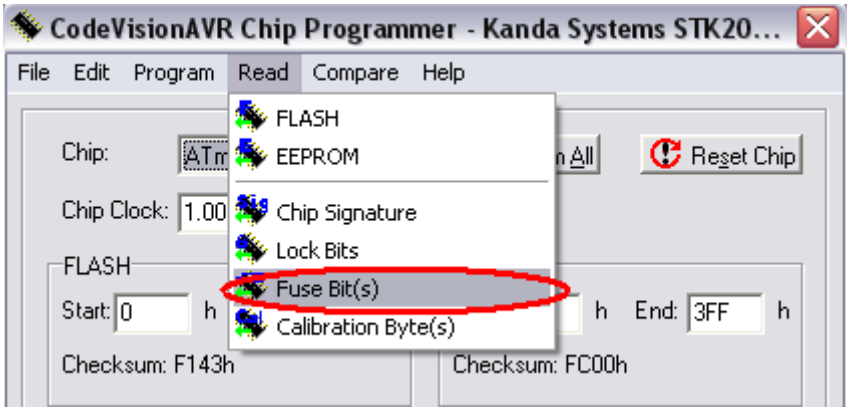
```
// LCD module initialization
lcd_init(16);
```

عدد ۱۶ به معنای این است که *LCD* دارای ۱۶ کاراکتر در هر سطر است،پس اگر از نوع دیگری استفاده میکنید این عدد را تغییر دهید.

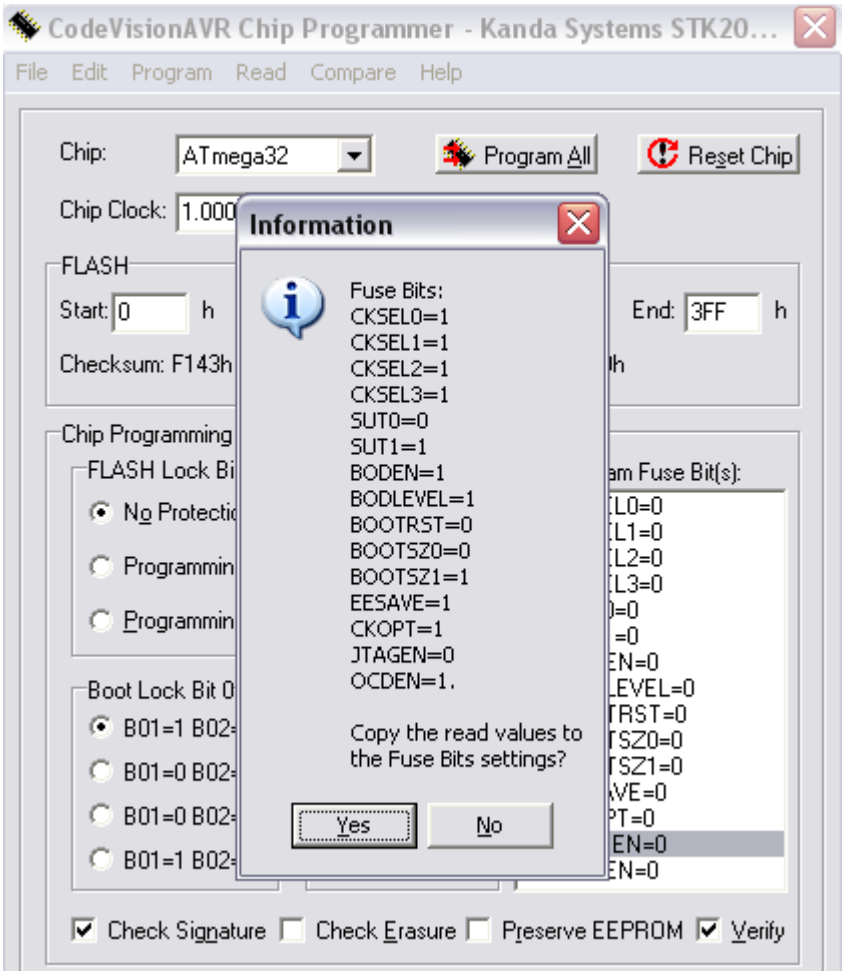
توجه: اگر *LCD* به پورت مشخص شده در اول برنامه متصل نباشد،میکروکنترلر کار نخواهد کرد!!! زیرا این تابع منتظر خواندن اطلاعات از *LCD* می ماند و چون *LCD* به آن متصل نیست میکروکنترلر بقیه کدهای برنامه را اجرا نمی کند.

توجه: ممکن است بعد از اتصال *LCD* به میکروکنترلر،با اینکه همه موارد به درستی رعایت شده است،*LCD* کار نکند!!! در نظر داشته باشید که بعضی از مدلهای میکروکنترلرهای *AVR* دارای قابلیتی به نام *JTAG* هستند. پایه های *TMS،TDO،TDI* و *TCK* مربوط به *JTAG* هستند.مثلا برای مدلهای *ATMEGA16* و *ATMEGA32* این پایه ها روی پورت *C* قرار دارند.برای اینکه بتوانید از این پایه ها به عنوان ورودی و خروجی استفاده کنید باید قابلیت *JTAG* را غیر فعال کنید.برای کار باید فیوزبیت *JTAGEN* را از حالت برنامه ریزی شده خارج کرد(برای این کار باید عدد یک را در آن نوشت).

برای اینکه فیوزبیت *JTAG* را غیر فعال کنیم اینتدا باید فیوز بیتها را بخوانیم.برای این کار ابتدا وارد محیط *Chip Programmer* می شویم(*Shift+F4*)،سپس از منوی *Read* گزینه *Fuse Bit(s)* را انتخاب می کنیم.

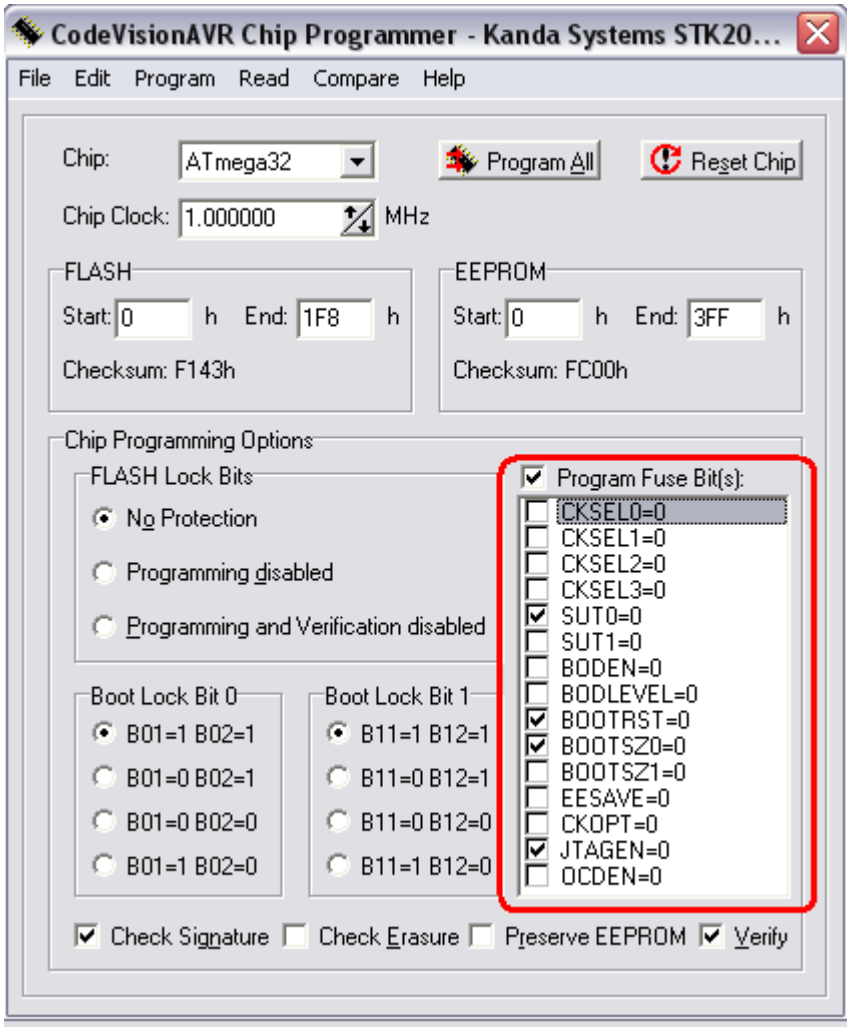


بعد از اینکه فیوزبیتها خوانده شوند،پیغام زیر نمایش داده می شود.

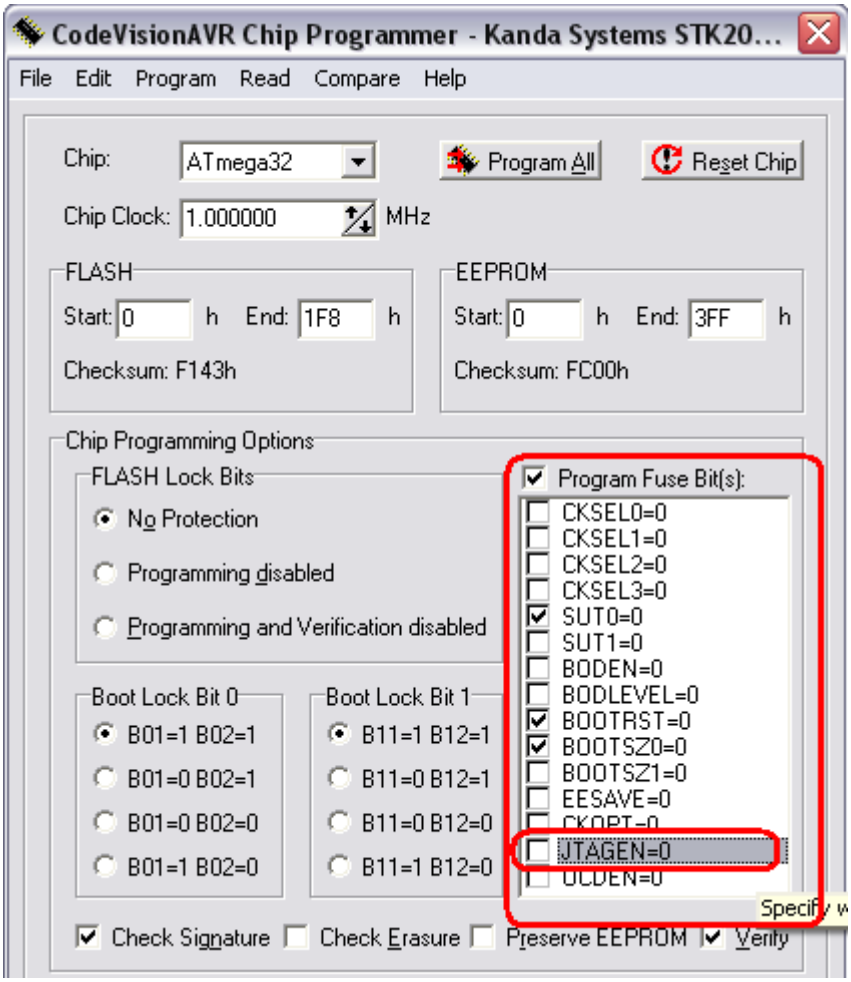


همانطور که در شکل بالا دیده می شود مقابل عبارت *JTAG* عدد صفر نوشته شده است،این بدان معنی است که قابلیت *JTAG* فعال است و نمی توان *LCD* را به پورتی که *JTAG* روی آن قرار دارد متصل کرد.

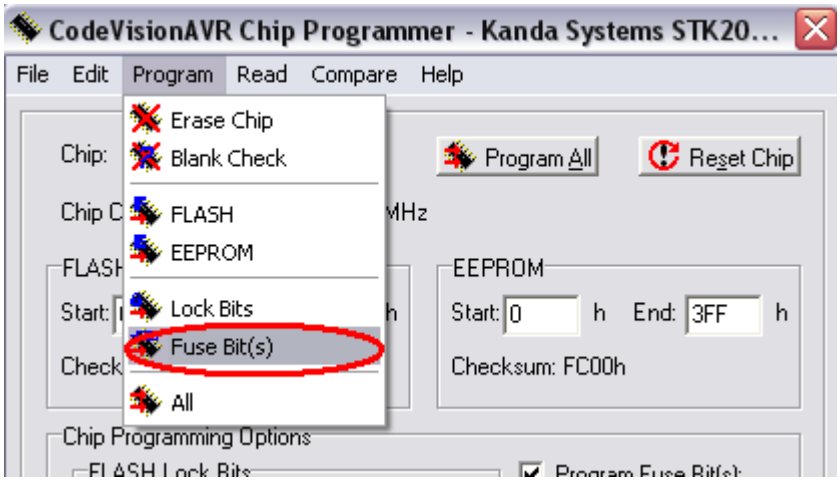
برای ادامه روی گزینه *Yes* کلیک می کنیم تا تغییرات خوانده شده را روی جدول فیوزبیتها در صفحه اصلی محیط *Chip Programmer* اعمال شود.در شکل زیر وضعیت فیوزبیتهای خوانده شده نمایش داده شده است.



اکنون با کلیک کردن بر روی علامت کنار گزینه *JTAG*این علامت را حذف می کنیم.



اکنون باید مقادیر جدید فیوز بیت ها را روی میکروکنترلر *Program* کنیم.برای این کار از منوی *Program* گزینه *Fuse Bit(s)* را انتخاب می کنیم.اکنون قابلیت *JTAG* غیر فعال شده است.



۵.توابع ارتباط با LCD در Codevision :

کتابخانه نوشته شده در *Codevision* برای ارتباط با *LCD* مدلهای زیر را پشتیبانی می کند:

1x8, 2x12, 3x12, 1x16, 2x16, 2x20, 4x20, 2x24 , 2x40

اما توابع این کتابخانه به دو دسته سطح پایین (*Low Level*) و سطح بالا (*High Level*) تقسیم می شوند.

توابع سطح پایین ارتباط با *LCD*:

void _lcd_ready(void)
این تابع منتظر می ماند تا *LCD* آماده دریافت اطلاعات شود.این تابع باید همیشه قبل از تابع *_lcd_write_data* فراخوانی شود.
void _lcd_write_data(unsigned char data)
این تابع محتویات *data* را درون رجیستر دستور *LCD* می نویسد.
void lcd_write_byte(unsigned char addr, unsigned char data);
این تابع محتویات *data* را در آدرس مشخص شده در *addr* درون حافظه *DDRAM*(*Display Data RAM*) یا *CGRAM* (*Character Generator RAM*) می نویسد.*
unsigned char lcd_read_byte(unsigned char addr);
این تابع محتویات آدرس مشخص شده در *addr* درون حافظه *DDRAM*(*Display Data RAM*) یا *CGRAM* (*Character Generator RAM*) را می خواند.*

*با این حافظه ها در قسمت پیشرفته بیشتر آشنا می شوید.

۶.توابع سطح بالا ارتباط با LCD:

unsigned char lcd_init(unsigned char lcd_columns)
این تابع *LCD* را پیکره بندی می کند،صفحه نمایش را پاک کرده و مکان نما را در سطر صفر و ستون صفر قرار میدهد،مقدار *lcd_columns* تعداد کاراکترهایی را که *LCD* در یک سطر نمایش میدهد را مشخص می کند. این تابع در صورتی که *LCD* وجود داشته باشد مقدار ۱ را بر میگرداند.
void lcd_clear(void)
این تابع صفحه نمایش را پاک کرده و مکان نما را در سطر صفر و ستون صفر قرار میدهد.
void lcd_gotoxy(unsigned char x, unsigned char y)
این تابع مکان نما را به ستون *x* و سطر *y* منتقل میکند. توجه کنید که شماره سطرها و ستون ها از صفر آغاز می شود.
void lcd_putchar(char c)
این تابع کاراکتر مشخص شده با متغییر *c* را در موقعیت فعلی مکان نما نمایش میدهد.
void lcd_puts(char *str)
این تابع رشته *str* ،که در حافظه *SRAM* وجود دارد را در موقعیت فعلی مکان نما نمایش می دهد.
void lcd_putsf(char flash *str)
این تابع رشته *str* ،که در حافظه *FLASH* وجود دارد را در موقعیت فعلی مکان نما نمایش می دهد.

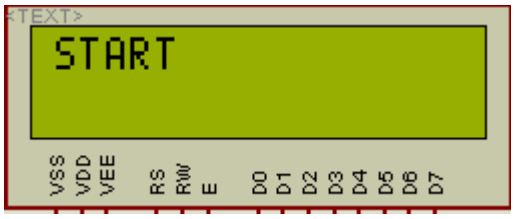
۷.نوشتن عبارات بر روی LCD

۷-۱.نمایش عبارات ثابت

منظور از عبارت ثابت،عبارتهایی است که در طول برنامه تغییر نمی کنند،به همین علت می توان این عبارتها را در حافظه *FLASH* ذخیره کرد(همانظور که می دانید حافظه *FLASH* را در طول اجرای برنامه نمی توان تعییر داد).با توجه به توضیحات توابع کتابخانه *Codevision* که در بالا ذکر شد برای نمایش این عبارات از دستور *lcd_putsf()* استفاده می کنیم.مثال:

```
lcd_clear();
lcd_putsf("START");
```

خروجی این دستورات بر روی *LCD* به صورت زیر است:



۷-۲.نمایش عبارات متغییر:

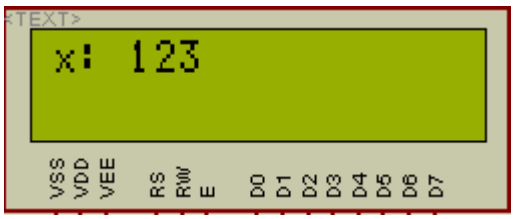
اگر بخواهید مقدار یک متغییر را بر روی *LCD* نمایش بدهید،چون این متغییر در حافظه *SRAM* جای دارد باید از دستور *lcd_puts()* استفاده کنید.

اگر متغییر از نوع صحیح باشد از دستورات زیر استفاده میکنیم:

```
char str[20];
int out;

lcd_clear();
sprintf(str,"x:%4u",out);
lcd_puts(str);
```

خروجی این دستورات به صورت زیر است:



توجه: برای استفاده از دستور *sprintf* باید کتابخانه توابع ورودی و خروجی استاندارد را به برنامه خود اضافه کنید.برای این کار عبارت زیر را به ابتدای برنامه خود اضافه کنید.

```
#include <stdio.h>
```

دستور *sprintf* متغییر *out* را به همراه یک رشته دلخواه درون رشته *str* قرار میدهد.سپس رشته حاصل توسط دستور *lcd_puts* روی *LCD* نمایش داده می شود.

عبارت *%4u* که درون آرگومان تابع *sprintf* آمده است،روی *LCD* نمایش داده نمی شود.حرف *u* نشان می دهد که خروجی باید به صورت یک عدد دسیمال بدون علامت باشد،عدد ۴ نشان می دهد که

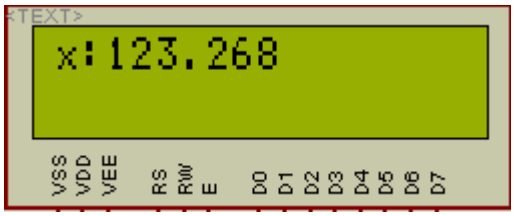
خروجی باید در یک فضایی ۴ تایی نمایش داده شود(با فرمت ۴ تایی به صورت *xxxx*)علامت % نشان می دهد که عبارت بعد از آن برای نمایش نیست بلکه حاوی علامتهایی برای مشخص کردن نوع خروجی است.

اما برای نمایش متغییر ها از نوع *Float* از دستورات زیر استفاده می کنیم:

```
char str1[20],str2[20];
float out=123.268;

lcd_clear();
ftoa(out,3,str1);
sprintf(str2,"x:%6s",str1);
lcd_puts(str2);
```

خروجی این دستورات به صورت زیر است:

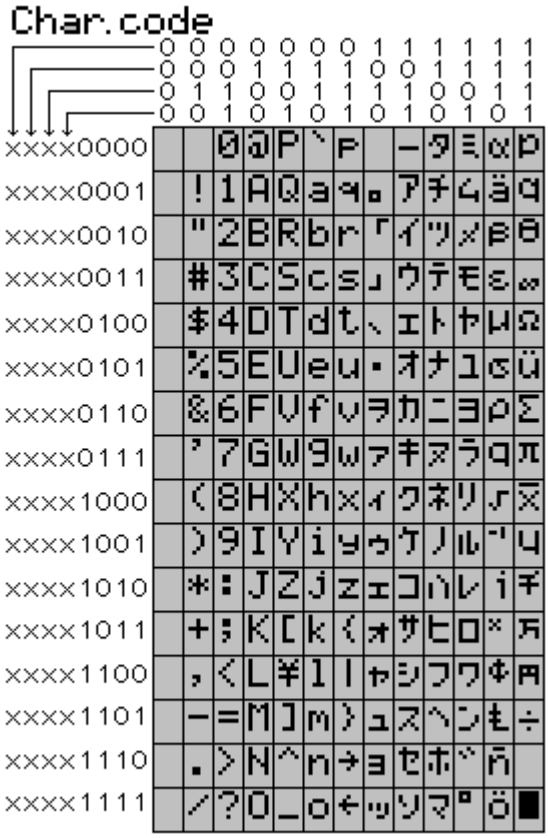


توجه: برای استفاده از دستور *ftoa* باید کتابخانه توابع استاندارد را به برنامه خود اضافه کنید.برای این کار عبارت زیر را به ابتدای برنامه خود اضافه کنید.

```
#include <stdlib.h>
```

۸.نمایش سایر کاراکترهای تعریف شده برای LCD

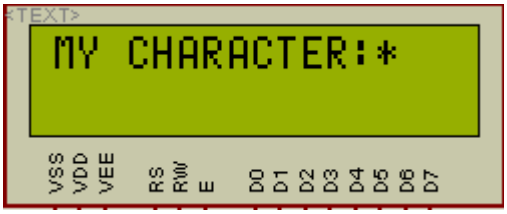
کاراکترهایی که از قبل برای *LCD* تعریف شده اند در جدول زیر آمده است.در این جدول برای هر کاراکتر یک کد ۸ بیتی اختصاص داده شده است،که ۴ بیت پرارزش آن برابر با ۴ بیت متناظر با ستون کاراکتر مورد نظر است و ۴ بیت کم ارزش آن متناظر با ۴ بیت سطر مورد نظر است.به عنوان مثال کد کاراکتر * برابر ۰۰۱۰۱۰۱۰ است،که برابر با *2A* هگزادسیمال است.



برای نمایش این کاراکترها از دستورات زیر استفاده می کنیم:

```
lcd_clear();
sprintf(str2,"MY CHARACTER:\x2A");
lcd_puts(str2);
```

خروجی این دستورات به صورت زیر است:

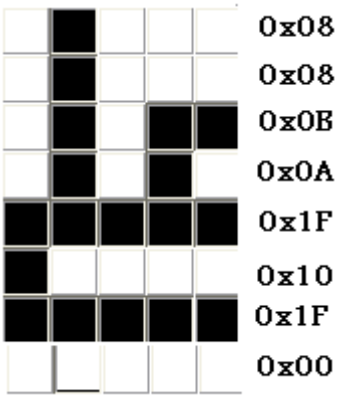


همانطور که مشاهدی می کنید برای نمایش این کاراکترها باید بعد از عبارت \x کد هگزادسیمال آنها را که از جدول بالا بدست می آید قرار دهیم.

۹.نمایش کاراکترهای تعریف شده توسط کاربر

کابر می تواند برای *LCD* یک کاراکتر جدید تعریف کند که در جدول قسمت قبل موجود نیست.

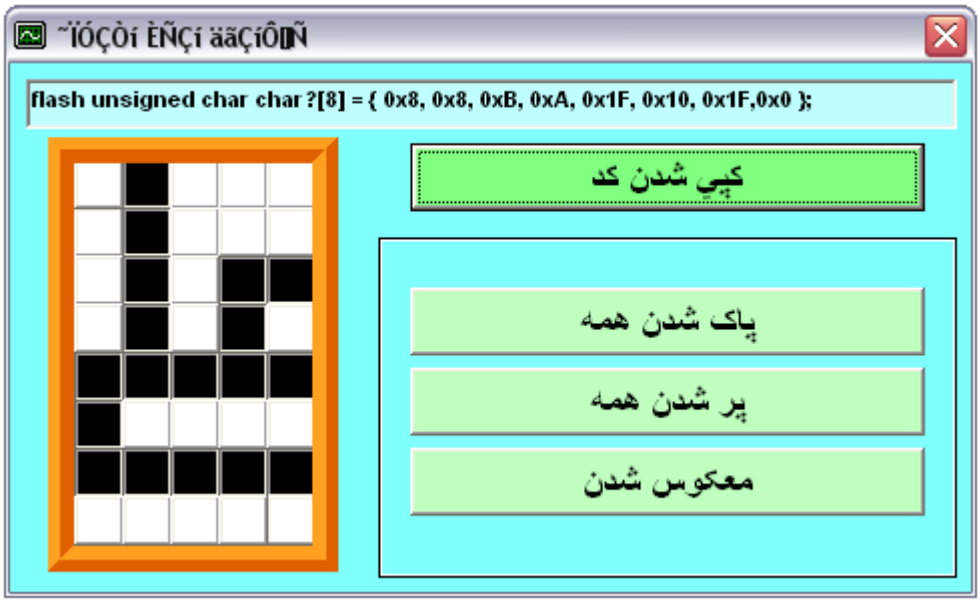
هر کاراکتر دارای *5x8* پیکسل است،برای تعریف کاراکتر جدید ابتدا کاراکتر را در یک مستطیل *5x8* به صورتی که در شکل نمایش داده شده تعریف می کنیم.سپس عدد هگزادسیمال مربوط به هر سطر را استخراج می کنیم.



اکنون به کمک این اعداد آرایه زیر را در حافظه *FLASH* ذخیره می کنیم.

```
flash unsigned char char0[8] = { 0x8, 0x8, 0xB, 0xA, 0x1F, 0x10, 0x1F,0x0 };
```

البته به کمک نرم افزار *LCD CHAR* می توان به راحتی عملیات بالا را انجام داد و کد مربوط را استخراج کرد.



بعد ساختن کاراکتر مورد نظر کافیست گزینه "کپی کردن کد" را انتخاب کنید،سپس در محیط برنامه نویسی آن را *Paste* کنید.

توجه کنید در هنگامی که کد را در محیط برنامه نویسی *Paste* می کنید،علامت ? را تبدیل به یک عدد دلخوا کنید.

روش دیگر تعریف این متغییر به صورت زیر است،که در این صورت در محیط برنامه نویسی هم می توانید به راحتی کاراکتر خود را تغییر دهید.

```
flash unsigned char char1[8]={

0b00001000,
0b00001000,
0b00001011,
0b00001010,
0b00111111,
0b00010000,
0b00111110,
0b00001000};
```

اکنون تابع *define_char* را به صورت زیر تعریف می کنیم:

```
void define_char(unsigned char flash *pc,unsigned char char_code)
{
unsigned char i,a;
a=(char_code<<3) | 0x40;
for (i=0; i<8; i++) lcd_write_byte(a++,*pc++);
}
```

این تابع برای ذخیره کردن کاراکترهای تعریف شده در *LCD* استفاده می شود.

اکنون کاراکتر خود را به کمک این دستور درون *LCD* ذخیره می کنیم:

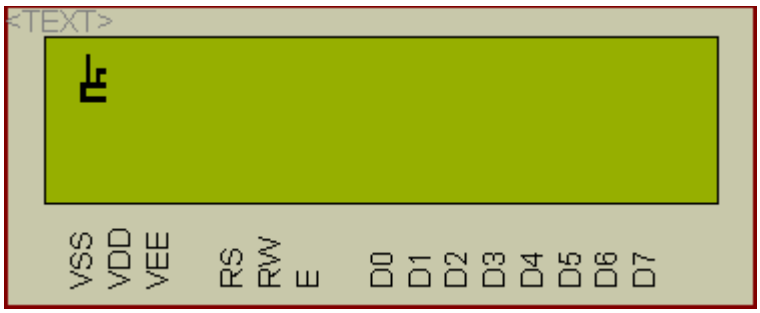
```
// LCD module initialization
lcd_init(16);

define_char(char0,0);

while (1)
      {
```

همانطور که ملاحظه می کنید این دستور بعد از دستور *lcd_init* و قبل از حلقه بی نهایت فراخوانی می شود.

خروجی بر روی *LCD* به صورت زیر است:



کاراکترهای دیگر را نیز می توان با تعریف متغییرهای دیگری در حافظه *FLASH* ساخت.

۱۰.فرمان دادن به LCD به کمک دستورات سطح پایین

به کمک دستورات سطح پایین کتابخانه نرم افزار *Codevision* می توانید دستورات زیر را به *LCD* ارسال کنید.

برای این کار از دستورات زیر استفاده می کنیم:

```
_lcd_ready();
_lcd_write_data(code);
```

که متغییر کد می تواند یکی از مقادیر زیر باشد(توجه کنید که مقادیر جدول زیر هگزادسیمال هستند):

عملکرد	کد هگزادسیمال فرمان
صفحه نمایش را پاک می کند	۱
مکان نما به محل اولیه برمی گردد	۲
مکان نما به چپ شیفت پیدا می کند	۴
مکان نما به راست شیفت پیدا می کند	۶
کاراکترها به راست شیفت پیدا می کند	۵
کاراکترها به چپ شیفت پیدا می کند	۷
کاراکترها خاموش و مکان نما خاموش می شود	۸
کاراکترها خاموش و مکان نما روشن می شود	*A*

کاراکترها روشن و مکان نما خاموش می شود	*C*
کاراکترها روشن و مکان نما روشن می شود	*E*
کاراکترها خاموش و مکان نما چشمک زن می شود	*F*
مکان نما به چپ شیفت پیدا می کند	*10*
مکان نما به راست شیفت پیدا می کند	*14*
کل به چپ شیفت پیدا می کند	*18*
کل به راست شیفت پیدا می کند	*1C*

به عنوان مثال برای اینکه کل صفحه به راست شیفت پیدا کند،دستورات زیر را می نویسیم:

```
_lcd_ready();
_lcd_write_data(0x1c);
```

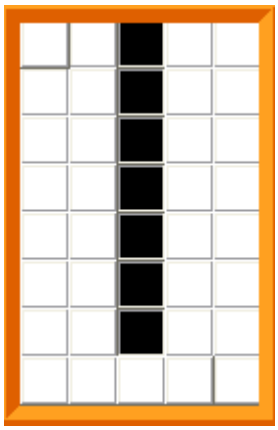
توجه:ممکن است این دستورات در برنامه *Proteus* به درستی عمل نکند،به همین علت بهتر است به صورت عملی اجرا شوند.

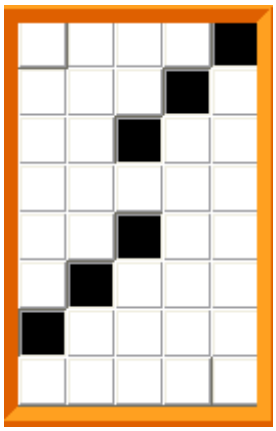
۱۱.ساختن انیمیشن های ساده

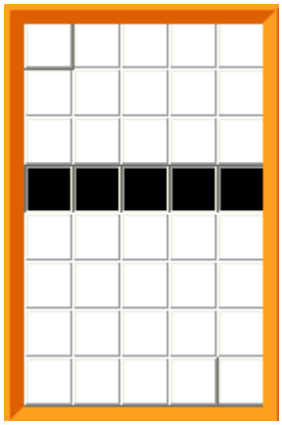
می توان به کمک دستورات بالا و نیز تعریف کاراکترهای دلخواه انیمیشن های ساده ای را ساخت و روی *LCD* نمایش داد.

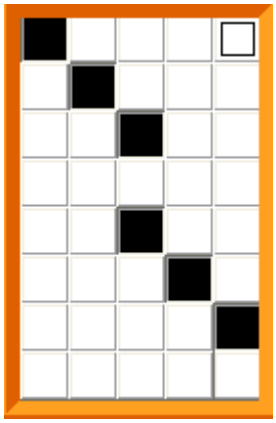
در اینجا به ذکر یک نمونه ساده از این انیمیشن ها به کمک کاراکترهای دلخواه می پردازیم،شما می توانید با استفاده از ایده های جالب خود نمونه های دیگری را بسازید.

ابتدا ۴ کاراکتر را به صورت زیر تعریف می کنیم:









کدهای مربوط به این کاراکترها به ترتیب به صورت زیر خواهد بود:

```
flash unsigned char char0[8] = { 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4,0x0 };
flash unsigned char char1[8] = { 0x1, 0x2, 0x4, 0x0, 0x4, 0x8, 0x10,0x0 };
flash unsigned char char2[8] = { 0x0, 0x0, 0x0, 0x1F, 0x0, 0x0, 0x0,0x0 };
flash unsigned char char3[8] = { 0x10, 0x8, 0x4, 0x0, 0x4, 0x2, 0x1,0x0 };
```

اکنون این کاراکترها را به کمک دستورات زیر درون حافظه *LCD* قرار می دهیم:

```
// LCD module initialization
lcd_init(16);

define_char(char0,0);
define_char(char1,1);
define_char(char2,2);
define_char(char3,3);

while (1)
```

سپس برنامه زیر را در حلقه بی نهایت اضافه می کنیم:

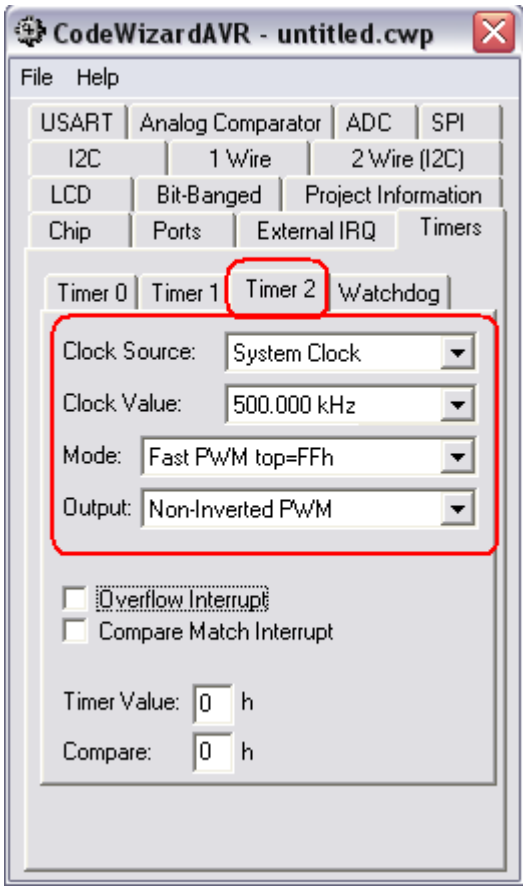
```
lcd_clear();
lcd_putchar(i);
i++;
if(i==3)
        i=0;
delay_ms(50);
```

نتیجه ای که روی *LCD* دیده می شود یک پاره خط کوچک است که حول نقطه میانی خود می چرخد.

کنترل شدت نور صفحه به صورت دیجیتال:

همانطور که قبلا ذکر شد به کمک پایه شماره ۳ و یک پتانسومتر می توانید شدت نور صفحه(کنتراست) را کنترل کنید،اما بیشتر وقتها مطلوبست که این تنظیمات توسط میکروکنترلر اعمال شود و بتوان آن را به صورت نرم افزاری تغییر داد.روشهای مختلفی برای انجام این کار وجود دارد اما یکی از ساده ترین راه ها استفاده از موج *PWM* است.این روش به سخت افزار زیادی نیز احتیاج ندارد.تنها کافیست که پایه سوم *LCD* را توسط یک مقاومت یک کیلو اهمی به یکی از پایه های *OCn* میکروکنترلر متصل کنید(به عنوان مثال اگر از تایمر ۲ برای تولید موج *PMW* استفاده می کنید پایه سوم *LCD* را به پایه *OC2* متصل کنید).

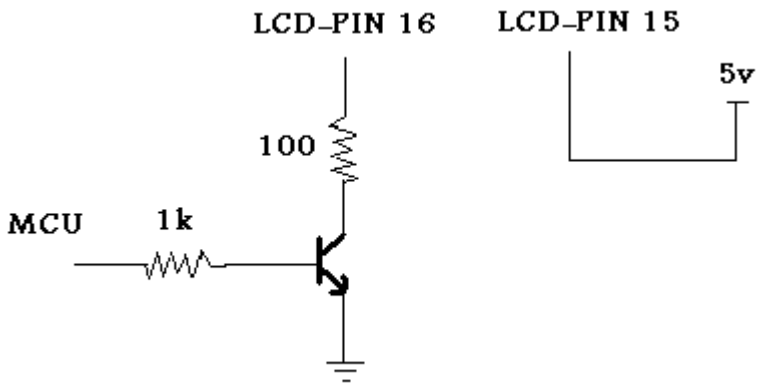
برای ساخت موج *PWM* تنظیمات زیر را در *CodeWizard* انجام دهید.



اکنون با تغییر مقدار رجیستر *OCR2* در درون برنامه می توانید شدت نور صفحه را تغییر دهید.

۱۲.کنترل نور پشت زمینه LCD

در مواقعی که به *LCD* نیازی نیست می توان نور پشت زمینه آن را خاموش کرد.برای این کار میتوان از مدار زیر استفاده کرد.



برای مدار بالا می توانید از یک ترانزیستور *C945* استفاده کنید.

کافیست قسمتی را که با کلمه *MCU* مشخص شده است را به یکی از پایه های میکروکنترلر متصل کرده و آن را به صورت خروجی تعریف کنید.سپس با صفر و یک کردن این پایه می توانید نور پشت زمینه *LCD* را کنترل کنید.

۱۳.عیب یابی مدار

اگر بعد از بستن مدار و پروگرام کردن میکروکنترلر موفق به راه اندازی *LCD* نشدید،موارد زیر را بررسی کنید.

- تغذیه میکروکنترلر و *LCD* را بررسی کنید.
- پایه کنتراست را بررسی کنید و مقدار پتانسیومتر را تغییر دهید.
- اتصال سیمهای بین میکروکنترلر و *LCD* را بررسی کنید.
- اگر *LCD* را به پورتی وصل کرده اید که پایه های مربوط به *JTAG* روی این پورت قرار دارند،*JTAG* را غیر فعال کنید(*PORTC* برای میکروکنترلرهای *ATMEGA16* و *ATMEGA32*).
- اطمینان حاصل کنید پورتی که *LCD* به آن متصل است همان پورتی است که در برنامه تعریف کرده اید.
- از کار کردن میکروکنترلر اطمینان حاصل کنید(می توانید با قرار دادن یک *LED* روی یکی از پایه های میکروکنترلر و خاموش و روشن کردن آن این کار را انجام دهید).
- یک برنامه خیلی ساده برای ارتباط با *LCD* بنویسید و از اجرا شدن آن اطمینان حاصل کنید.