1.چکیده : نوشتن کدی برای چراغ چشمک زن

روش آزمایش :

سورس کد چراغ چشمک زن :

```
#include <io.h>
#include <delay.h>

void main(void)
}
DDRC.4 = 1;
PORTC.4 = 0;
while(1)
}
 //Please write your application code here
PORTC.4 = ! PORTC.4;

delay_ms(700)

{
                                                                    {
```

---> در خط اول و دوم کتابخانه های لازمه تعریف گردیده
---> پورت 4 از پایه C به عنوان خروجی تعیین کردیم و مقدار اولیه آن را صفر گذاشتیم
---> در یک حلقه بی نهایت مقدار پورت را عوض میکنیم و در هر بار مقداری دیلی تعیین میکنیم
نیتجه گیری : کد به درستی کار میکند و چراغ روشن و خاموش میشود

2. ریست (Reset) یک وقفه است. می تواند توسط هر یک از منابع تحریک کننده آن رخ دهد. هنگامی که در میکرو کنترلر ریست رخ می دهد تمامی رجیستر های ورودی و خروجی و همچنین دیگر رجیسترهای کنترلی با توجه به مقادیر پیش فرض در نظر گرفته شده ، تنظیم می شوند. بعد از Reset یک مدت زمان کوتاه به اندازه زمان تعیین شده توسط start-upطول می کشد ،سپس بعد از پایداری نوسان ساز ،برنامه از بردار Reset شروع می شود. در میکرو کنترلر های خانواده avr و دیگر خانواده ها می توان به چندین روش ، میکرو کنترلر را Reset کرد. بعضی از آنها را می توان برنامه ریزی کرد. و بعضی دیگر را می توان با ایجاد شرایط خاص بیرونی در مدار الکترونیکی بوجود آورد. مقدار ولتاژ آستانه میکروکنترلر اولیه : هنگامی که ولتاژ تغذیه متصل به پایه VCC میکرو کنترلر کمتر از مقدار ولتاژ آستانه شود .میکرو کنترلر reset می شود.قرار دادن کلید ریست خروجی متصل به پایه ریست میکروکنترلر : اگر یک پالس با سطح صفر منطقی به مدت طولانی تر از سیکل پالس ساعت میکرو کنترلر ، به پایه خارجی Reset اعمال شود ، میکرو کنترلر ریست می شود. برای این منظور همانند تصویر زیر عمل می شود. باید توجه داشت برای اجرای برنامه در میکرو کنترلر پایه ریست با مقاومت بالاکش R1 به vcc مدار متصل گردد. همچنین بدلیل عدم وجود دیود داخلی در اکثر میکرو کنترلر ها در پایه ریست ، بهتر است از دیود خارجی جهت جلوگیری از اثر ناخواسته الکتریسیته ساکن استفاده شود. و نویزهای احتمالی محیط را نیز می توان توسط یک خازن عدسی خنثی کرد .با فشار کلید sw میکرو کنترلر Reset می شود. مقدار watchdog timer : اگر تایمر نگهبان (Watchdog Timer) فعال شود. این تایمر با تنظیمات برنامه ای فعال می شود. میکروکنترلر ریست می شود. این تایمر وظیفه عملکرد صحیح برنامه میکرو کنترلر را برعهده دارد.

3. زمانی که ممکن است پایه در وضعیت یک منطقی باشد یا صفر . علاوه بر عدم مشخص بودن وضعیت منطقی یک پایه در این حالت، امکان ورود نویز به مدارات داخلی میکروکنترلر نیز از این طریق وجود دارد. برای جلوگیری از این شرایط از مقاومت های Pull-up یا بالاکش یا مقاومت های **Pull-down** یا پایین کش استفاده می شود. مقاومت های بالاکش بین تغذیه مدار و پایه

میکروکنترلر وصل می شوند و مقاومت های پایین کش بین پایه میکروکنترلر و زمین وصل می شوند. با وجود شباهت هر دو، استفاده از مقاومت های بالاکش در مدارات رایج تر است. با توجه به انکه هر LED نسبت به جریان حساس بوده طبق رابطه می توان مقدار مقاومت را محاسبه کرد . با جایگذاری در فرمول مقاومت (به جای ولتاژ تفاضل ولتاژ منبع و ولتاژ led را میگذاریم) مقاومت 250 بدست می آید.

.4

کد خواسته شده سمت گیرنده به شرح زیر است :

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <string.h>
#include <avr/delay.h>
#define DDRB DDRB
#define DDRA DDRA
#define PORTB PORTB
#define PORTA PORTA
#define RS PC0
#define RW PC1
#define EN PC2
void com(unsigned char cmnd)
{
PORTA &= ~(1 << RS);
PORTA &= ~(1 << RW);
PORTA |= (1 << EN);
PORTA &= ~(1 << EN);
PORTB = cmnd;
_delay_ms(3);
}
void init(void)
{
DDRA = 0xFF;
DDRB = 0xFF;
_delay_ms(20);
com(0x38);
com(0x0C);
com(0x06);
com(0x01);
com(0x80);
}
void charlcd(unsigned char char_data)
{
PORTB = char_data;
PORTA |= (1 << RS);
PORTA &= ~(1 << RW);
```

```
PORTA |= (1 << EN);
PORTA &= ~(1 << EN);
}
void str(char *str)
{
for (int i = 0; str[i] != 0; i++)
charlcd(str[i]);
}
char content[200];
int main()
{
DDRC = 0;
_delay_ms(100);
while (PINC == 0xFF){}
for (int i = 0; i != 200; i++)
{
content[i] = PINC;
_delay_ms(1);
}
init();
int lce_length = 16;
for (int i = 0; i != 200 - lce_length; i++)
{
char lcd_text[lce_length];
strncpy(lcd_text, content + i, lce_length);
com(0x01);
com(0x80);
str(lcd_text);
_delay_ms(10);
}
return 0;
}
```

کد خواسته شده سمت فرستنده به شرح زیر است :

```
#define F_CPU 8000000UL /* Define CPU Frequency e.g. here 8MHz */
#include <string.h>
#include <avr/io.h>
#include <avr/delay.h>
#include <avr/eeprom.h>
char d[200], c[200] = "this is hesam talking to you ";
int main()
{
```

```
DDRB = 0xFF;
PORTB = 0xFF;
eeprom_busy_wait();
eeprom_write_block(c, 0, strlen(c)); //Write the content to
eeprom_read_block(d,0,strlen(c)); // Read the content from EEPROM
_delay_ms(100);
for (int i = 0; i != 200; i++)
{
PORTB = d[i];
_delay_ms(1);
}
PORTB = 0;
return 0;
}
```