micro by c language

البوابات (ports) :

البوابة : تتألف البوابة في كل مايكرو من ثمانية أقطاب وكل قطب في المايكرو له عدة أعمال :

1- أقطاب التغذية (POWER PINS) ACC – GND
2- أقطاب الدخل والخرج (I/O PINS)
3- محول تشابهي رقمي (ADC PINS)
4- أقطاب المؤقتات (TIMER PINS)
5- أقطاب المقارن (COMPARATOR PINS)
6- أقطاب النبضات المعدلة (PWM PINS)
7- أقطاب البرمجة (SPI PINS)
8- قطبين الكرستالة (OSC/RTC PINS)
9- قطبين الوصل التسلسلي مع الذواكر الخارجية (TWI PINS)
10- أقطاب الوصل التسلسلي مع الكمبيوتر (USART PINS)

تتألف كل بوابة من ثلاثة مسجلات (REGISTERS) :

1-المسجل DDRn : حيث n اسم البوابة المستخدمة ويستخدم لتحديد أقطاب المسجل إن كانت دخل أو خرج وذلك بإسناد قيمة للمسجل والمؤلف من بايت والقيمة إما أن تكون ثنائية أو عشرية أو سداسية عشر عندما نسند DDR B=0 عندها البوابة ستكون كلها دخل أما عندما نسند DDR B=255 فتكون البوابة كلها خرج أما في الثنائي DDR B=0B00000000 أو DDR B=0B11111111 والسداسي عشر DDR B=0X00 أو DDR B=0XFF ويسند مرة واحدة فقط

The Port B Data Direction Register – DDRB

Bit	7	6	5	4	3	2	1	0	
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

2-المسجل PORTn : يستخدم عندما تكون البوابة مستخدمة كخرج لوضع إما 0 أو 1 منطقي على البوابة أثناء عمل البرنامج لإظهارها على البوابة مباشرة وذلك بإسناد قيمة لها أيضا نفس الطريقة السابقة فتظهر على أقطاب البوابة PORT B=0X00 فتصبح الأقطاب كلها صفر منطقي PORT B=0XFF تصبح كلها واحدات وذلك عندما يكون البوابة كلها خرج أما إذا DDR B=0X0F نصف البوابة دخل والنصف الآخر خرج عندها نقوم بالإخراج كل قطب لوحده وذلك بالطريقة التالية PORT B.0=1 أو PORT B.0=0 لأنه في خانات الدخل لا نستطيع كتابة 1 أو 0 منطقي وهكذا الأقطاب الآخرة

The Port B Data Register – PORTB

Bit	7	6	5	4	3	2	1	0	
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

3-المسجل PINn : يستخدم عندما تكون البوابة مستخدمة كدخل وهنا أيضا يطبق إما صفر منطقي أو واحد منطقي لكن الأفضل تطبيق صفر منطقي لعدم حصول أي خطأ متوقع أو لسحب تيار غير متوقع من المايكرو ويستخدم هذا المسجل لاستخدام القطب كمفاتيح وهنا سيتم استخدام مفهوم مقاومة الرفع الداخلية أو الخارجية

The Port B Input Pins Address – PINB

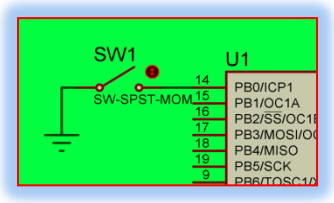
Bit	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Ex :

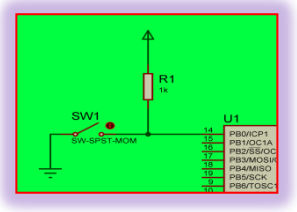
```
#include <mega8.h>                          مكتبية المايكرو المستخدم
#include <delay.h>                          مكتبية التأخير
void main()                                 التابع الرئيسي ودائما يبدأ البرنامج به
{
DDRB=0xff;                                  تهيئة البوابة كخرج
DDRD=0x00;                                  تهيئة البوابة كدخل
if (PIND.0==0)                              فحص إذا القطب يساوي صفر منطقي
PORTB=0x0f;                                 إسناد قيمة للبوابة
}
```

A – مقاومة الرفع الداخلية : وهنا يتم إسناد قيمة إلى المسجل PIN كالتالي PINB=0XFF أصبح البوابة كلها دخل لكن هذا لا يكفي DDRB=0X00 أيضا وهنا أيضا يجب إعطاء قيمة بدائية للبوابة وتدعى التهيئة وذلك بإسناد قيمة للبوابة PORTB=0XFF وذلك ليعلم المعالج أنه انتقل من الحالة الواحد المنطقي إلى الصفر

```
void main()
{
DDRB=0X00;                                  تهيئة البوابة كدخل
PINB=0x00;                                  إسناد لمسجل الدخل
PORTB=0Xff;                                 تهيئة البوابة كمقاومة رفع
}
```

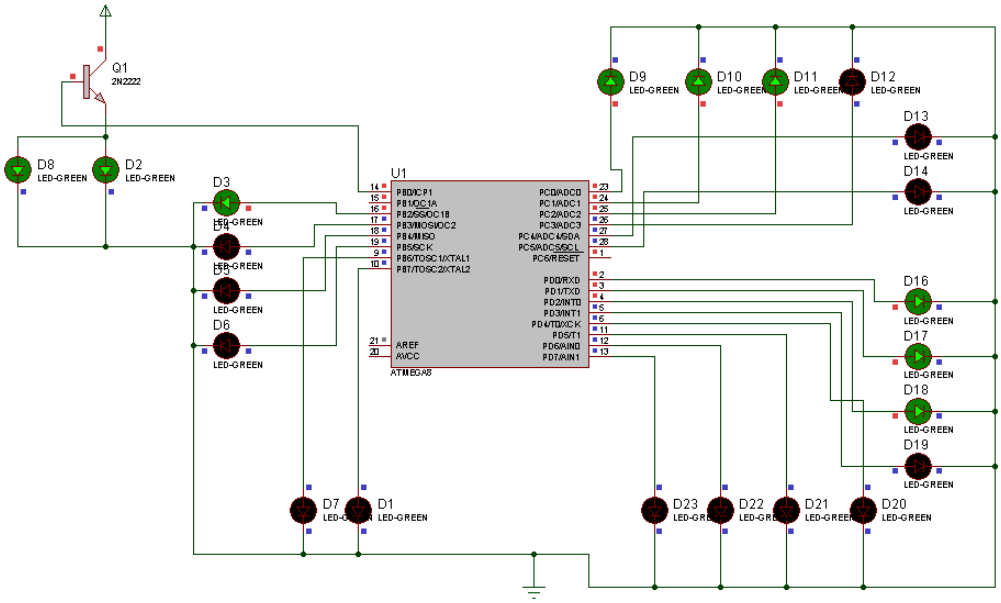


B – مقاومة الرفع الخارجية : وهنا يتم وصل مقاومة خارجية 1K موصولة بمنبع جهد ويتم وصل من القطب نفسه مع مفتاح ثم الأرضي حيث يتم الانتقال من الواحد المنطقي إلى الصفر ويحدث هنا فحص القطب



الأمثلة التطبيقية :

مثال 1 :

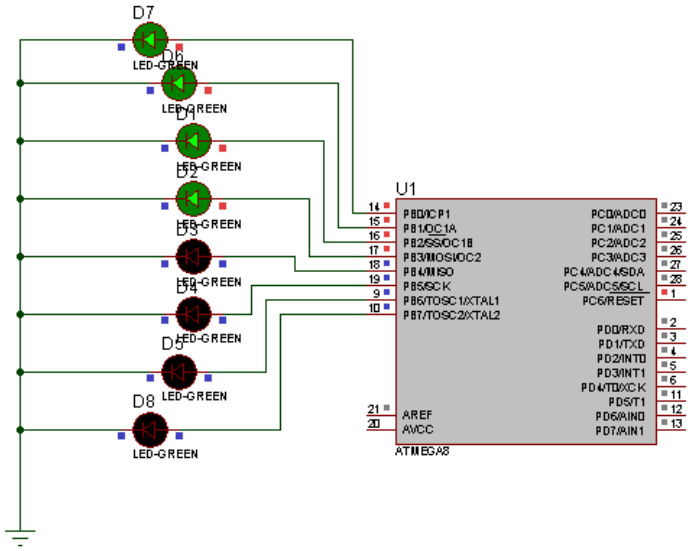


الكود

```
#include <MEGA8.h>

#include<delay.h>

void main()

{
```

```
int I,f ;
DDRB=0xff;
DDRC=0xff;
DDRD=0xff;
while (1)
    {
    PORTB=0xff;
     PORTC=0xff;
     PORTD=0xff;
    for(I=1;I<=8;++I)
    {
     PORTB=PORTB<<1;
     PORTC=PORTC<<1;
     PORTD=PORTD<<1;
     delay_ms(10);
     }
     PORTB=0xff;
     PORTC=0xff;
     PORTD=0xff;
     for(I=1;I<=8;++I)
    {
     PORTB=PORTB>>1;
     PORTC=PORTC>>1;
;PORTD=PORTD>>1
     delay_ms(10);
     }
    };
    }
```

مثال 2 :



الكود :

```
#include<mega8.h>
#include<delay.h>
void main()
{
int i=0;
DDRB=0XFF;
while(1)
{
PORTB=0XFF;
delay_ms(50);
for (i=0;i<4;i++)
{PORTB=PORTB<<2;        //إزاحة لليمين
delay_ms(50);}
PORTB=0XFF;
for (i=0;i<8;i++)
```

```
{
PORTB=PORTB>>1;        //إزاحة لليسار
delay_ms(50);}
}
}
```

مثال 4 :

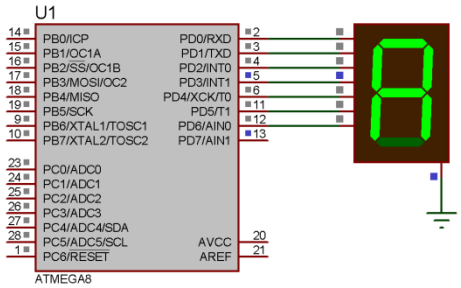
نفس الدارة السابقة لكن كافة البوابات :

الكود

```
#include <MEGA8.h>
#include<delay.h>
#include<math.h>
void main()
{
int I,f ;
DDRB=0xff;
DDRC=0xff;
DDRD=0xff;
while (1)
    {
    for(I=0;I<=7;++I)
    {
     f=pow(2,I);
     PORTB=f;
     PORTC=f;
     PORTD=f;
     delay_ms(100);
     }
    };
```

}

مثال 4 :



الكود :

```
#include <mega8.h>

#include <delay.h>

void main()

{

unsigned char
decode[16]={0x3f,0x06,0x5B,0x4f,0x66,0x6d,0x7c,0x07,0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71};

int i;

DDRD=0xff;

for(;;)

{

for(i=0;i<16;i++)

{

PORTD=decode[i];

delay_ms(100);

}}}
```

الذواكر (memories) :

تتألف الذواكر في المايكرو من عدة أنواع للتعامل مع البيانات وهذه الأنواع هي :

1 - ذاكرة (Flash program memory) : ويتم تخزين البرنامج فيها وهي عبارة عن ذاكرة وميضية تنظيمها البنيوي 2byte تستخدم لتخزين البرنامج ، وسبب هذا التنظيم أن معظم التعليمات تأخذ 2byte أو 4byte إن ديمومة المعلومات في هذه الذاكرة غالباً ما تكون 10 سنوات

2 - ذاكرة (EEPROM) : ويتم فيها تخزين البيانات أثناء عمل البرنامج

3 - ذاكرة (SRAM) : حيث يتم فيها تحميل البرنامج الرئيسي المخزن على الفلاش وهي مخصصة لتخزين البارامترات بشكل مؤقت وتفقد جميع المعلومات عند انقطاع التيار الكهربائي وكل موقع ذاكري هنا يكون بطول 8 بت

وتقسم إلى :

A-المسجلات العامة : وهي عبارة عن مسجلات وسيطية بين الدخل والخرج والمعالج، فعلى سبيل المثال إن أي عملية جمع أو طرح .... يكون المسجل العام طرف في هذه العملية.
B- مسجلات التحكم (الدخل والخرج): وهي المسجلات التي تحوي تعليمات التحكم، حيث أنها مخصصة لوظائف المعالج المحيطية كالعدادات والمؤقتات ووظائف دخل وخرج أخرى.وكل مسجل من هذه المسجلات يكون مرتبط بكيان كالمبدل التشابهي الرقمي مثلاً
الذاكرة SRAM (static Ram): وهي عبارة عن ذاكرة غير دائمة يستخدمها المعالج كمسودة عمل.
سميت هذه الذاكرة بـ static أي أن بنيتها الداخلية عبارة عن قلابات ، أما إذا قلنا dynamic فهذا يعني أن بنية الذاكرة الداخلية هي مكثفات وبالتالي فهي بحاجة لإنعاش كل فترة كهربائياً. تختلف الميكروات من ناحية حجم هذه الذاكرة.

1- تخزين المتحولات الصحيحة و المحرفية : يتم تخزين المتحولات العامة التي يتم تعريفها أثناء عمل البرنامج في المسجلات من R16 حتى R21 لكن ربما لا تكفي هذه المسجلات فقط لذلك يستعاض بمسجلات أخرى للتخزين فيها وهذه المسجلات حددتها الشركة للتعامل معها إلا سيحدث خطأ في تنفيذ البرنامج وهي ابتداءً من R4 وحتى المسجل R14 إذا كانت متحولات عامة و ابتداء من R16 وحتى المسجل R21 إذا كانت متحولات محلية والمتحولات العامة هي التي تتوضع بعد تتضمين المكتبيات أما العامة التي تتوضع ضمن توابع البرنامج

مسجل الأغراض العامة

	7	0	Addr.	
General Purpose Working Registers	R0		0x00	
	R1		0x01	
	R2		0x02	
	...			
	R13		0x0D	
	R14		0x0E	
	R15		0x0F	
	R16		0x10	
	R17		0x11	
	...			
	R26		0x1A	X-register Low Byte
	R27		0x1B	X-register High Byte
	R28		0x1C	Y-register Low Byte
	R29		0x1D	Y-register High Byte
	R30		0x1E	Z-register Low Byte
	R31		0x1F	Z-register High Byte

Ex :

تعريف متحولات إسناد قيمة إليهم

```
char a; int b=1;int c=19;
```

2- تعريف متحولات البيت : لا تأخذ حيز في الذاكرة لذلك يفضل استخدامها تخزن متحولات البت من البت الأول في المسجل R2 وحتى البت الأخير من المسجل R15

```
bit a=1;
bit g;   bit h=1;
```

3- تعريف متحولات المسجلات : وهي نفس المتحولات السابقة ولكنها تسبق بكلمة register

```
register int a=48;
register char b='0';
```

ملاحظة : يمكن التخزين بعنوان محدد

```
register int c @13;
```

4-أنواع المتحولات المستخدمة للتعريف :

Type	Size (Bits)	Range
Bit	1	0 , 1
Char	8	-128 to 127
unsigned char	8	0 to 255
signed char	8	-128 to 127
int	16	-32768 to 32767
short int	16	-32768 to 32767
unsigned int	16	0 to 65535
signed int	16	-32768 to 32767
long int	32	-2147483648 to 2147483647
unsigned long int	32	0 to 4294967295
signed long int	32	-2147483648 to 2147483647
float	32	±1.175e-38 to ±3.402e38
double	32	±1.175e-38 to ±3.402e38

5 - تخزين المتحولات أثناء عمل البرنامج : وهنا يتم تعريف متحولات من EEPROM والتي تحافظ على قيمتها حتى لو انقطع الكهرباء ويتم التعريف كالتالي

EX :

تعريف متحولات EEPROM

```
eeprom int a =0;
eeprom int b="ismael";
```

ولا ننسى هذه المتحولات يجب أن تسند بقيمة بدائية

6 – تخزين المصفوفات :

```
int days[7]={1,2,3,4,5,6,7};
char txt[6] = "micro";
```

التوابع (function) :

تستخدم التوابع لتبسيط عمل البرنامج أو في حالة تكرار أجزاء محددة من البرنامج كثيرا

1 – إنشاء التوابع : ويتم كما يلي :

وهذا النوع لا يرجع أي قيمة

نوع التابع اسم التابع متحولات التابع

```
void Ismael(int a,int b);
```

وهذا النوع يرجع قيمة من نوع int

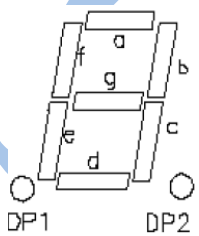
```
int Ismael(int a,int b);
{
int i;
.
.
.
return(i);
}
```

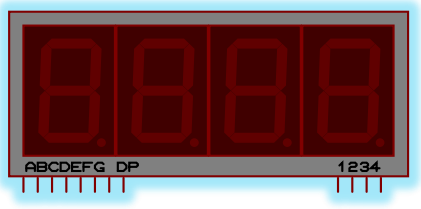
جسم التابع ويكتب كأي تابع عادي

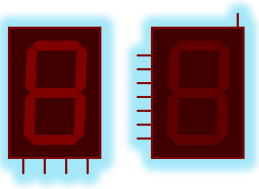
والتابع يمكن أن يعيد أو حسب نوعه إذا كان من نوع void فإنه لا يعيد شيء أما إذا كان من نوع int فإنه يعيد قيمة حصرا يجب أن تكون من نفس نوع التابع أغلب التوابع المستخدمة في المايكرو من نوع void ولا تحوي متحولات للتابع لأنه كل جزء يقوم التابع بإظهار شيء محدد وسيتم شرح كل تابع للسهولة لاحقا

شاشات الإظهار

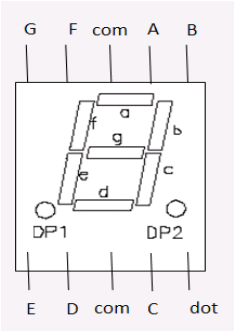
هي عبارة عن أداة الكترونية تحتوي على سبع قطع ضوئية (ليدات) متوضعة بشكل يسمح بإظهار جميع الأرقام الإنكليزية كما يمكن إظهار الأحرف الست عشرية , يوجد عدة أنواع من هذه الشاشات : بخانة واحدة أو بخانتين أو أربع , وهذه بعض الأشكال العملية لها:







وباختلاف نوع الليد المستخدم يختلف نوع الضوء الصادر عنها فمنها الأحمر و الأخضر و الأصفر و البرتقالي



ومهما اختلف اللون فإن البنية العامة لهذه القطع تكون بالشكل:

إن الترتيب السابق هو ترتيب عالمي مهما اختلفت الشركة الصانعة أو اختلف شكل القطعة.حيث يكون للخانة الواحدة عادة عشرة أقطاب متوضعة على الطرفين أو من الأعلى والأسفل، حيث تأخذ القطع السبعة 7 أقطاب و القطب الثامن يكون للنقطة DOT والقطبين الأخيرين هما تغذية (يوصل أحد قطبي التغذية).

أنواع شاشات الإظهار :

1 - شاشة الإظهار ذات المصاعد المشتركة (موجب مشترك) : يكون المشترك هو المصعد والباقي يغذى بصفر منطقي

Character	*	G	F	E	d	C	B	A	Hexa
==	Pd7	Pd6	Pd5	Pd4	Pd3	Pd2	Pd1	Pd0	==
	*	1	0	0	0	0	0	0	$40
	*	1	1	1	1	0	0	1	$79
	*	0	1	0	0	1	0	0	$24
	*	0	1	1	0	0	0	0	$30
	*	0	0	1	1	0	0	1	$19
	*	0	0	1	0	0	1	0	$12

	*	0	0	0	0	0	1	1	$03
	*	1	1	1	1	0	0	0	$78
	*	0	0	0	0	0	0	0	$00
	*	0	0	1	1	0	0	0	$18

2 - شاشة الإظهار ذات المهابط المشتركة (سالب مشترك) : يكون المشترك هو المهبط والباقي يغذى بواحد منطقي

Character	*	G	F	E	D	C	B	A	Hexa
==	Pd7	Pd6	Pd5	Pd4	Pd3	Pd2	Pd1	Pd0	==
	*	0	1	1	1	1	1	1	$3f
	*	0	0	0	0	1	1	0	$06
	*	1	0	1	1	0	1	1	$5b
	*	1	0	0	1	1	1	1	$4f
	*	1	1	0	0	1	1	0	$66
	*	1	1	0	1	1	0	1	$cd
	*	1	1	1	1	1	0	0	$7c

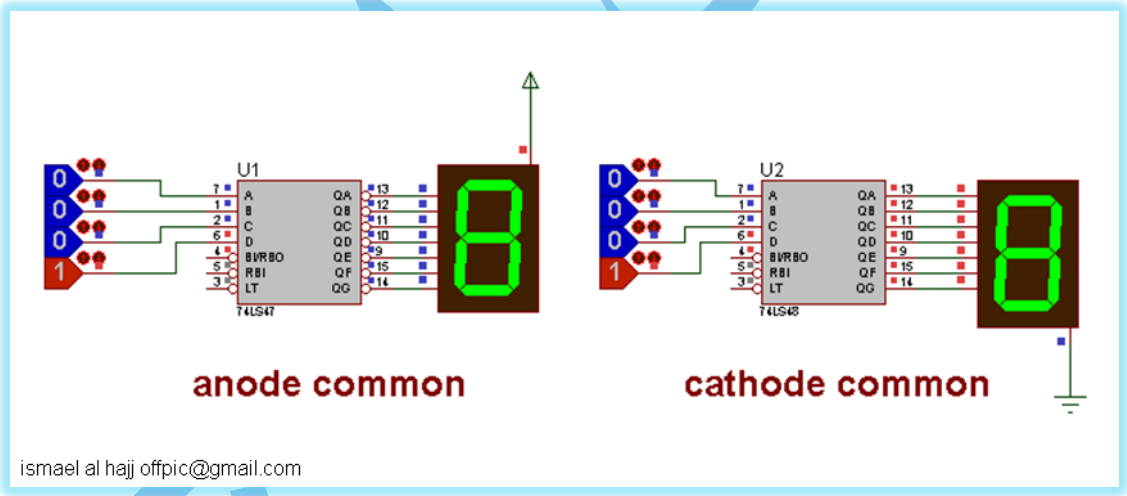
	*	0	0	0	0	1	1	1	$07
	*	1	1	1	1	1	1	1	$7f
	*	1	1	0	0	1	1	1	$67

قيادة شاشة الإظهار :

إن الأرقام من 0 .....9 وكذلك الأحرف الست عشرية 10....15 تعطى عادة في معظم الأنظمة الرقمية بالنظام الثنائي المرمز عشرياً BCD وبالتالي دارة القيادة المطلوبة هي عبارة عن دارة رقمية دخلها 4 أقطاب و خرجها 7 أقطاب .، يوجد في الأسواق نوعين من دارات القيادة تكون على شكل دارة متكاملة وظيفة هذه الدارات هي تحويل الرقم الثنائي المدخل إلى شيفرته المقابلة والتي تسمح بإظهار هذا الرقم على شاشة الإظهار وهي :

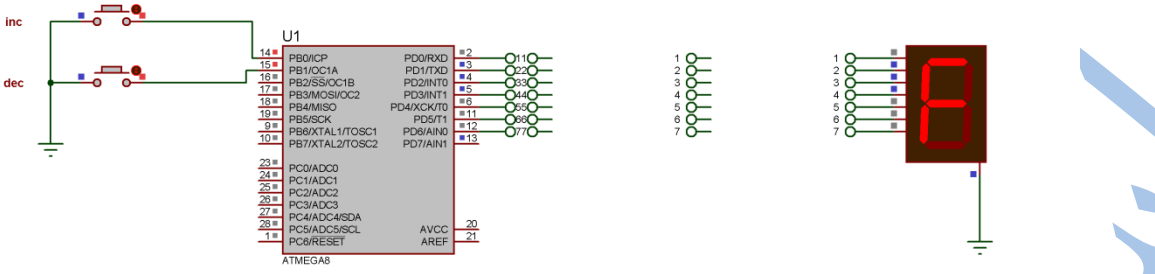
1 -7447: وهي خاصة بالتعامل مع شاشات الإظهار ذات المصاعد المشتركة .

2 - 7448: وهي خاصة بالتعامل مع شاشات الإظهار ذات المهابط المشتركة.



أمثلة تطبيقية :

مثال 1 :



الكود :

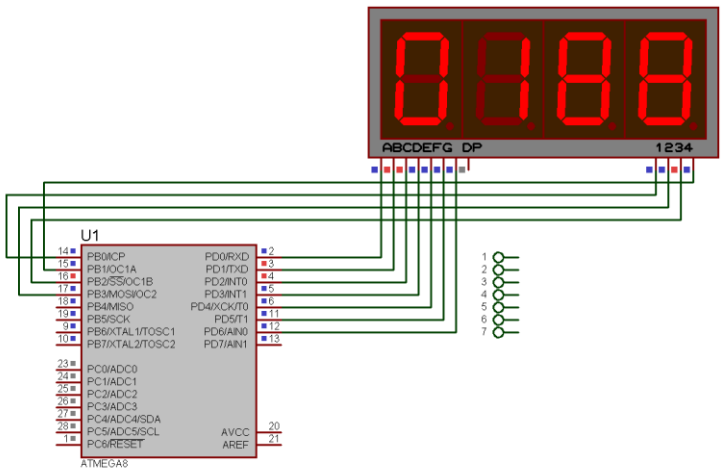
```
#include <mega8.h>

#include <delay.h>

void main()

{

unsigned char
decode[16]={0x3f,0x06,0x5B,0x4f,0x66,0x6d,0x7c,0x07,0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71};

int i=0;

DDRD=0xff;

PINB=0x03;

PORTB=0x03;

PORTD=decode[i];

for(;;)

{

if(PINB.0==0)

{i++;

if(i==16)

i=0;

PORTD=decode[i];

delay_ms(50);

}
```

```
if(PINB.1==0)
{
if(i==0)
i=16;
i--;
PORTD=decode[i];
delay_ms(50);
}
}}
```

طريقة المسح للإظهار :

وتتم عن طريق إشعال وإطفاء الليد الضوئي من 60 حتى 80 مرة بحيث العين لا ترى هذه التغيرات وتتم كالتالي على سبيل المثال كان لدنيا شاشتي seven segment من نوع مهبط مشترك فنقوم بوصل أرجل الشاشة مع بوابة من المايكرو وأخذ قطبين من البوابة فنقوم بإرسال صفر منطقي للأولى والثانية واحد منطقي وإرسال المعلومات على البوابة فيظهر الرقم على الشاشة الأولى ثم نرسل واحد منطقي للأولى والثانية صفر منطقي ونرسل المعلومات من للبوابة للشاشة وهكذا على السرعة لا نرى هذه التغيرات

مثال : عداد من صفر حتى 9999

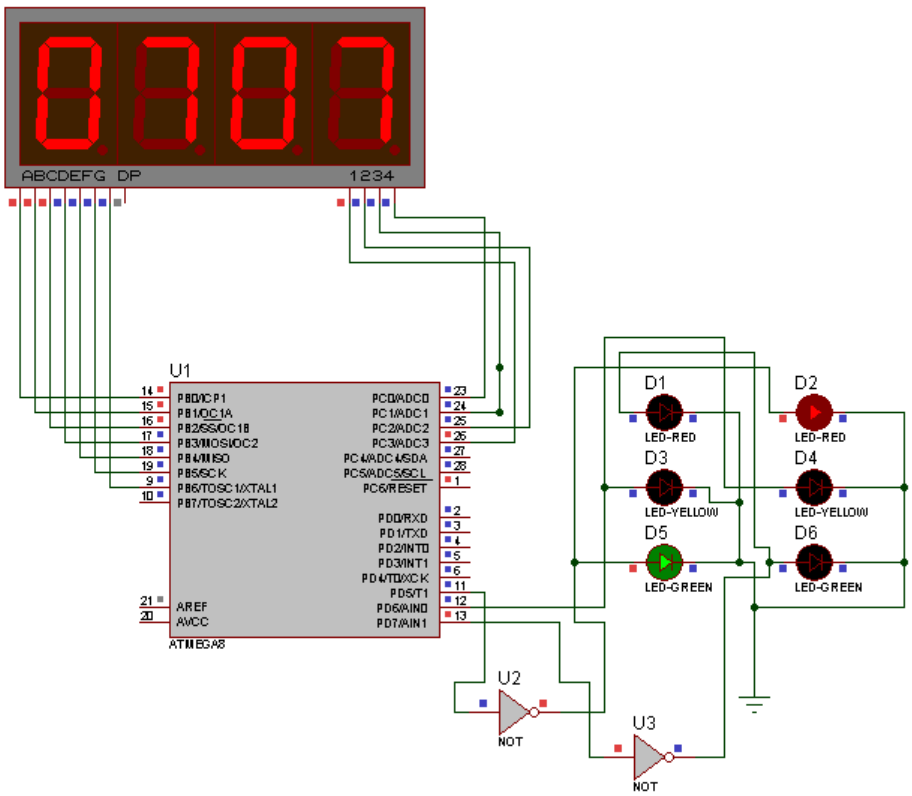


الكود :

```
#include <mega8.h>
```

```
#include <delay.h>
void main()
{
int i,j,a,b,c,d,e,f;
unsigned char decode[10]={0x3f,0x06,0x5B,0x4f,0x66,0x6d,0x7c,0x07,0x7f,0x6f};
DDRB=0xff;
DDRD=0xff;
for(i=0;i<9999;i++)
{
a=i/1000;
b=i%1000;
c=b/100;
d=b%100;
e=d/10;
f=d%10;
for(j=0;j<3;j++)
{
PORTB=0x01;
PORTD=decode[f];
delay_ms(1);
PORTB=0x02;
PORTD=decode[e];
delay_ms(1);
PORTB=0x04;
PORTD=decode[c];
delay_ms(1);
PORTB=0x08;
PORTD=decode[a];
delay_ms(1);
}}}
```

مثال 2 إشارة مرور



الكود

```
#include <mega8.h>
#include <delay.h>
int i;
void main()
{
int j,a,b;
unsigned char decode[10]={0x3f,0x06,0x5B,0x4f,0x66,0x6d,0x7c,0x07,0x7f,0x6f};
DDRB=0xff;
DDRC=0xff;
DDRD=0xff;
for(;;)
{
```

```
PORTD.5=1;
PORTD.7=0;
for(i=0;i<31;i++)
{
if(i==0)
PORTD.6=1;
if(i==1)
PORTD.6=0;
a=i/10;
b=i%10;
for(j=0;j<20;j++)
{
PORTC=0x01;
PORTB=decode[a];
delay_ms(1);
PORTC=0x02;
PORTB=decode[b];
delay_ms(1);
PORTC=0x04;
PORTB=decode[a];
delay_ms(1);
PORTC=0x08;
PORTB=decode[b];
delay_ms(1);
}}
i=0;
PORTD.5=0;
PORTD.6=0;
PORTD.7=1;
for(i=0;i<31;i++)
```

```
{
if(i==0)
PORTD.6=1;
if(i==1)
PORTD.6=0;
a=i/10;
b=i%10;
for(j=0;j<20;j++)
{
PORTC=0x01;
PORTB=decode[a];
delay_ms(1);
PORTC=0x02;
PORTB=decode[b];
delay_ms(1);
PORTC=0x04;
PORTB=decode[a];
delay_ms(1);
PORTC=0x08;
PORTB=decode[b];
delay_ms(1);
}}}
i=0;
}
```

المقاطعات (interrupts) :

هي عبارة عن أمر مفاجئ من أحد أقطاب المعالج أو داخل المعالج تسبب توقف عمل البرنامج فينفذ البرنامج الفرعي الخاص بالمقاطعة بعد الانتهاء من تنفيذها يعود للبرنامج من حيث توقف عنده المعالج وللمقاطعات عدة أنواع

1 -المقاطعات الداخلية : ويوجد لها عدة أنواع سنتعرف عليها لا حقا

2 –المقاطعات الخارجية : ويأتي من أحد أقطاب المعالج وهذه الإشارة عبارة إشارة صفر أو واحد منطقي وله عدة أشياء للتعرف على هذه الإشارة على أنها مقاطعة كي لا تعتبر كقطب دخل كما السابق

3 – مسجلات المقاطعة : تتألف من ثلاثة مسجلات أساسية ولكل مايكرو أسماء مسجلات محددة يتم التعرف عليها من خلال data sheet :

1 – المسجل SREG : مسجل الحالة ويستخدم لتفعيل المقاطعات كافة الداخلية والخارجية وهو البت السابع فقط ويسند إليه واحد منطقي ويتم تمكينه بالطريقة التالية إما SREG.7=1; أو عن طريق الاسمبلي ; #asm("sei") أو عدم تمكين SREG.7=0; أو عن طريق الاسمبلي ; #asm("cli")

The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – I: Global Interrupt Enable**

2 – المسجل GICR : مسجل التحكم ويستخدم لتفعيل أحدا من المقاطعتين أو تفعيلهما كلاهما INT1 – INT0 وهو البت السادس والسابع فقط , لتفعيل INT0 نسند GICR=40; لتفعيل INT1 نسند GICR=80 ; لتفعيل الاثنين GICR=C0;

General Interrupt Control Register – GICR

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	–	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

3 – المسجل MCUCR : مسجل الوحدة المركزية للمعالجة ويستخدم لتحديد نوع الإشارة التي سوف تفعل عندها المقاطعة وهناك عدة إشارات وتدعى جبهة وهي أربعة 1 - صاعدة 2 - هابطة 3 - مستوى الصفر 4 - متغيرة والبتات المحجوزة لها هي البت 0 و 1 للمقاطعة INT0 والبت 2 و 3 للمقاطعة INT1

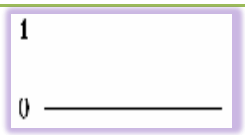
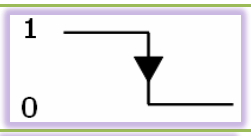
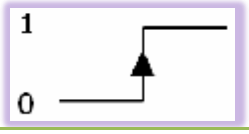
MCU Control Register – MCUCR

The MCU Control Register contains control bits for interrupt sense control and general MCU functions.

Bit	7	6	5	4	3	2	1	0	
	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

والجدول التالي يوضح القيم الواجب إسنادها لتفعيل إحدا الجبهات وهي للمقاطعة الثانية INT1

ISC11	ISC10	العمل

0	0		مستوى منخفض
0	1		أي تغير منطقي من الصفر إلى الواحد أو العكس
1	0		جبهة هابطة
1	1		جبهة صاعدة

1 - أولويات المقاطعة : وهي تنفيذ أكثر من مقاطعة في وقت واحد عندها تأتي أهمية أولويات المقاطعة وهو جدول لتسلسل المقاطعات الأهم فالأقل أهمية :

حيث يأخذ زر ال RESET المرتبة الأولى من الأولوية ثم INT0 ثم INT1 وهكذا الباقي

شعاع المقاطعة	عنوان برنامج الخدمة	مصدر المقاطعة	تعريف المقاطعة
1	$000	RESET	تصفير القطب الخارجي (خارجية) أو حدوث انخفاض بالجهد (خارجية) أو انتهاء دورة مؤقت المراقبة (داخلية)
2	$001	INT0	مقاطعة خارجية من النمط 0 (خارجية)
3	$002	INT 1	مقاطعة خارجية من النمط 1 (خارجية)
4	$003	TIMER COMP2	مقارنة نظير المؤقت/العداد 2 (داخلية)
5	$004	TIMER OVER2	طفحان المؤقت / العداد 2 (داخلية)
6	$005	TIMER CAPt1	مسك المؤقت / العداد 1 (خارجية)
7	$006	TIMER1 COMPA	مقارنة نظير المؤقت/العداد A1 (داخلية)
8	$007	TIMER1 COMPB	مقارنة نظير المؤقت/العداد B1 (داخلية)
9	$008	TIMER OVER1	طفحان المؤقت / العداد 1 (داخلية)
10	$009	TIMER OVER0	طفحان المؤقت / العداد0 (داخلية)
11	$00A	SPI , STC	مقاطعة اكتمال النقل التسلسلي
12	$00B	USART ,RXC	اكتمال استقبال وحدةUART (داخلية)
13	$00C	USART ,UDRE	فراغ مسجل معطيات UART (داخلية)
14	$00D	USART ,TXC	اكتمال ارسال وحدةUART (داخلية)
15	$00E	ADC	اكتمال التحويل التشابهي الرقمي (داخلية)
16	$00F	EE_RDY	مقاطعة جاهزية ذاكرة الـ EEPROM
17	$010	ANA - COMP	المقارن التشابهي (داخلية)
18	$011	TWI	ملائمة النقل التسلسلي ثنائي الكبل
19	$012	SPM_RDY	ذاكرة تخزين البرنامج جاهزة

2 - إستدعاء المقاطعة : وتستدعى بالطرق التالية :

```
interrupt [EXT_INT0] void ext_int0_isr(void)

interrupt [EXT_INT1] void ext_int1_isr(void)
```

interrupt [USART_RXC] void usart_rx_isr(void)

interrupt [USART_TXC] void usart_tx_isr(void)

interrupt [TIM1_OVF] void timer1_ovf_isr(void)

interrupt [TIM1_CAPT] void timer1_capt_isr(void)

interrupt [TIM2_OVF] void timer2_ovf_isr(void)

interrupt [TIM2_COMP] void timer2_comp_isr(void)

interrupt [ADC_INT] void adc_isr(void)

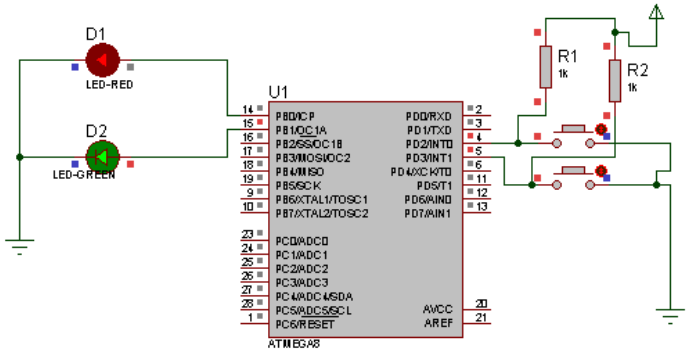
وهناك المزيد سنتعرف عليها لاحقا مجرد نظرة على هذه المقاطعات

ملاحظة :

لا يجوز استخدام مكتبية التأخير ضمن برامج المقاطعات لان تعليمة التأخير تتضمن عدم تأهيل المقاطعات ثم تأهيلها

أمثلة تطبيقية :

مثال 1 :

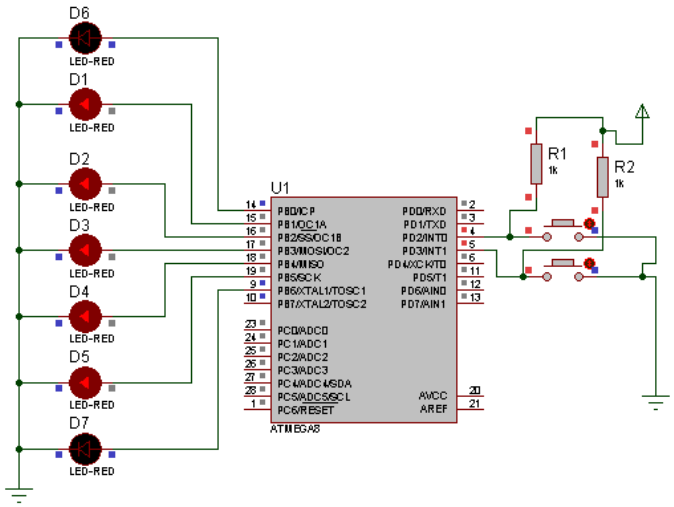


الكود : استدعاء المقاطعة الأولى عند الضغط على الزر الأعلى إضاءة الليد العلوي واستدعاء المقاطعة الثانية و إضاءة الليد السفلي

```
#include <mega8.h>

interrupt [EXT_INT0] void ext_int0_isr(void)

interrupt [EXT_INT1] void ext_int1_isr(void)


void main()
```

```
{


DDRB=0x03;

SREG.7=1;

GICR=0xc0;

MCUCR=0x0f;

GIFR=0xc0;

PORTB=0x00;

PORTB=0x03;

}

interrupt [EXT_INT0] void ext_int0_isr(void)

{

PORTB=1;

}

interrupt [EXT_INT1] void ext_int1_isr(void)

{

PORTB=2;

}
```

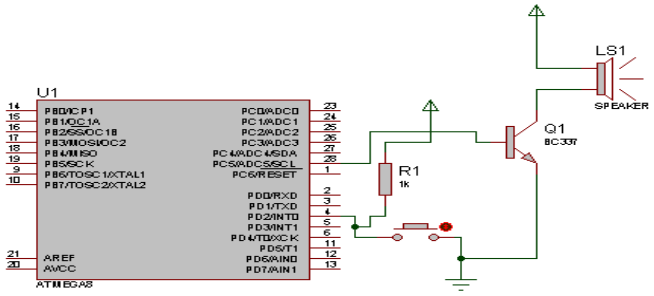
مثال 2 :



الكود :

```
#include <mega8.h>

interrupt [EXT_INT0] void ext_int0_isr(void)

interrupt [EXT_INT1] void ext_int1_isr(void)

void main()

{

DDRB=0xff;

SREG.7=1;

GICR=0xc0;

MCUCR=0x0f;

GIFR=0xc0;

PORTB=0xff;

}

interrupt [EXT_INT0] void ext_int0_isr(void)

{

PORTB=PORTB<<1;

}

interrupt [EXT_INT1] void ext_int1_isr(void)

{

PORTB=PORTB>>1;

}
```

مثال 4 : توليد صوت



الكود :

```
#include <mega8.h>
void delayms(int x);
interrupt [EXT_INT0] void toggle1();
void main()
{
DDRC=0xFF;
SREG.7=1;
GICR=0xc0;
MCUCR=0x0f;
GIFR=0xc0;
PORTB=0x03;
}
interrupt [EXT_INT0] void toggle1()
{
int i,j;
for(j=0;j<100;j++)
{
delayms(400);
for(i=0;i<50;i++)
{
PORTC.5=1;
delayms(1);
PORTC.5=0;
delayms(1);
}
}
}
void delayms(int x)
{
int a;
for(;x>0;x--)
for( a=560;a>0;a--)
```

```
{}
}
```

شاشة الإظهار الكريستالية (LCD) :

إن شاشة الإظهار الكريستالية LCD عبارة عن مصفوفة نقطية تستخدم لعرض المعلومات والنتائج حيث يمكن من خلالها عرض جميع رموز شيفرة الأسكي كما أنها تحوي على ذواكر تقسم إلى : DD-ram ذاكرة المعطيات و CG-ram ذاكرة مولد الرمز حيث تقوم هذه الذواكر بحفظ الرموز المراد إظهارها .

أنواع شاشات LCD:

تتوفر بنوعين :

1- CHARACTERS : أي محارف وهي التي سندرسها

2 - GRAPHICS: مثل شاشات الموبايل الأسود والأبيض أو الملون

ميزات شاشة LCD:

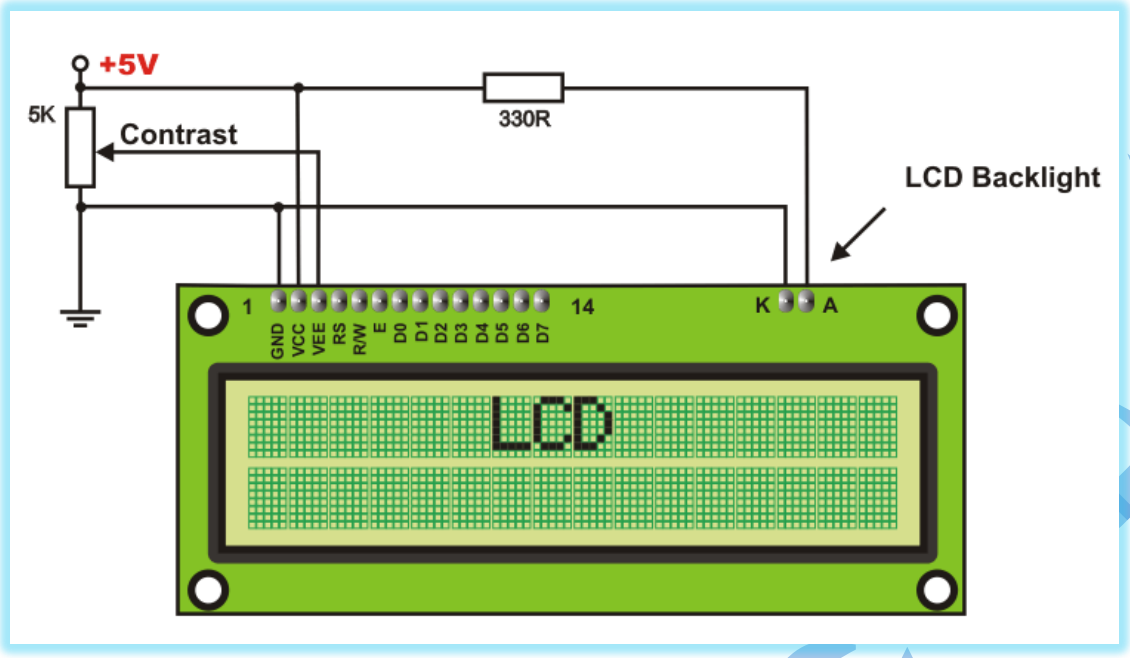
1- تغذيتها من 4.5 فولت إلى 5.5 فولت ، إذا تم تغذيتها بجهد أقل فإنها لا تعمل وإذا تم رفع الجهد فإنها تسخن لذلك يفضل تغذيتها بجهد 5 فولت تماماً

2- يمكن وصلها بطريقة 8 أقطاب (BUS) أو 4 أقطاب (PIN)

3- توجد بأحجام عديدة وأشهرها:

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

4- استهلاك قليل للطاقة.



1 - القطب VSS :

هو قطب التغذية لشاشة الـ LCD , وهو جهد الأرضي (0) منطقي (GND) .

2 - القطب VDD :

هو أيضاً قطب التغذية لشاشة الـ LCD , ولكن ذو القيمة (+5V) .

3 - القطب VEE :

ويدعى أيضاً بـ VO وهو قطب جهد التباين , ويقصد بالتباين هو حدة ظهور الرمز على الشاشة وأقل تباين أن لا نرى شيئاً على الشاشة يكون عند تطبيق (+5V) على هذا القطب , وأعلى تباين للشاشة يكون عند تطبيق (0 V) على هذا القطب .ويمكن التحكم بتباين الشاشة عن طريق وصل قطب التباين (VO) إلى مقاومة متغيرة .

4 - القطب RS :

وهو مسجل اختيار الدخل لشاشة الـ LCD وذلك في حال طبق عليه (0) منطقي عندها نريد إرسال كلمة تحكم , بينما في حال طبق (1) منطقي فعندها نريد إرسال معطيات إلى الشاشة , وفي الحالتين عن طريق أقطاب المعطيات .

5 - القطب R/W :

وهو للقراءة أو الكتابة إلى الشاشة , نطبق (0) منطقي على هذا القطب عندما نريد كتابة (إرسال) المعلومات إلى شاشة الـ LCD , ونطبق (1) منطقي عندما نريد قراءة (استقبال) المعلومات من شاشة الـ LCD , وعادةً ما يوصل هذا القطب إلى الأرضي في حال لم يكن هناك داعٍ للقراءة من الشاشة .

6 - القطب E :

وهو قطب تمكين شاشة الـ LCD , فكل معلومة يتم كتابتها أو قراءتها من شاشة الـ LCD يجب إرفاقها بنبضة تمكين على هذا القطب , ونبضة التمكين هذه تحدث عند الجبهة الهابطة , أي تتم هذه النبضة برفع القطب إلى الـ (1) منطقي وإنزاله إلى الـ (0) منطقي بعد تأخير مناسب .

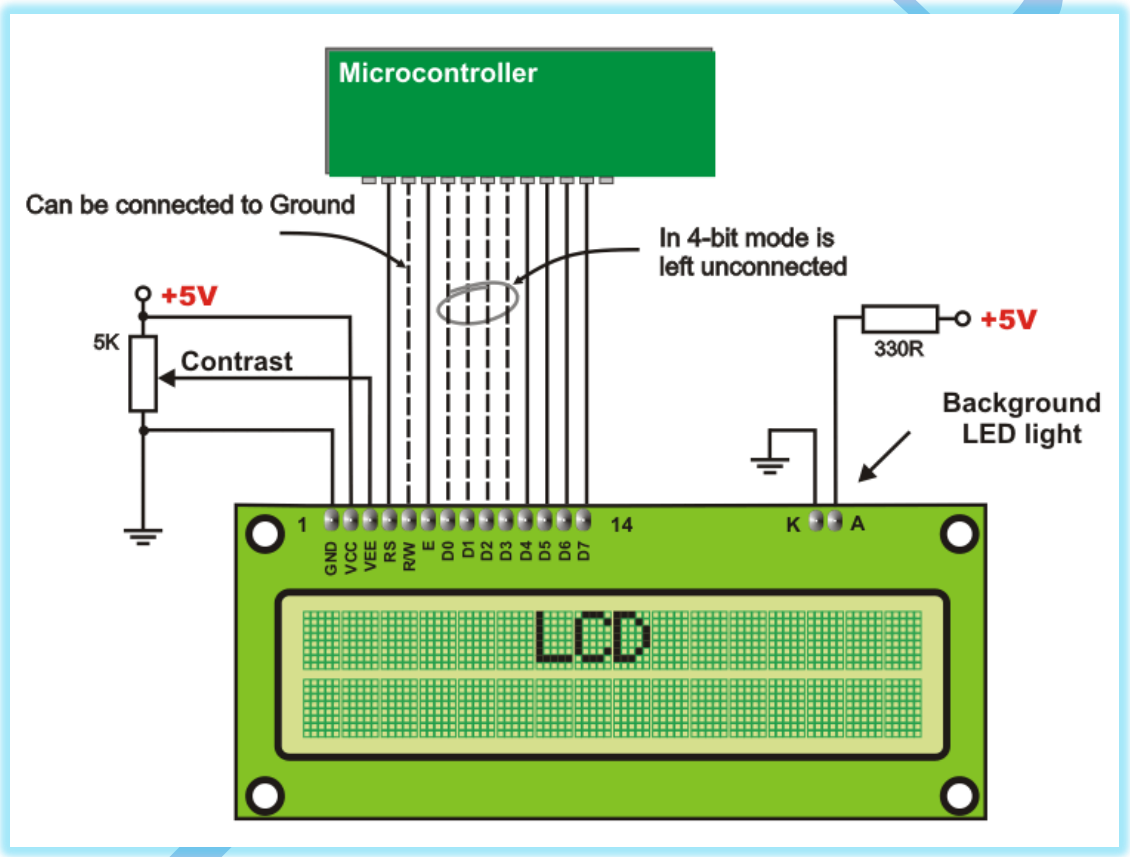
7 - الأقطاب DB0 . . . DB7 :

وهي أقطاب المعطيات (DATA) , حيث يتم كتابة المعطيات أو كلمات التحكم عبر هذه الخطوط إلى شاشة الـ LCD وكذلك قراءة المعطيات منها . . .

8 - القطبين K و A :

تزود بعض الشاشات بهذين القطبين , وهما على الترتيب قطبي المهبط و المصعد للّيد المسطّح المستطيل الموجود خلف شاشة الـ LCD , والذي يضاء عند تطبيق الـ (1) منطقي على المصعد (A) و الـ (0) منطقي على المهبط (K) .

وظيفة هذا الليد هو تأمين الإضاءة الكافية لشاشة الـ LCD ليتمكن المستخدم من رؤية العبارات المكتوبة عليها في ظلمة الليل . . .



تعليمات شاشة ال LCD

1 – التهيئة : ويتم قبل البدء بتضمين مكتبية أل lcd وذلك عبر السطر الأسمبلي كالتالي

```
#asm

.equ __lcd_port=0x15 ;

#endasm
```

ولكل بوابة رقم محدد بدل من 0x15

PORTA	0x1B
PORTB	0x18
PORTC	0x15
PORTD	0x12
PORTE	0x03
PORTF	0x11

تتضمين المكتبية

```
#include <lcd.h>
```

bit 6 = DB6	bit 4 = DB4	bit 2 = EN	bit 0 = RS
bit 7 = DB7	bit 5 = DB5	bit 3 = free	bit 1 = RD

أقطاب التوصيل حصرا توصل هكذا

وهذه المكتبية تدعم الأحجام التالية :

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

2 – تعليمات الكتابة على الشاشة :

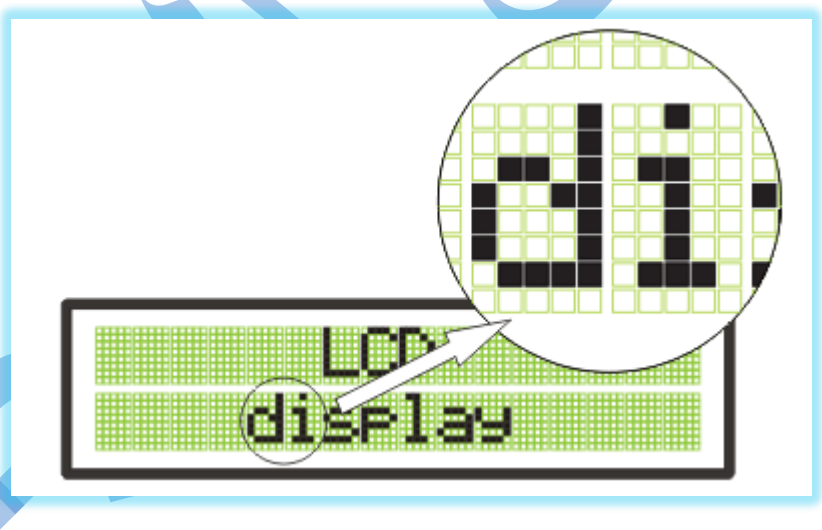
1 – التهيئة : وتستغرق بعض الوقت ولا يظهر أي شيء دون كتابتها

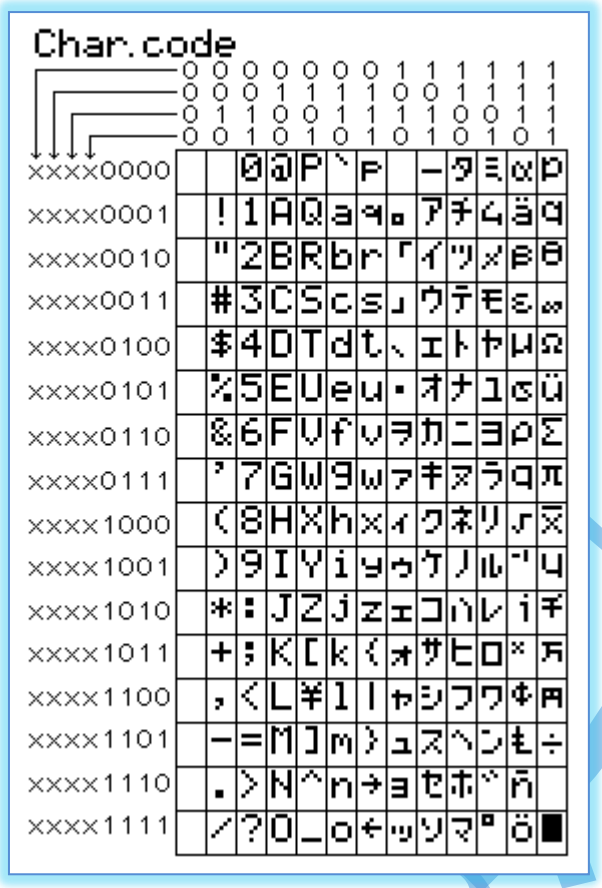
```
lcd_init(16);
```

2 – وضع المؤشر في مكان ما : تضع المحرف في أي مكان نريده

```
lcd_gotoxy(x,y);
```

حيث X هو رقم العمود للمحرف وY هو رقم السطر للمحرف ركز على كلمة محرف لأننا نتعامل مع المحارف حيث المحرف مؤلف 5*7 بيكسلات تشكل المحرف وهذه الأحرف الممكن إظهارها كما الشكلين





3 – الكتابة : ويتم كتابة الكلام بين إشاراتي التنصيص lcd_putsf("ismael al hajji ");

يمكن ضمن الإظهار بعض التنسيقات الخاصة بالكتابة \t تضع فراغ \n تقفز إلى السطر الذي بعده

4 – مسح الشاشة : ويتم مسح الشاشة كليا lcd_clear();

5 – إظهار محارف من جدول الأسكي : ويتم عبر التعليمة التالية lcd_putchar(0xff);

حيث وضع قيم الأسكي الموجودة في الجدول لإظهارها والمؤلفة من بايت مثلا لإظهار الرمز @ فنجد أنه

بالثنائي 01000000 أي بالسداسي عشر 0x40 lcd_putchar(0x40);

3 – إظهار الأرقام : ويتم بتضمين المكتبة #include<stdlib.h> والتي من خلالها نحصل على التابع الذي يحول الذي يحول الرقم إلى حرف

تعريف مؤشر من نوع char unsigned char *str;

تعريف متحول من نوع int وهو الرقم الذي سيحول char int i=1;

تعليمة التحويل من الرقم إلى char itoa(i,str);

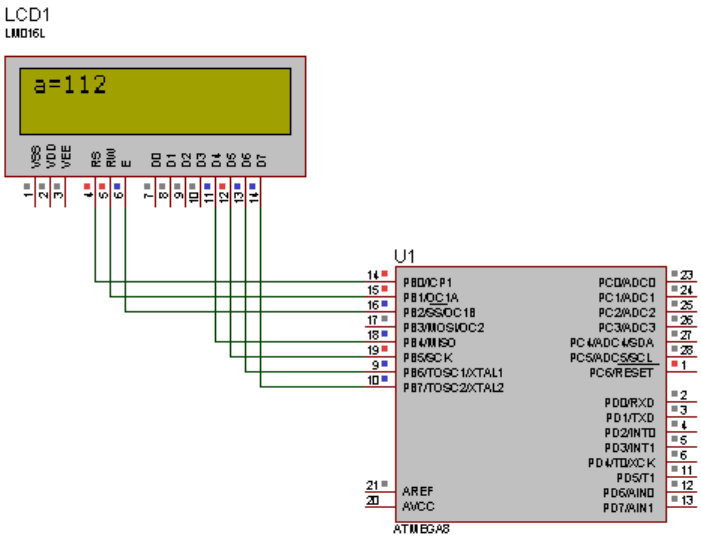
تعليمة الإظهار لإظهار المحرف المؤشر lcd_puts(str);

انتباه (attention) :

أثناء عملية الكتابة يجب وضع تأخير زمني كافي لرؤية النص الذي سيكتب في حال كتابة عدة جمل في نفس الوقت لكي لا يظن الإنسان أنها لا تعمل , أيضا تعليمة مسح الشاشة .

أمثلة تطبيقية :

مثال 1 : عداد من 0 حتى 999 أو أي رقم حسب البرمجة

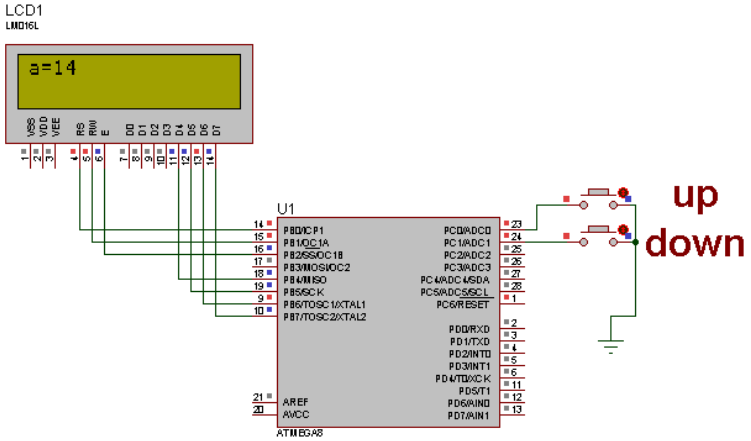


الكود :

```
#include <mega8.h>
#include<delay.h>
#include <stdlib.h>
#asm
  .equ __lcd_port=0x18;
#endasm
#include <lcd.h>
void main(void)
{
int i;
unsigned char *str;
lcd_init(16);
    for(i=0;i<999;i++)
    {
```

```
    lcd_putsf("a=");
    lcd_gotoxy(2,0);
    itoa(i,str);
    lcd_puts(str);
    delay_ms(10);
    lcd_clear();
    }
}
```

مثال 2 : عداد تصاعدي تنازلي

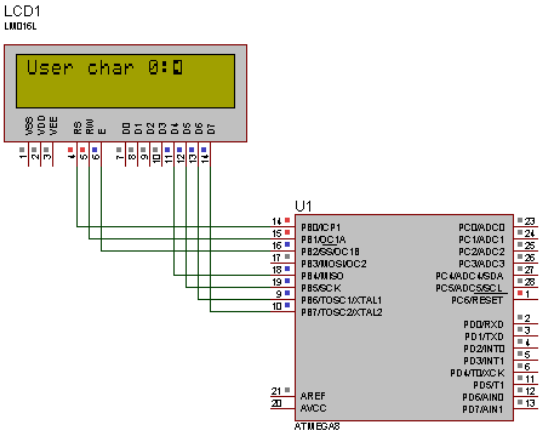



```
#include <mega8.h>
#include<delay.h>
#include <stdlib.h>
#asm
  .equ __lcd_port=0x18;
#endasm
#include <lcd.h>
int i;
unsigned char *str;
void disp();
void main()
{
```

```
lcd_init(16);
PINC=0x03;
PORTC=0x03;
lcd_putsf("     ismael");
delay_ms(500);
lcd_clear();
lcd_putsf("press up or down");
for(;;)
{
if(PINC.0==0)
{
lcd_clear();
i++;
disp();
delay_ms(50);
}
if(PINC.1==0)
{
lcd_clear();
i--;
disp();
delay_ms(50);
}
}}
void disp()
    {
    lcd_putsf("a=");
    lcd_gotoxy(2,0);
    itoa(i,str);
    lcd_puts(str);
```

}

مثال 3 : توليد المحرف الخاص رمز الصوت

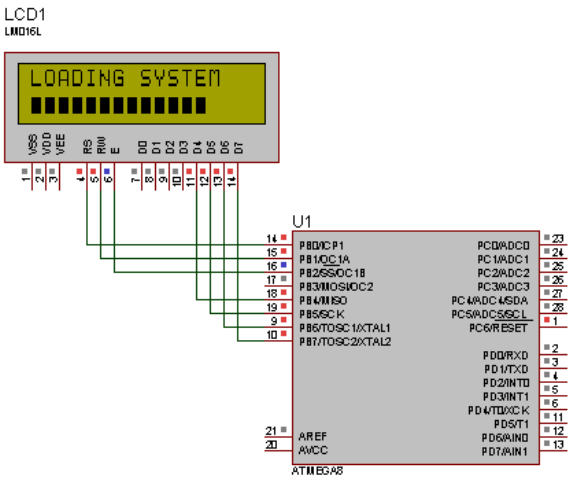


الكود :

```
#include<mega8.h>
#include<delay.h>
#asm
.equ __lcd_port=0x18 ;PORTC
#endasm
#include <lcd.h>
typedef unsigned char byte;
flash byte char0[8]={
0b1111111,
0b1111101,
0b1110001,
0b1110001,
0b1110001,
0b1111101,
0b1111111,
0b1111111};
void define_char(byte flash *pc,byte char_code)
{
```

```
byte i,a;
a=(char_code<<3) | 0x40;
for (i=0; i<8; i++) lcd_write_byte(a++,*pc++);
}
void main(void)
{
lcd_init(16);
_lcd_ready();    //enable cursor always
_lcd_write_data(0xe);    // enable cursor always
 delay_ms(500);
define_char(char0,0);
lcd_gotoxy(0,0);
lcd_putsf("User char 0:");
lcd_putchar(0);
while (1);
}
```

مثال 4 : رسالة تحميل نظام



الكود :

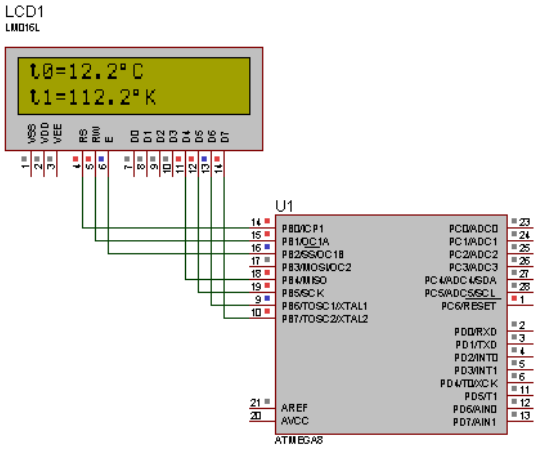
```
#include <mega8.h>
#include<delay.h>
#include <stdlib.h>
```

```
#asm
  .equ __lcd_port=0x18;
#endasm
#include <lcd.h>
void main(void)
{
int i;
lcd_init(16);
for(;;)
{
lcd_clear();
     lcd_putsf("LOADING SYSTEM");
    for(i=0;i<17;i++)
     {
     lcd_gotoxy(i,1);
     lcd_putchar(0xff);
     delay_ms(50);
     }
     }
}
```

التعليمة **sprintf() :** للإظهار مع تنسيقات على lcd ولا تقوم بالإظهار حيث يقوم بالإظهار lcd_puts

```
char display_buffer[33];
sprintf(display_buffer,"t0=%-i.%-u%cC\nt1=%-i.%-u%cK",12,abs(12%10),0xdf,12+100,abs((12+100)%10),0xdf);
lcd_puts(display_buffer);
```

مثال :

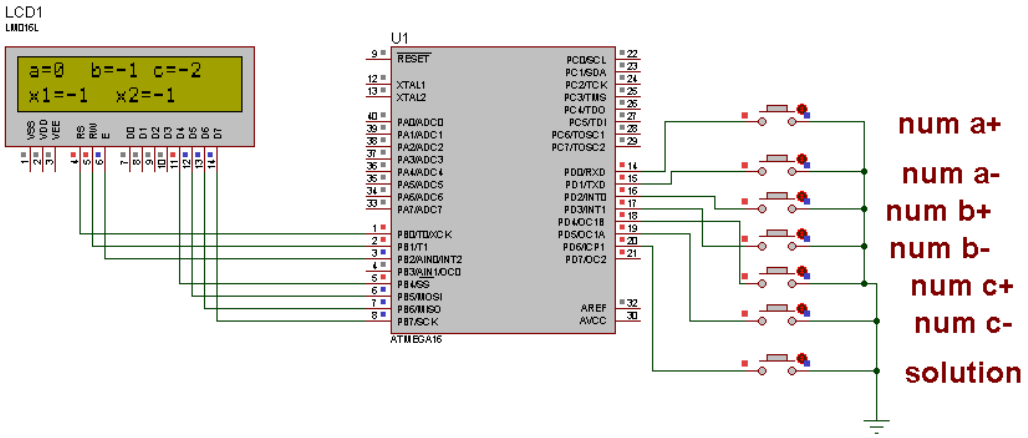


الكود :

```
#include <mega8.h>
#include<math.h>
#include <stdlib.h>
#include <stdio.h>
#asm
  .equ __lcd_port=0x18;
#endasm
#include <lcd.h>
void main(void)
{
char display_buffer[33];

lcd_init(16);
sprintf(display_buffer, "t0=%-i.%-u%cC\nt1=%-i.%-u%cK",12,abs(12%10),0xdf,12+100,abs((12+100)%10),0xdf);
lcd_clear();
lcd_puts(display_buffer);
}
```

تطبيقات مختلفة :

مثال 1 : حل معادلة من الدرجة الأولى



الكود :

```
#include <mega16.h>
#include<delay.h>
#include <stdlib.h>
#include <math.h>
#asm
  .equ __lcd_port=0x18;
#endasm
#include <lcd.h>
int i,j,k;
unsigned char *str;
void disp();
void exe();
void main()
{
lcd_init(16);
PIND=0xff;
PORTD=0xff;
lcd_putsf("a=0  b=0  c=0");
for(;;)
{
```

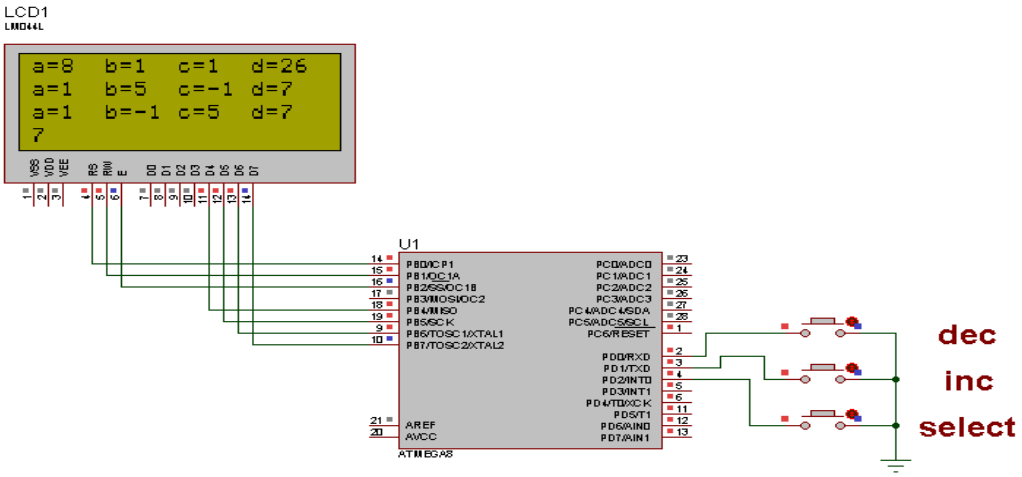
```
if(PIND.0==0)
{
lcd_clear();
i++;
disp();
delay_ms(50);
}
if(PIND.1==0)
{
lcd_clear();
i--;
disp();
delay_ms(50);
}
if(PIND.2==0)
{
lcd_clear();
j++;
disp();
delay_ms(50);
}
if(PIND.3==0)
{
lcd_clear();
j--;
disp();
delay_ms(50);
}
if(PIND.4==0)
{
```

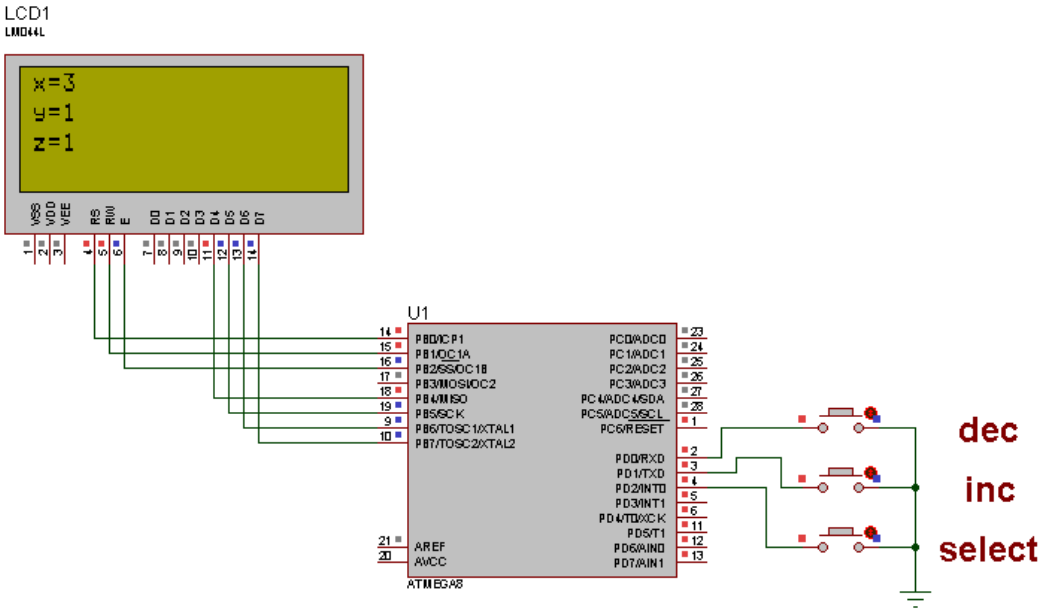
```
lcd_clear();
k++;
disp();
delay_ms(50);
}
if(PIND.5==0)
{
lcd_clear();
k--;
disp();
delay_ms(50);
}
if(PIND.6==0)
exe();
}
}

void exe()
    {
    int delta,x1,x2,a,b;
    b=pow(j,2);
    delta=b-4*i*k;
    if (delta >0)
{
a=sqrt(delta);
x1=(-j+a)/(2*i);
x2=(-j-a)/(2*i);
lcd_gotoxy(0,1);
lcd_putsf("x1=");
lcd_gotoxy(3,1);
```

```
itoa(x1,str);
lcd_puts(str);
lcd_gotoxy(7,1);
lcd_putsf("x2=");
lcd_gotoxy(10,1);
itoa(x2,str);
lcd_puts(str);
}
    if (delta==0)
{x1=-j/(2*i);
lcd_gotoxy(0,1);
lcd_putsf("x1=");
lcd_gotoxy(3,1);
itoa(x1,str);
lcd_puts(str);}
if (delta<0)
{lcd_gotoxy(0,1);
lcd_putsf("impossible solution");}
}



void disp()
{
    lcd_gotoxy(0,0);
    lcd_putsf("a=");
    lcd_gotoxy(2,0);
    itoa(i,str);
    lcd_puts(str);
```

```
lcd_gotoxy(5,0);
lcd_putsf("b=");
lcd_gotoxy(7,0);
itoa(j,str);
lcd_puts(str);
lcd_gotoxy(10,0);
lcd_putsf("c=");
lcd_gotoxy(12,0);
itoa(k,str);
lcd_puts(str);
}
```

مثال 2 : حل معادلة بثلاث مجاهيل حيث تختار select تختر العدد بعد زيادة هذا لعدد أو إنقاصه بواسطة inc& dec





الكود :

```
#include <mega16.h>
#include<delay.h>
#include <stdlib.h>
#asm
  .equ __lcd_port=0x18;
#endasm
#include <lcd.h>
int D,d1,d2,d3;
int a[3][4];
int k;
unsigned char *str;
void disp();
void exe();
void main()
{
lcd_init(20);
PIND=0xff;
```

```
PORTD=0xff;
disp();
for(;;)
{
while(1)
{
if(PIND.0==0)
{
lcd_clear();
k++;
a[0][0]=k;
disp();
delay_ms(50);
}
if(PIND.1==0)
{
lcd_clear();
k--;
a[0][0]=k;
disp();
delay_ms(50);
}
if(PIND.2==0)
{k=0;
delay_ms(100);
break;
}
}
while(1)
{
```

```
if(PIND.0==0)
{
lcd_clear();
k++;
a[0][1]=k;
disp();
delay_ms(50);
}
if(PIND.1==0)
{
lcd_clear();
k--;
a[0][1]=k;
disp();
delay_ms(50);
}
if(PIND.2==0)
{k=0;
delay_ms(100);
break;
}
}
while(1)
{
if(PIND.0==0)
{
lcd_clear();
k++;
a[0][2]=k;
disp();
```

```
delay_ms(50);
}
if(PIND.1==0)
{
lcd_clear();
k--;
a[0][2]=k;
disp();
delay_ms(50);
}
if(PIND.2==0)
{k=0;
delay_ms(100);
break;
}
}
while(1)
{
if(PIND.0==0)
{
lcd_clear();
k++;
a[0][3]=k;
disp();
delay_ms(50);
}
if(PIND.1==0)
{
lcd_clear();
k--;
```

```
a[0][3]=k;
disp();
delay_ms(50);
}
if(PIND.2==0)
{k=0;
delay_ms(100);
break;
}
}
while(1)
{
if(PIND.0==0)
{
lcd_clear();
k++;
a[1][0]=k;
disp();
delay_ms(50);
}
if(PIND.1==0)
{
lcd_clear();
k--;
a[1][0]=k;
disp();
delay_ms(50);
}
if(PIND.2==0)
{k=0;
```

```
delay_ms(100);
break;
}
}
while(1)
{
if(PIND.0==0)
{
lcd_clear();
k++;
a[1][1]=k;
disp();
delay_ms(50);
}
if(PIND.1==0)
{
lcd_clear();
k--;
a[1][1]=k;
disp();
delay_ms(50);
}
if(PIND.2==0)
{k=0;
delay_ms(100);
break;
}
}

while(1)
```

```
{
if(PIND.0==0)
{
lcd_clear();
k++;
a[1][2]=k;
disp();
delay_ms(50);
}
if(PIND.1==0)
{
lcd_clear();
k--;
a[1][2]=k;
disp();
delay_ms(50);
}
if(PIND.2==0)
{k=0;
delay_ms(100);
break;
}
}
while(1)
{
if(PIND.0==0)
{
lcd_clear();
k++;
a[1][3]=k;
```

```
disp();
delay_ms(50);
}
if(PIND.1==0)
{
lcd_clear();
k--;
a[1][3]=k;
disp();
delay_ms(50);
}
if(PIND.2==0)
{k=0;
delay_ms(100);
break;
}
}
while(1)
{
if(PIND.0==0)
{
lcd_clear();
k++;
a[2][0]=k;
disp();
delay_ms(50);
}
if(PIND.1==0)
{
lcd_clear();
```

```
k--;
a[2][0]=k;
disp();
delay_ms(50);
}
if(PIND.2==0)
{k=0;
delay_ms(100);
break;
}
}
while(1)
{
if(PIND.0==0)
{
lcd_clear();
k++;
a[2][1]=k;
disp();
delay_ms(50);
}
if(PIND.1==0)
{
lcd_clear();
k--;
a[2][1]=k;
disp();
delay_ms(50);
}
if(PIND.2==0)
```

```
{k=0;
delay_ms(100);
break;
}
}
while(1)
{
if(PIND.0==0)
{
lcd_clear();
k++;
a[2][2]=k;
disp();
delay_ms(50);
}
if(PIND.1==0)
{
lcd_clear();
k--;
a[2][2]=k;
disp();
delay_ms(50);
}
if(PIND.2==0)
{k=0;
delay_ms(100);
break;
}
}
```

```
while(1)
{
if(PIND.0==0)
{
lcd_clear();
k++;
a[2][3]=k;
disp();
delay_ms(50);
}
if(PIND.1==0)
{
lcd_clear();
k--;
a[2][3]=k;
disp();
delay_ms(50);
}
if(PIND.2==0)
{k=0;
delay_ms(100);
break;
}
}
break;
}
exe();
}
void exe()
    {
```

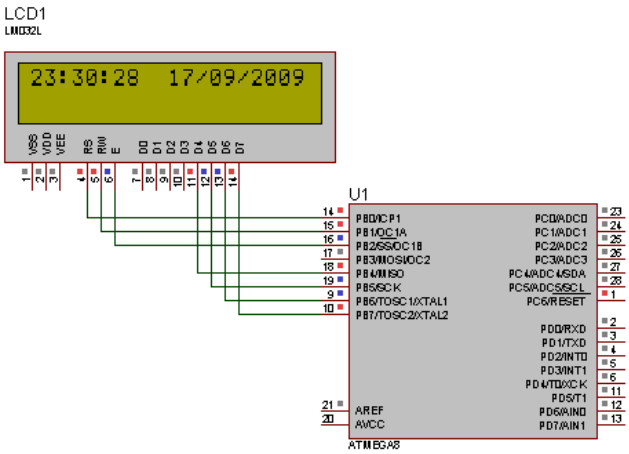
```
int c,d,e;
D=(a[0][0]*(a[1][1]*a[2][2]-a[1][2]*a[2][1]))-(a[0][1]*(a[1][0]*a[2][2]-a[1][2]*a[2][0]))+(a[0][2]*(a[1][0]*a[2][1]-a[1][1]*a[2][0]));
d1=(a[0][3]*(a[1][1]*a[2][2]-a[1][2]*a[2][1]))-(a[0][1]*(a[1][3]*a[2][2]-a[1][2]*a[2][3]))+(a[0][2]*(a[1][3]*a[2][1]-a[1][1]*a[2][3]));
d2=(a[0][0]*(a[1][3]*a[2][2]-a[1][2]*a[2][3]))-(a[0][3]*(a[1][0]*a[2][2]-a[1][2]*a[2][0]))+(a[0][2]*(a[1][0]*a[2][3]-a[1][3]*a[2][0]));
d3=(a[0][0]*(a[1][1]*a[2][3]-a[1][3]*a[2][1]))-(a[0][1]*(a[1][0]*a[2][3]-a[1][3]*a[2][0]))+(a[0][3]*(a[1][0]*a[2][1]-a[1][1]*a[2][0]));
lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("x=");
lcd_gotoxy(2,0);
c=d1/D;
itoa(c,str);
lcd_puts(str);
lcd_gotoxy(0,1);
lcd_putsf("y=");
lcd_gotoxy(2,1);
d=d2/D;
itoa(d,str);
lcd_puts(str);
lcd_gotoxy(0,2);
lcd_putsf("z=");
lcd_gotoxy(2,2);
e=d3/D;
itoa(e,str);
lcd_puts(str);
}
void disp()
{
lcd_gotoxy(0,0);
lcd_putsf("a=");
lcd_gotoxy(2,0);
```

```
itoa(a[0][0],str);
lcd_puts(str);
lcd_gotoxy(5,0);
lcd_putsf("b=");
lcd_gotoxy(7,0);
itoa(a[0][1],str);
lcd_puts(str);
lcd_gotoxy(10,0);
lcd_putsf("c=");
lcd_gotoxy(12,0);
itoa(a[0][2],str);
lcd_puts(str);
lcd_gotoxy(15,0);
lcd_putsf("d=");
lcd_gotoxy(17,0);
itoa(a[0][3],str);
lcd_puts(str);

lcd_gotoxy(0,1);
lcd_putsf("a=");
lcd_gotoxy(2,1);
itoa(a[1][0],str);
lcd_puts(str);
lcd_gotoxy(5,1);
lcd_putsf("b=");
lcd_gotoxy(7,1);
itoa(a[1][1],str);
lcd_puts(str);
lcd_gotoxy(10,1);
lcd_putsf("c=");
```

```
lcd_gotoxy(12,1);
itoa(a[1][2],str);
lcd_puts(str);
lcd_gotoxy(15,1);
lcd_putsf("d=");
lcd_gotoxy(17,1);
itoa(a[1][3],str);
lcd_puts(str);

lcd_gotoxy(0,2);
lcd_putsf("a=");
lcd_gotoxy(2,2);
itoa(a[2][0],str);
lcd_puts(str);
lcd_gotoxy(5,2);
lcd_putsf("b=");
lcd_gotoxy(7,2);
itoa(a[2][1],str);
lcd_puts(str);
lcd_gotoxy(10,2);
lcd_putsf("c=");
lcd_gotoxy(12,2);
itoa(a[2][2],str);
lcd_puts(str);
lcd_gotoxy(15,2);
lcd_putsf("d=");
lcd_gotoxy(17,2);
itoa(a[2][3],str);
lcd_puts(str);
```

```
lcd_gotoxy(0,3);
itoa(k,str);
lcd_puts(str);
}
```

مثال 3 : الساعة مع التاريخ



الكود :

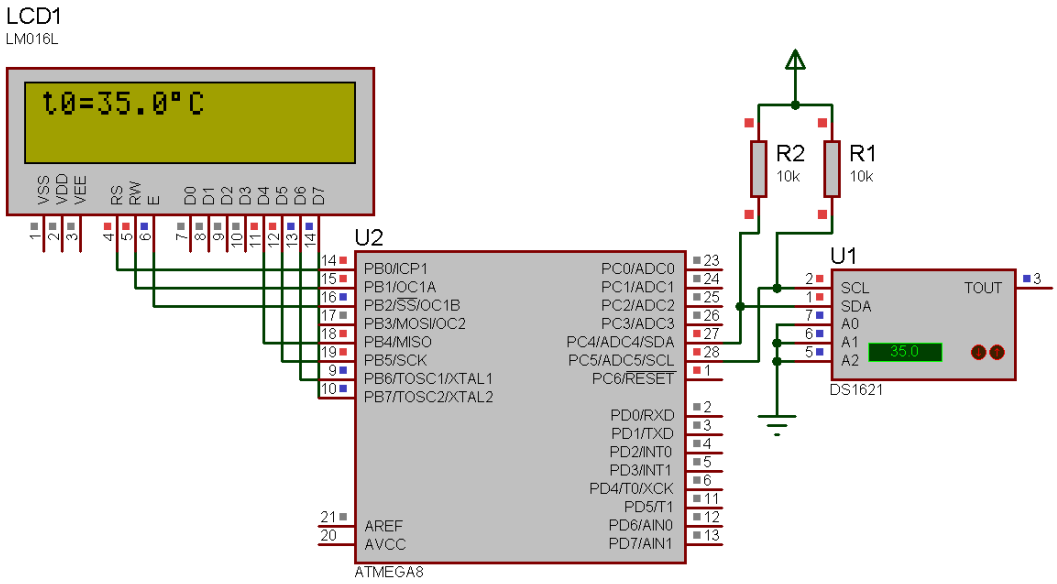
```
#include <mega8.h>
#include <lcd.h>
#include <stdio.h>
#asm
.equ __lcd_port=0x18
#endasm
unsigned char second;   //enter the current time, date, month, and year
unsigned char minute;
unsigned char hour;
unsigned char date;
unsigned char month;
unsigned int year;
char lcd_buffer[33];
char not_leap(void);
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
```

```
if (++second==60)       //keep track of time, date, month, and year
  {
    second=0;
    if (++minute==60)
    {
      minute=0;
      if (++hour==24)
      {
        hour=0;
        if (++date==32)
        {
          month++;
          date=1;
        }
        else if (date==31)
        {
          if ((month==4) || (month==6) || (month==9) || (month==11))
          {
            month++;
            date=1;
          }
        }
        else if (date==30)
        {
          if(month==2)
          {
            month++;
            date=1;
          }
        }
        else if (date==29)
```

```
            {
               if((month==2) && (not_leap()))
               {
                  month++;
                  date=1;
               }
            }
            if (month==13)
            {
               month=1;
               year++;
            }
         }
      }
   }
   sprintf(lcd_buffer,"%02d:%02d:%02d  %02d/%02d/%04d",hour, minute, second, date, month, year);
   lcd_gotoxy(0,0);
   lcd_puts(lcd_buffer);
}


char not_leap(void)      //check for leap year
{
   if (!(year%100))
      return (char)(year%400);
   else
      return (char)(year%4);
}
void main()
{
lcd_init(20);
TCCR1A=0x00;
TCCR1B=0x0B;
```

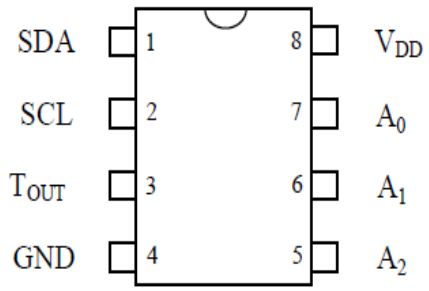
```
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0xF4;
OCR1AL=0x24;
OCR1BH=0x00;
OCR1BL=0x00;
TIMSK=0x10;
#asm("sei")
hour = 23;
minute = 30;
second = 00;
date = 17;
month = 9;
year = 2009;
}
```

مثال 4 : مقياس درجة حرارة



PIN DESCRIPTION

SDA	- 2-Wire Serial Data Input/Output
SCL	- 2-Wire Serial Clock
GND	- Ground
T_{OUT}	- Thermostat Output Signal
A0	- Chip Address Input
A1	- Chip Address Input
A2	- Chip Address Input
V_{DD}	- Power Supply Voltage



DS1621 8-PIN DIP (300mil)

الكود :

```
#include <mega8.h>
#include <math.h>
#include <stdio.h>
#include <delay.h>
#asm
  .equ __lcd_port=0x18;
#endasm
#include <lcd.h>
#asm
   .equ __i2c_port=0x15
   .equ __sda_bit=4
   .equ __scl_bit=5
#endasm
#include <ds1621.h>

char display_buffer[33];
void main(void)
{
int t0;
lcd_init(16);
i2c_init();
```

```
ds1621_start(0) ;
ds1621_init(0,20,25,0);
for(;;)
{
    t0=ds1621_temperature_10(0);
    sprintf(display_buffer,"t0=%-i.%-u%cC",t0/10,abs(t0%10),0xdf);/* display the temperatures */
    lcd_clear();
    lcd_puts(display_buffer);
    delay_ms(100);
}
}
```

المؤقتات والعدادات :

1 - الفرق بين المؤقت والعداد :

1-المؤقت : بمجرد ورود إشارة على قطب التفعيل للمؤقت أو بمجرد تفعيل المؤقت فإنه سيبدأ بالتوقيت أي يعد فترات زمنية محددة وهكذا حتى مرور فترة زمنية محددة ليعطي أمر ما في نهاية هذه الفترة أي عند حدوث طفحان في قيمته .

مثال: إطفاء محرك كهربائي بعد مرور 5 دقائق على إقلاعه.

2-العداد: يقوم العداد بزيادة القيمة التي يحويها عند كل نبضة ترد على قطب تفعيله وكذلك يقوم بأمر ما عند مرور عدد من النبضات .

مثال : إطفاء سير ناقل بعد مرور 50 قطعة على هذا السير.

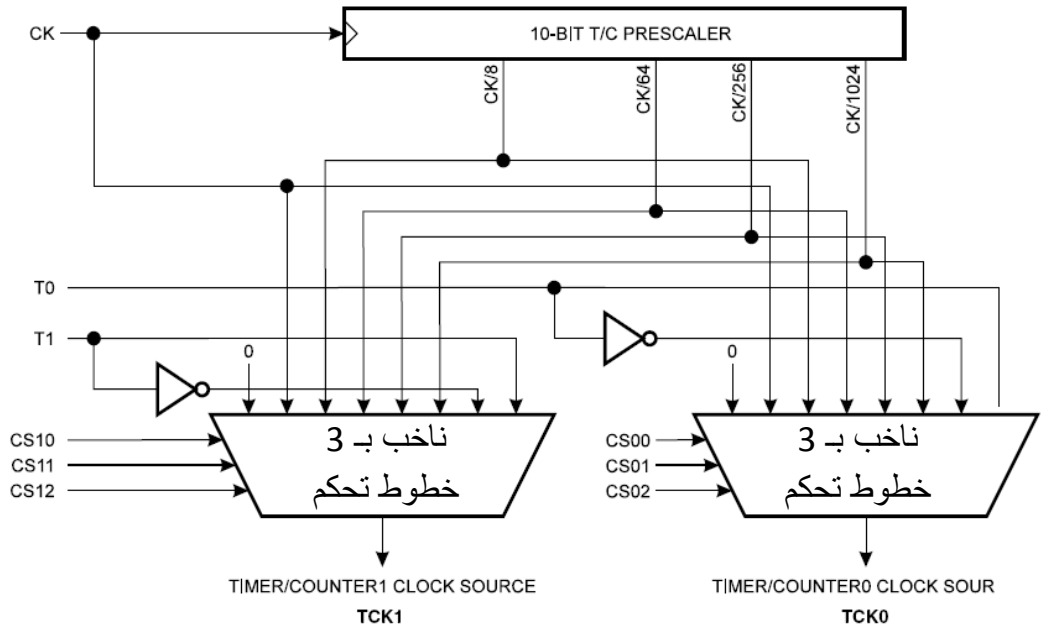
2 – المؤقت والعداد ضمن atmega8 :

لكل مايكرو مؤقتات وعدادات خاصة بيه لذا عندا استخدام أي مايكرو يجب الإطلاع على data sheet

1- مؤقت / عداد T/C0 : وهو يمثل القطب رقم 6 PD4 وهو عبارة عن مؤقت / عداد بـ 8 بت أي أن أعظم قيمة يمكن أن يعدها أو يؤقتها هي 256 = 2^8. بقناة واحدة يملك مقسم داخلي بدقة 10 بت حيث يمكن اختيار نبضات الساعة كاملة كمدخل للتوقيت أو نبضات ساعة مقسمة. .

3- البنية الداخلية للمؤقت :

Figure 28. Timer/Counter Prescaler



نلاحظ أن المؤقت / العداد 0 و 1 يتألفان مما يلي :

مقسم تردد أولي بدقة 10 بت يأخذ دخله من الكريستالة الخارجية للمتحكم ويعطي على خرجه 4 تقسيمات للتردد CK / 1024 , CK / 256 , CK / 64 , CK / 8 وهذا يعطي مجالات واسعة لاختيار التردد المراد العمل عليه في المؤقت.

المؤقت / العداد 1 و 0 زودا بناخب بـ 3 خانات تحكم وبالتالي فإن الخرج سيكون حالة من إحدى 8 حالات دخل هي كما نلاحظ على الرسم (مرتبة من يسار الناخب إلى يمينه):

1 - خيار تطبيق نبضات الساعة كاملة على المؤقت

2 - خيار تطبيق نبضات الساعة مقسمة على 8 على المؤقت

3 - خيار تطبيق نبضات الساعة مقسمة على 16على المؤقت

4 - خيار تطبيق نبضات الساعة مقسمة على 256 على المؤقت

5 - خيار تطبيق نبضات الساعة مقسمة على 1024 على المؤقت

6 - خيار العد الخارجية عند الجبهات الهابطة على العداد

7 - خيار العد الخارجية عند الجبهات الصاعدة على العداد

حيث يتم الحصول على نبضات الساعة من المقسم أو الكريستالة في حالة العمل كمؤقت ومن القطب الخارجي في حالة العمل كعداد.

4 - مسجلات المؤقت والعداد (0) :

1 – المسجل TCCR0 : حيث تحدد هذه الخانات نسبة التقسيمات للتردد الأساسي FCK والتي ستصبح دخل فعال للمؤقت / العداد 0

Bit	7	6	5	4	3	2	1	0	
$33 ($53)	–	–	–	–	–	CS02	CS01	CS00	TCCR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

كما أن هذه الخانات تؤمن حجب المؤقت / العداد عن طريق نمط التوقف ،بما أنه لدينا 3 خانات فيكون لدينا 8 حالات 2^3 = 8 ، وفق الجدول التالي:

الشرح	مؤقت / عداد	CS02	CS01	CS00
نمط توقف المؤقت / العداد	حجب	0	0	0
CK	مؤقت	0	0	1
CK / 8	مؤقت	0	1	0
CK / 64	مؤقت	0	1	1
CK / 256	مؤقت	1	0	0
CK / 1024	مؤقت	1	0	1
عد النبضات المطبقة على القطب T0 عند الجبهات الهابطة	عداد	1	1	0
عد النبضات المطبقة على القطب T0 عند الجبهات الصاعدة	عداد	1	1	1

2 – المسجل TCNT0 : وهو عبارة عن مسجل تحكم عرض 8 بت يتم فيه تخزين القيمة الحالية (الآنية) التي وصل إليها المؤقت / العداد 0 وباعتبار أنه 8 بت فإن أعظم قيمة يمكن تخزينها هي $FF = 255

Bit	7	6	5	4	3	2	1	0	
$32 ($52)	MSB							LSB	TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

3 – المسجل TIMSK : الخانة TOIE0 خانة مقاطعة الطفحان حيث يتم تفعيل هذه المقاطعة عند وضع هذه الخانة 1 منطقي وبالتأكيد يجب أن يكون علم المقاطعة العام SREG مفعلا I = 1

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

حادثة الطفحان تعني امتلاء مسجل العداد / المؤقت TCNT0 بقيمة تتجاوز 255 عندها تحدث المقاطعة حيث ينتقل المعالج إلى روتين خدمة المقاطعة الواقع عند العنوان $006 وكذلك تفعل الخانة TOV0 = 1 الواقعة في مسجل أعلام المؤقت/ العداد TIFR .

4 – المسجل TIFR : الخانة TOV0 خانة علم طفحان المؤقت / العداد 0 تتفعل هذه الخانة عندما يحدث الطفحان في مسجل المؤقت / العداد 0 لكي تدل المعالج على حدوث الطفحان و يتم تصفير هذه الخانة من قبل المتحكم بعد تنفيذ روتين خدمة المقاطعة .

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	–	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

5 – الحصول على زمن واحد ثانية :

للحصول على مؤقت بمقدار 1 ثانية عن طريق شاشة الإظهار 7-SEG أي زيادة العدد الظاهر على شاشة الإظهار كل 1 ثانية باستخدام المؤقت / العداد 0 ، يمكن اختيار أي تقسيمة تناسب العمل ولكن بفرض اخترنا التقسية CK / 256

لننتبه للحسابات التالية :

بفرض أن تردد الكريستالة 4MHZ وبالتالي فالتردد المطبق على المؤقت عند التقسيمة CK / 256 هو :

4000000/256 = 15625 HZ ومنه نحسب زمن النبضة الواحدة (الدور) : 1/15625 = 0.000064 SEC

وبالتالي كل زمن قدره 0.000064 SEC سوف يزداد مسجل المؤقت بمقدار 1 وللوصول للقيمة الأعظمية (الطفحان) يجب أن يزداد بمقدار 255 مرة ، ويكون الزمن اللازم لطفحان المؤقت هو :

255 * 0.000064 = 0.01632 SEC

عند مرور هذا الزمن سيحدث الطفحان و تحدث مقاطعة طفحان المؤقت 0 ولكن نلاحظ أن هذا الزمن ليس 1 ثانية وبالتالي يجب تكراره عدد من المرات يساوي K حتى نحصل على زمن 1 ثانية :

0.01632 * K = 1 SEC → K = 1/0.01632 = 61 = $3D

0.01632 * 61 = 0.99552SEC ~ 1 SEC

وبالتالي برنامج المقاطعة يجب أن يكرر هذه العملية K مرة وزيادة المسجل الذي يحوي على القيمة الظاهرة على الشاشات بعد انقضاء 1 ثانية .

تردد الكرستالة	نسبة التقسيم	التردد بعد التقسيم	زمن التردد 1/F	الزمن * 256	1 تقسيم الزمن
1000000	8	125000	8*10^-6	2.048*10^-3	**488.288**
1000000	16	62500	1.6*10^-5	4.096*10^-3	**244.1416**
1000000	256	3906.25	2.56*10^-4	0.0655	**15.267**
1000000	1024	976.5625	1.024*10^-3	0.262144	**3.814**
4000000	8	500000	2*10^-6	5.12*10^-4	**1953.125**
4000000	16	250000	4*10^-6	1.024*10^-3	**976.5625**
4000000	256	15625	6.4*10^-5	0.016384	**61.036…**
4000000	1024	3906.25	2.56*10^-4	0.065536	**15.25…**

ملاحظة: لا يمكن باستخدام المؤقت / العداد 0 الحصول على زمن دقيق تماماً لأنه لا يملك أي نمط مسك أو مقارنة أي أنه سيعد حتى الوصول إلى الطفحان وهي سيئة.

مؤقت المراقبة (watchdog) :

هو عبارة عن مؤقت داخلي يعمل على مراقبة عمل زمن البرنامج حيث يقوم بتصفير المعالج بعد مدة زمنية محددة يحددها المبرمج حسب الزمن الذي يستغرقه البرنامج وذلك لحصول بعد الأخطاء الناتجة ضمن المعالج كالوقوف عند تعليمة ما

حساب الزمن :

بفرض أن البرنامج يستغرق زمن 1.8 فإننا نضبط المؤقت على الزمن 1.9 بجهد تغذية 5 v لأن الجهد يؤثر أيضا على توليد تردد الكريستالة

ملاحظة :

تعليمات التأخير يجب الانتباه إليها حيث لا نستطيع استخدامها ضمن المؤقت لصعوبة إيجاد الزمن المطلوب من المراقب

مسجلات المؤقت :

لا يحوي سوى مسجل واحد هو :

1 - المسجل WDCE :

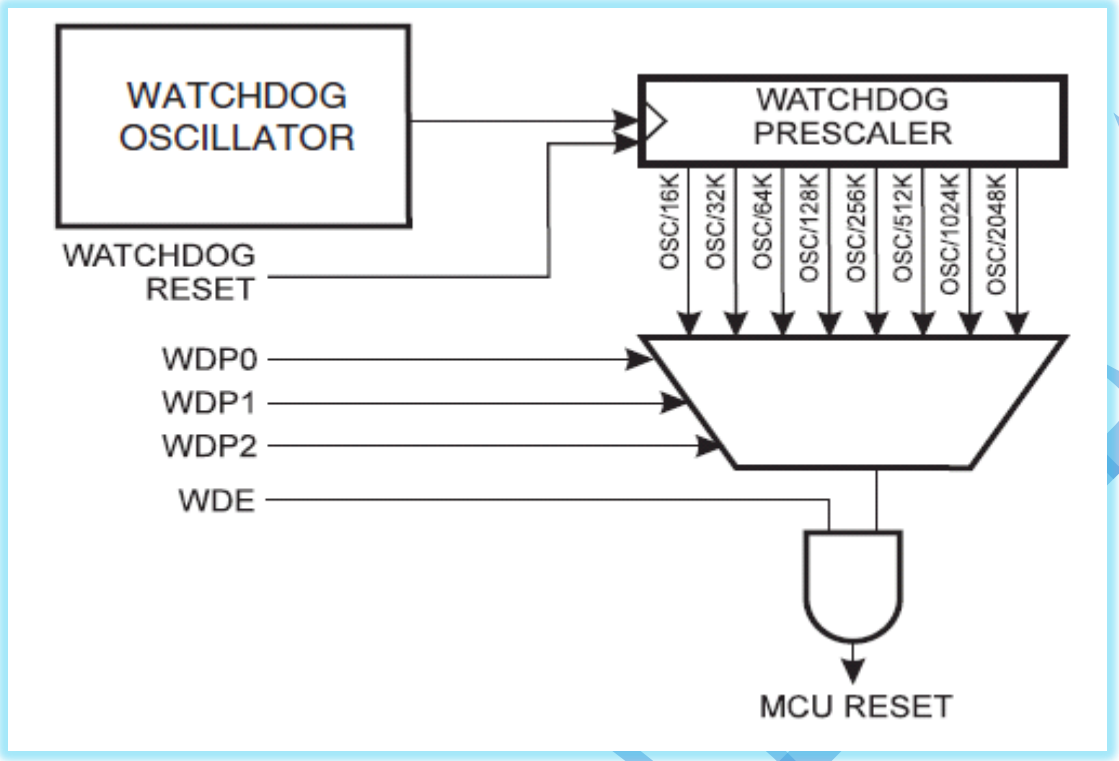
الخانة 0 و 1 و 2 : خانات الناخب حيث يتم اختيار التقسيمات لاختيار الزمن المطلوب للبرنامج المطلوب له

الخانة 3 : لتفعيل عمل المؤقت من لحظتها يجب وضع 0 في الخانة 4

الخانة 4 : خانة إيقاف عمل مؤقت المراقبة

Bit	7	6	5	4	3	2	1	0	
	–	–	–	WDCE	WDE	WDP2	WDP1	WDP0	WDTCR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

المخطط الداخلي :



والجدول التالي يوضح معظم الأزمنة الممكن استخدامها عند التغذيتين 3v و 5v

WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V_{CC} = 3.0V	Typical Time-out at V_{CC} = 5.0V
0	0	0	16K (16,384)	17.1 ms	16.3 ms
0	0	1	32K (32,768)	34.3 ms	32.5 ms
0	1	0	64K (65,536)	68.5 ms	65 ms
0	1	1	128K (131,072)	0.14 s	0.13 s
1	0	0	256K (262,144)	0.27 s	0.26 s
1	0	1	512K (524,288)	0.55 s	0.52 s
1	1	0	1,024K (1,048,576)	1.1 s	1.0 s
1	1	1	2,048K (2,097,152)	2.2 s	2.1 s

رغم أهمية مؤقت المراقبة فإنه لا يستخدم لندرة البرامج التي تحتاج إلى زمن دقيق

الأعداد السالبة والموجبة وطباعتها على lcd :

ويتم بإضافة signed قبل نوع المتحول المستخدم للأعداد الموجبة والسالبة و unsigned للأعداد الموجبة فقط

لطباعة الأعداد الموجبة

لطباعة الأعداد الموجبة

printf("%u\n",num)

تعريف الثوابت :

1 – الشكل الأول

```
const  int  num;
```

2 – الشكل الثاني

```
#define pi  3.14
```

ولا تنتهي بفاصلة منقوطة

عند استدعاء pi فإنه يعطي قيمة pi بدلا منها ويتم اختيار نوع التابع تلقائيا

مؤشر الزيادة والنقصان وباقي القسمة :

1 - مؤشر الزيادة ++ : وتقوم بزيادة واحد إلى المتحول المستخدم معه

```
int i;
++i;
```

2 – مؤشر النقصان -- : وتقوم بإنقاص واحد إلى المتحول المستخدم معه

```
int i;
--i;
```

3 – مؤشر باقي القسمة % : يعطي باقي قسمة عدد int على عدد ما

```
12%10
```

النتيجة

2

4 - مؤشر الإزاحة >> لليسار و << لليمين : وتقوم بإزاحة بت من البايت إما لليمين أو اليسار بخطوة محددة

```
int  Ismael ;
```

```
Ismael<<1;
```

إزاحة بخطوة واحدة

```
Ismael>>1;
```

```
Ismael<<4;
```

إزاحة بأربع خطوات

4 - مؤشر خاصة بالبتات :

المؤثر/المتحول	b	a
&	a&b	and
|	a|b	or
^	a^b	xor
~	~a	not

التعليمة scanf() : للإدخال مع تنسيقات

scanf("%d%d",a,b)

الرمز	الشرح
%d %i	للأعداد الصحيحة int
%f %e %g	للأعداد الحقيقية double float
%u	للأعداد بدون إشارة
%c	للرموز والأحرف
%s	للنصوص والسلاسل
%o	النظام الثماني
%x	النظام السداسي عشر
%p	للمؤشرات

المؤشرات :

المؤشر عبارة عن متحول نقوم بإسناد عنوان متحول وليس قيمته

تعريف مؤشر int *a;

تعريف متحول int i;

إسناد عنوان المتحول إلى المؤشر a=&i

إسناد قيمة للمتحول عن طريق المؤشر وهذه ميزات المؤشر *a=10;

الملفات الرأسية (header files) :

حيث توجد بها ثوابت , نماذج توابع و struct تساعدنا في برامجنا الضخمة ويتم إنشاءها عن طريق ملف نصي وحفظه بالامتداد .h وتستخدم لإنشاء مكاتب خاصة بالمبرمج

```
int add(int a,int b)
{
return a+b;
}
int sub(int a,intb)
{
return a-b;
}
```

البرنامج محفوظ باسم function

الاستدعاء :

```
#include<function.h>
int i,j=10,k=20;
i=add(j,k);
```

التراكيب (structures) :

تسمى البنية أيضا هي مجموعة من متغير واحد أو أكثر تجمع تحت اسم واحد يسهل استعمالها يمكن أن تكون بها متغيرات مختلفة الأنواع

تعريف بنية

```
struct ismael
{
الثوابت والمتغيرات
int a;
float b;
};
```

استخداماها

```
ismael.a=10;
ismael.b=3.1419;
```

التركيب باستخدام union :

نفس ال struct ولكن إسناد قيمة لأحد متحولاتها فإنه يسند للأخرى فإذا كانت من أنواع مختلفة فإنا النتائج ستكون غير موثوق بها

التعليمة (typdef) : وتستخدم لكي تعوض قيمة عن قيمة أخرى

```
typedef int decimal;
void main()
{
decimal a ,b;
}
```

decimal تكافئ التعليمة int

التعليمة extern :

تستخدم لجعل متحول محلي (local) مشترك مع متغير عام (global)

```
function()
{
extern int a;
}
```

```
void main()
{
int a
}
```

التوجيه #undef :هو معاكس #define حيث يقوم بإلغاء الثوابت والمختصرات

الملف الرأسي assert : ويحوي تعليمة واحدة ()assert

```
assert(4>1);
```

الملف الرأسي ctype :

ويحوي التعليمات التالية :

isalnum(char c)
returns 1 if c is alphanumeric. **يعيد واحد إذا رقم أو حرف**
isalpha(char c)
returns 1 if c is alphabetic. **يعيد واحد إذا حرف من حروف الأبجدية**
isascii(char c)
returns 1 if c is an ASCII character (0..127). **يعيد واحد إذا محرف أسكي**
iscntrl(char c)
returns 1 if c is a control character (0..31 or 127). **يعيد واحد إذا محارف التحكم من 0 حتى 31 ومن 127 حتى 255**
isdigit(char c)
returns 1 if c is a decimal digit. **يعيد واحد إذا كان من 48 حتى 57 الأسكي**
islower(char c)
returns 1 if c is a lower case alphabetic character. **يعيد واحد إذا حرف صغير**
isprint(char c)
returns 1 if c is a printable character (32..127). **يعيد واحد إذا محرف قابل للطباعة من 32 حتى 126**
ispunct(char c)
returns 1 if c is a punctuation character (all but control and alphanumeric). **يعيد واحد إذا رموز الترقين مثل الفاصلة والنقطة الخ .....**
isspace(char c)

returns 1 c is a white-space character (space, CR, HT). **يعيد واحد إذا space مضغوطة**
isupper(char c)
returns 1 if c is an upper-case alphabetic character. **يعيد واحد إذا حرف كبير**
isxdigit(char c)
returns 1 if c is a hexadecimal digit. **يعيد واحد إذا رقم سداسي عشر**
toascii(char c)
returns the ASCII equivalent of character c. **يعيد الأسكي للمحرف c**
toint(char c)
interprets c as a hexadecimal digit and returns an usigned char from 0 to 15. **يترجم المحرف c السداسي عشر ويعيد قيمة 0 حتى 15**
tolower(char c)
returns the lower case of c if c is an upper case character, else c. **يحول المحرف C إلى حرف صغير**
toupper(char c)
returns the upper case of c if c is a lower case character, else c. **يحول المحرف c إلى حرف كبير**

بعض الخصائص في لغة ال c :

```
#define SEI() #asm("sei")

#define CLI() #asm("cli")

#define NOP() #asm("nop")

#define WDR() #asm("wdr")


#define SET_BIT(byte,bit) (byte |= bit) // Set bit in byte

#define CLR_BIT(byte,bit) (byte &= ~bit) // Clear bit in byte

#define TGL_BIT(byte,bit) (byte ^= bit) // Toggle bit in byte


#define IS_SET(reg,bit) (reg & bit) // TRUE if bit = 1

#define IS_CLR(reg,bit) !(reg & bit) // TRUE if bit = 0
```

```
#define BIT(x) (1 << (x)) // Define bit value

#define HI(x) ((x) >> 8) // Highbyte of 16-bit value


#define ABS(x) ((x >= 0) ? x : -x) // Absolute value of x


#define HI_HEX(x) ("0123456789ABCDEF" [x >> 4]) // Create hex of high nibble

#define LO_HEX(x) ("0123456789ABCDEF" [x & 0x0F]) // Create hex of low
nibble
```

أنماط عمل الطاقة :

يوجد عدة أنواع لعمل المعالج لحفظ الطاقة وذلك حفاظا على الطاقة الكهربائية والاستخدام أثناء قطع الكهرباء وذلك بتغذية المعالج من بطارية خارجية تتضمن عمل البرنامج أثناء أنقاطع الكهرباء لأنه أحيانا يستوجب علينا تفعيل هذه الأنماط ففي تطبيق الساعة الإلكترونية لا يمكن عند كل أنقاطع للكهرباء استوجب علينا إعادة الضبط

مسجلات الطاقة :

MCU Control Register – MCUCR The MCU Control Register contains control bits for power management.

Bit	7	6	5	4	3	2	1	0	
	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

الخانات 4 و5 و7 :تعمل وفق الجدول التالي

نمط النوم	SM1	SM1	SM0
idle	0	0	**0**
adc noise reduction	0	0	**1**
power-down	0	1	**0**
Power-save	0	1	**1**
reserved	1	0	**0**
reserved	1	0	**1**
standby	1	1	**0**

وهذه الأجزاء التي تعمل في كل نظام من الأنظمة السابقة

الخانة 7 :خانة تفعيل نظام عمل الطاقة SLEEP

SLEEP MODE	CLK CPU	CLK FLASH	CLK IO	CLK ADC	CLK ASY	MAIN CLOCK SOURCE ENABLED	TIMER OSC ENABLED	INT1 INT0	TWI ADDRESS MATCH	TIMER	SPM EEPROM READY	ADC	OTHER I/O
idle			x	x	x	x	x	x	x	x	X	x	**x**
Adc noise reduction				x	x	x	x	x	x	x	X	x	
Power down								x	x				
Power Save					x		x	x	x	x			
standby								x	x				

إنشاء الملفات الرأسية (library) :

تستخدم ملف رأسي خاص بالشخص المبرمج لتسهيل البرامج التي تحتوي على برامج مكررة دائما

طريقة الإنشاء :

1 – نضغط file ثم new ثم source ثم ok

2 – نكتب الآن ما يلي ضمن #pragma used السالبة والموجبة التوابع التي سوف تتضمن ملف ال source المحفوظ بالامتداد .c و الاسم التالي ismael

```
#pragma used+
int sum(char a, char b);
int mul(char a, char b);
#pragma used-
```

3 – نحفظ الملف بالامتداد ismael.h
4 – نضغط file ثم new ثم source ثم ok
5 – نكتب كود التوابع التي ضمنها ضمن الملف السابق

```
int sum(char a, char b)
 {
return a+b;
}
int mul(char a, char b)
{
return a*b;
}
```

6 – نحفظ الملف بالامتداد ismael.c
7 – نضغط file ثم convert to library فتظهر رسالة بنفس الاسم الذي حفظ بيه الملفان السابقين وحصرا يجب أن يكونا نفس الاسم فنضغط ok فتظهر رسالة بنجاح عملية إنشاء المكتبية

العداد والمؤقت 1 :

1 - عمله كمؤقت و عداد :

يتميز بعدة خصائص عن العداد 0 كماذكرنا حيث يتميز أنه بطول 16 بت

- مؤقت / عداد T/C1 : وهو يمثل القطب رقم 11 PD5 وهو عبارة عن مؤقت / عداد بـ 16 بت يملك مقسم داخلي بدقة 10 بت وله الميزات التالية:

1- تصميم حقيقي بـ 16 بت أي أن أعظم قيمة يمكن أن يعدها أو يؤقتها هي 2^16 = 65536 .

2-وحدتي مقارنة خرج منفصلتين مع مسجلي مقارنة خرج منفصلتين .

3-وحدة مسك قيمة.

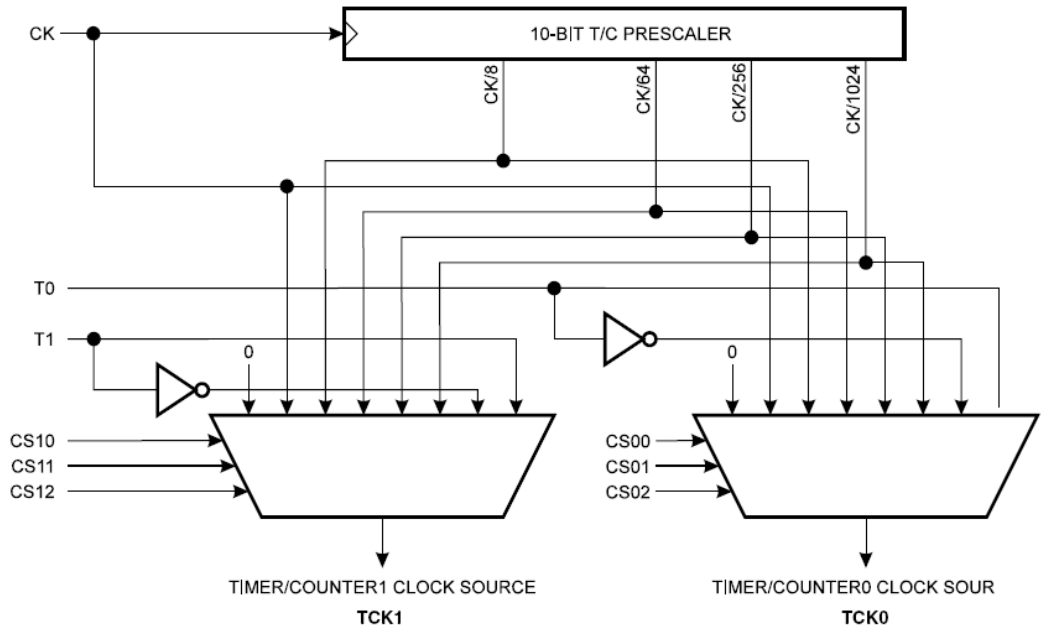
4-تصفير المؤقت عند حدوث تساوي المقارنة (إعادة تحميل آلية)

5-أزمنة وأنماط PWM متعددة.

6 - (4) مقاطعات خارجية منفصلة .

7 - يمكن أن يعمل عداد لحوادث خارجية.

Figure 28. Timer/Counter Prescaler



المسجل TCCR1B : حيث تحدد هذه الخانات نسبة التقسيمات للتردد الأساسي FCK والتي ستصبح دخل فعال للمؤقت / العداد 1

Timer/Counter 1 Control Register B – TCCR1B

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

والتقسيمات وفق الجدول التالي :

الشرح	مؤقت / عداد	CS12	CS11	CS10
نمط توقف المؤقت / العداد	حجب	0	0	0
CK	مؤقت	0	0	1
CK / 8	مؤقت	0	1	0
CK / 64	مؤقت	0	1	1
CK / 256	مؤقت	1	0	0
CK / 1024	مؤقت	1	0	1
عد النبضات المطبقة على القطب T0 عند الجبهات الهابطة	عداد	1	1	0
عد النبضات المطبقة على القطب T0 عند الجبهات الصاعدة	عداد	1	1	1

المسجل TCNT1H AND TCNT1L : وهو عبارة عن مسجل تحكم عرض 16 بت يتم فيه تخزين القيمة الحالية (الآنية) التي وصل إليها المؤقت / العداد 1 وباعتبار أنه 16 بت فإن أعظم قيمة يمكن تخزينها هي $FFFF = 65536

Timer/Counter 1 – TCNT1H and TCNT1L

Bit	7	6	5	4	3	2	1	0	
	TCNT1[15:8]								TCNT1H
	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

3 – المسجل TIMSK : الخانة TOIE1 خانة مقاطعة الطفحان حيث يتم تفعيل هذه المقاطعة عند وضع هذه الخانة 1 منطقي وبالتأكيد يجب أن يكون علم المقاطعة العام SREG مفعلا I = 1

Timer/Counter Interrupt Mask Register – TIMSK(1)

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

4 – المسجل TIFR : حيث يتم تفعيل خانة TOVO1 حيث هو مؤشر الأعلام حيث يستخدمه المايكرو لا نستخدمه أبدا

Timer/Counter Interrupt Flag Register – TIFR[1]

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	–	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

الكتابة على مسجل المؤقت / العداد 1 : عندما يكتب المعالج البايت العلوي فإن هذا البايت يتوضع في مسجل مؤقت TEMP وعندما يبدأ المعالج بكتابة البايت الثاني (السفلي) على مسجل المؤقت / العداد يقوم المسجل اللحظي بكتابة البايت العلوي على مسجل العداد / المؤقت .

وبالتالي عندما نقوم بالكتابة على هذا المسجل يجب البدء بالبايت العلوي TCNT1H ثم السفلي.

القراءة فتتم على الشكل التالي : عندما يقرأ المعالج المعطيات يبدأ بالبايت السفلي على عكس عملية الكتابة حيث يأخذ البايت السفلي ويضعه في المسجل TEMP وعندما يقرأ المعالج البايت العلوي فإنه في نفس اللحظة تتم قراءة البايت السفلي من المسجل اللحظي.

وبالتالي عند عملية القراءة يجب قراءة البايت السفلي TCNT1 L ثم العلوي .

2 – عمله كمقارن بين مسجلين :

يحتوي على مقارنين لقيمتين مختلفتين

المسجل TCCR1B :

Timer/Counter 1 Control Register B – TCCR1B

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

الخانة ICNC1 : حيث مدخل حذف الضجيج لمدخل المسك العدة

الخانة ICES1 : خانة اختيار جبهة مدخل المسك عندما تكون ICES1 =0 فإن عملية المسك ستتم عند تطبيق جبهة هابطة على مدخل المسك ICP بينما عندما تكون ICES = 1 فإن حادثة المسك ستحدث عند الجبهة الصاعدة

المسجل TCCR1A :

Timer/Counter 1 Control Register A – TCCR1A

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	W	W	R/W	R/W	

الخانات 4 و5 و6 و7 :

COM1A0 COMB0	COM1A1 COM1B1	الحالــة
0	0	**المؤقت / العداد لا يتصل مع القطب OC1A وOC1B ولا نحصل على أية إشارة**

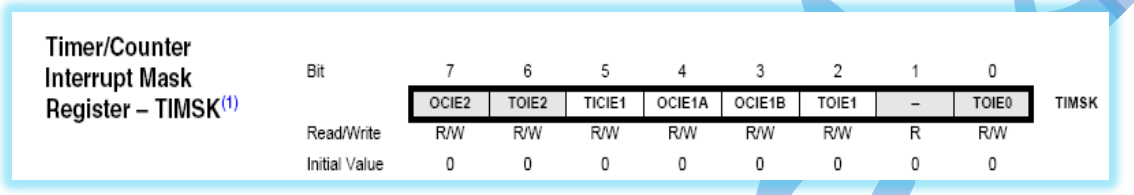
إعطاء إشارة معاكسة لحالة القطب toggle عند حصول المقارنة للمقارنين	0	**1**
إعطاء صفر منطقي عند حصول المقارنة للمقارنين	1	**0**
واحد منطقي عند حصول المقارنة للمقارنين	**1**	**1**

وذلك للحصول على إشارة عند حصول مقارنة على القطب الخارجي للمقارن

3 – المسجل TIMSK :

الخانة OCIE1A : خانة تمكين مقاطعة خرج المقارن الأول المؤقت / العداد1 : لتفعيل هذه المقاطعة يجب أن يكون علم المقاطعة العام مفعلا I=1 وكذلك يجب أن تكون هذه الخانة مفعلة OCIE1A = 1 .

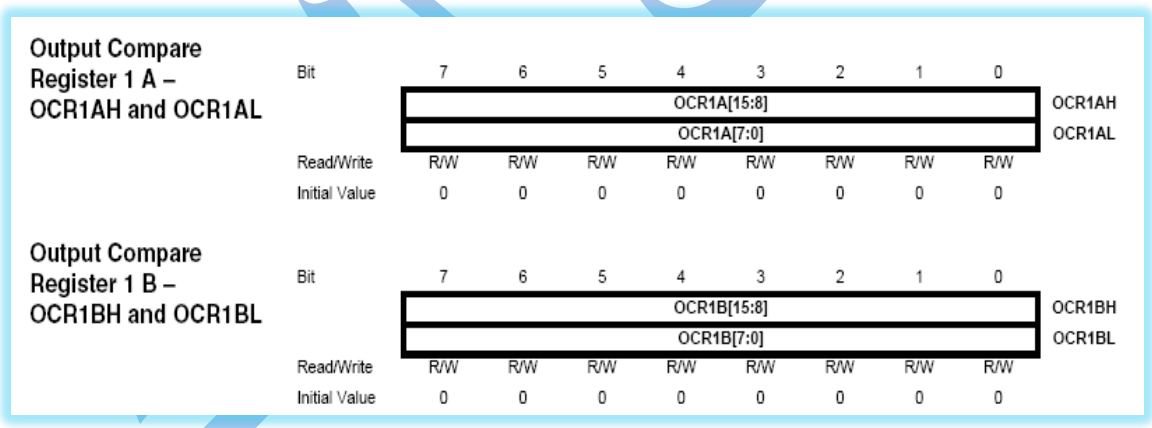
الخانة OCIE1B : خانة تمكين مقاطعة خرج المقارن الثاني المؤقت / العداد1



عند تحقق إحدى المقاطعتين السابقتين سوف ينفذ المتحكم روتين خدمة المقاطعة الواقعة عند العنوان $006 و $007 وذلك عند تساوي القيمة المخزنة في مسجلي المقارنة مع مسجل قيمة المؤقت / العداد 1

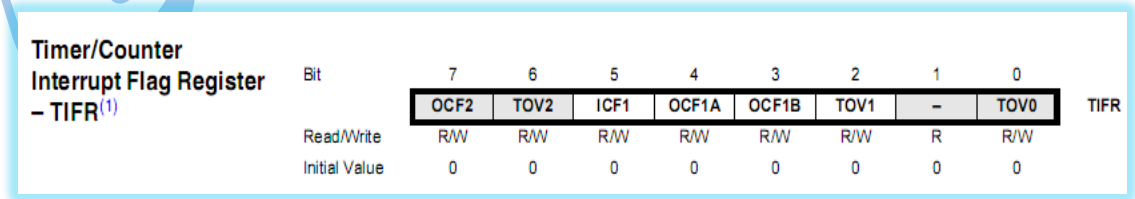
4 - مسجلي مقارنة الخرج للمؤقت / العداد 1 OCR1AH , OCR1AL و OCR1BH , OCR1BL :

يستخدم المسجل OCR1A و OCR1B كمسجل يحوي على المعطيات التي سوف تقارن مع المعطيات المحتواة في مسجل المؤقت / العداد TCNT1 حيث تخزن القيمة التي سوف تقارن فيه



5 - مسجل أعلام مقاطعة المؤقت/ العداد 1 TIFR :

كما قلنا لا يستخدم



الخانة OCF1A :تحدث عندما المقارنة تتحقق مع مسجل المقارنة 1 ومسجل العداد 1

الخانة OCF1B : تحدث عندما المقارنة تتحقق مع مسجل المقارنة 2 ومسجل العداد 1

3 – عمله كمولد عرض نبضة :

يستخدم التعديل عرض النبضة للحصول على إشارات متغيرة العرض هذا الاختلاف في العرض يؤدي إلى الاختلاف في زمن عرض النبضة والذي يستفاد منه في تحديد زمن العمل فعند عرض نبضة مدتها الزمنية ثانية غير عرض النبضة التي مدتها ربع ثانية ففي الأولى سيستغرق مدة العمل في الأولى أكثر من الثانية

---------- ---------- ---- ----

---------- ----

figure 0 Figure 1

نستنتج في الشكل الأول 1 تستغرق زمن أكثر من الشكل الثاني 0

عندما يتم وضع المؤقت / العداد في وضع الـ PWM فإن مسجل المؤقت / عداد 1 TCNT1 و مسجل مقارنة الخرج OCR1A يشكلان مولداً لنبضات معدلة .

حيث يعمل المؤقت / عداد 1 كعداد صاعد / هابط من القيمة الأصغرية $0000 وحتى القيمة الأعظمية (حسب دقة الـ PWM) المبينة بالجدول :

دقة عرض النبضة	القيمة العظمى للمؤقت	تردد عرض النبضة
8 bit	$00ff(255)	Ftc1/510
9 bit	$01ff(511)	Ftc1/1022
10 bit	$03ff(1023)	Ftc1/2046

حيث ftc1 هو التردد بعد التقسيم

زمن النبضة الواحدة هو 1/Ftc1 وحتى يعد المؤقت من الصفر إلى القيمة الأعظمية (255) فإنه يحتاج إلى 255*1/Ftc1 وعند الوصول للقيمة الأعظمية يعود ليعد إلى القيمة صفر مستغرقاً نفس الزمن وبالتالي فإن الدور لدورة واحدة من دورات المؤقت (الموجة المثلثية) هو :

$$T = 2 * 255 * 1/Ftc1$$

مسجلات عرض النبضة :

المسجل TCCR1A و TCCR1B :

يعمل هذان المسجلان مع بعض لتشكيل أنماط ال PWM المختلفة

Timer/Counter 1 Control Register A – TCCR1A

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	W	W	R/W	R/W	

Timer/Counter 1 Control Register B – TCCR1B

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

نمط مقارن PWM ثابت Compare Output Mode, Fast PWM

الحالـــــة	COM1A1 COM1B1	COM1A0 COMB0
النمط العمل الطبيعي OC1A وOC1B مفصولين	0	**0**
WGM13:0=15 تغيير حالة OC1A عند المقارنة فصل OC1B (نمط العمل الطبيعي للبوابة) ولكافة إعدادات WGM1 البوابة في حالة العمل الطبيعي OC1A و OC1B مفصولين	0	**1**
مسح OC1A وOC1B عند المقارنة وضع OC1A وOC1B للأسفل (بدون قلب الإشارة)	1	**0**
وضع واحد OC1A وOC1B عند المقارنة مسح OC1A وOC1B للأسفل (مع قلب الإشارة)	**1**	**1**

نمط التالي --- < Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM

الحالـــــة	COM1A1 COM1B1	COM1A0 COMB0
النمط العمل الطبيعي OC1A وOC1B مفصولين	0	**0**
WGM13:0=15 تغيير حالة OC1A عند المقارنة فصل OC1B (نمط العمل الطبيعي للبوابة) ولكافة إعدادات WGM1 البوابة في حالة العمل الطبيعي OC1A و OC1B مفصولين	0	**1**
مسح OC1A وOC1B عند المقارنة عند العد التصاعدي ووضع OC1A وOC1B للأسفل عند العد التنازلي	1	**0**
وضع واحد OC1A وOC1B عند المقارنة عند العد التصاعدي مسح OC1A وOC1B للأسفل عند العد التنازلي	**1**	**1**

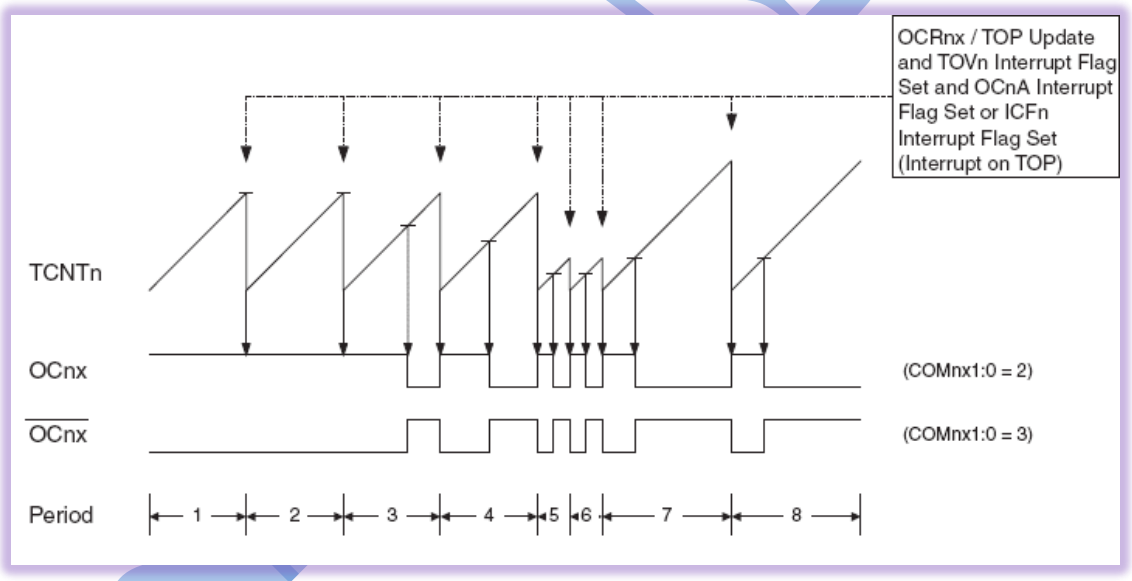
حيث الخانات WGM10 WGM11 WGM12 WGM13 مع بعضها أنماط عمله وفق الجدول التالي

وضع علم TOV1	تغير OCR1x	القمة	عمل النمط	WGM10 (PWM10)	WGM11 (PWM11)	WGM12 (CTC1)	WGM13	النمط
أعظمي	مباشر	0xFFFF	عادي	0	0	0	0	**0**
الأسفل	الأعلى	0x00FF	PWM, Phase Correct, 8-bit	1	0	0	0	**1**
الأسفل	الأعلى	0x01FF	PWM, Phase Correct, 9-bit	0	1	0	0	**2**
الأسفل	الأعلى	0x03FF	PWM, Phase Correct, 10-bit	1	1	0	0	**3**
أعظمي	مباشر	OCR1A	CTC	0	0	1	0	**4**
الأعلى	الأسفل	0x00FF	FAST PWM 8 BIT	1	0	1	0	**5**
الأعلى	الأسفل	0x01FF	FAST PWM 9 BIT	0	1	1	0	**6**
الأعلى	الأسفل	0x03FF	FAST PWM	1	1	1	0	**7**

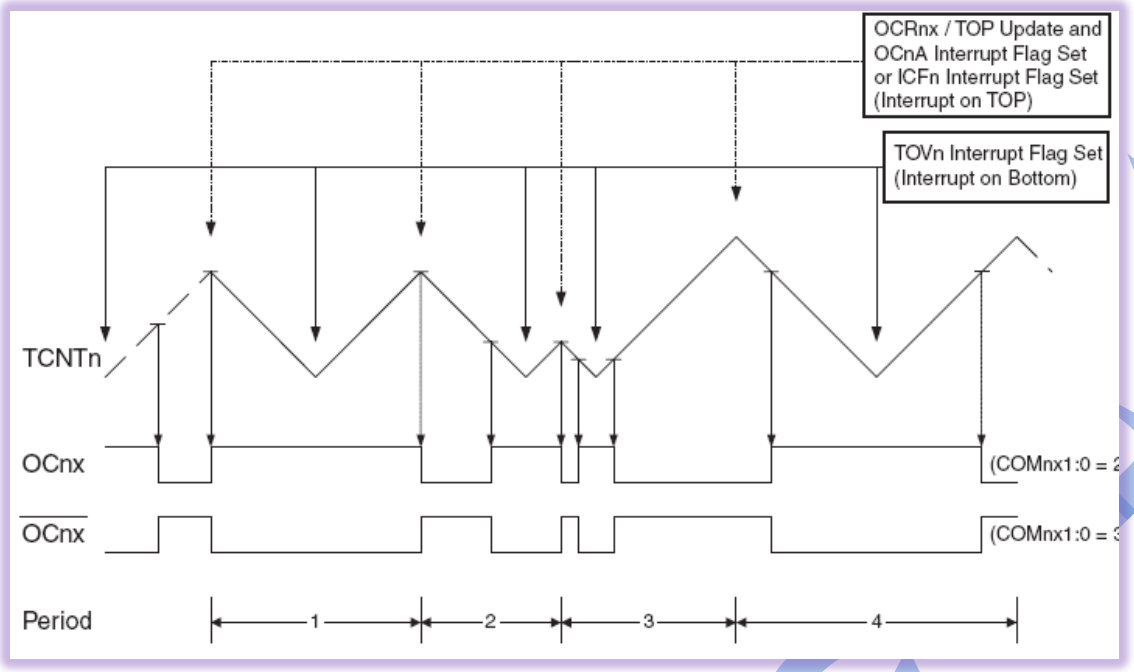
10 BIT								
الأسفل	الأسفل	ICR1	PWM, Phase and Frequency Correct	0	0	0	1	**8**
الأسفل	الأسفل	OCR1A	PWM, Phase and Frequency Correct	1	0	0	1	**9**
الأسفل	الأعلى	ICR1	PWM, Phase Correct	0	1	0	1	**10**
الأسفل	الأعلى	OCR1A	PWM, Phase Correct	1	1	0	1	**11**
أعظمي	مباشر	ICR1	CTC	0	0	1	1	**12**
-	-	-	محجوز	1	0	1	1	**13**
الأعلى	الأسفل	ICR1	Fast PWM	0	1	1	1	**14**
الأعلى	الأسفل	OCR1A	Fast PWM	1	1	1	1	**15**

حيث يعني الأعلى أعظم قيمة للمسجل عندها يصبح 0xff أما الأسفل أقل قيمة وصل إليها 0x00 حسب الدقة أما أعظمي فيعني القيمة الأعظمية المخزنة في المسجل المقارنة حيث الأنماط 3 2 1 نمط PWM طور معدل بدقات مختلفة للأعلى , النمط 4 نمط مقارن , الأنماط 7 6 5 نمط PWM طور ثابت على حاله , الأنماط 9 8 PWM طور وتردد معدل , النمطين 11 10 PWM طور معدل , النمط CTC نمط مقارن , النمطين 15 14 PWM ثابت

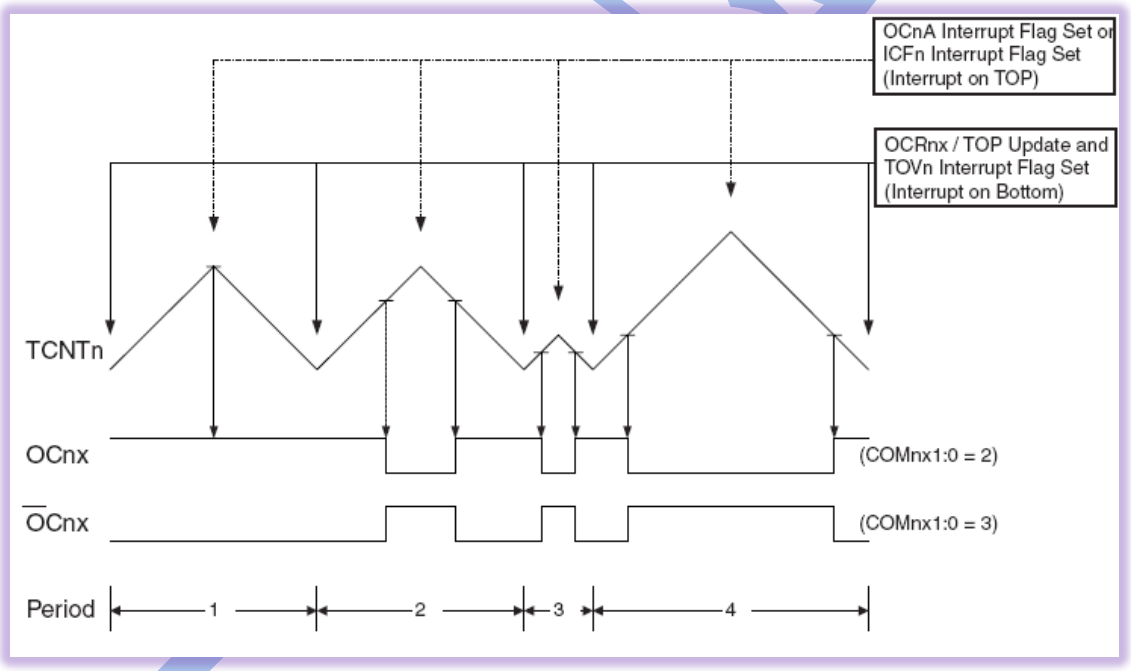
وهذه أنماط عمل ال pwm



Fast PWM Mode, Timing Diagram



Phase Correct PWM Mode, Timing Diagram



Phase and Frequency Correct PWM Mode, Timing Diagram

الخانتين 2 و 3 من المسجل TCCR1A خانتي قوة المقارن وتعملان على تغيير قيم الخانات COM1x1:0
عند حصول مقارنة هذه الخانات تتطابق مع FOC1A و FOC1B هذه الخانتين لا تولد أي مقاطعة عند حصول المقارنة ولا تمسح المؤقت على الوضع CTC مستخدم OCR1A كوضع TOP الأعلى

هذه الخانتين دائما توضع صفر

المؤقت / العداد 2 :

يحتوي هذا المؤقت على عدة أشياء :

1 – قناة عد وحيدة

2 – مسح المؤقت عند قيمة المقارنة

3 – يتألف من 8 بت

4 – مقسم تردد عدة تقسيمات

5 – مقاطعة طفحان العداد ومقاطعة المقارن

6 – نمطين PWM

1 – عمله كمؤقت وعداد

المسجل TIMSK :

Timer/Counter Interrupt Mask Register – TIMSK

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

2 - حيث يتم تفعيل مقاطعة الطفحان TOIE2

3 - حيث يتم تفعيل مقاطعة المقارنة OCIE2

المسجل TCCR2 :

نلاحظ أنه يختلف عن السابق حيث الخانات الأخيرة لاختيار الجبهة هنا لا يوجد حالة للعد الخارجي

Timer/Counter Control Register – TCCR2

Bit	7	6	5	4	3	2	1	0	
	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	TCCR2
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

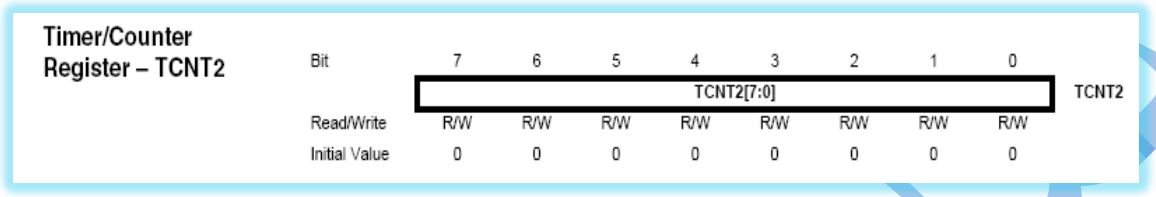
الخانات 0 و1 و2 :

الشرح	مؤقت / عداد	CS22	CS21	CS20
نمط توقف المؤقت / العداد	حجب	0	0	0
CK	مؤقت	0	0	1
CK / 8	مؤقت	0	1	0
CK / 32	مؤقت	0	1	1
CK / 64	مؤقت	1	0	0

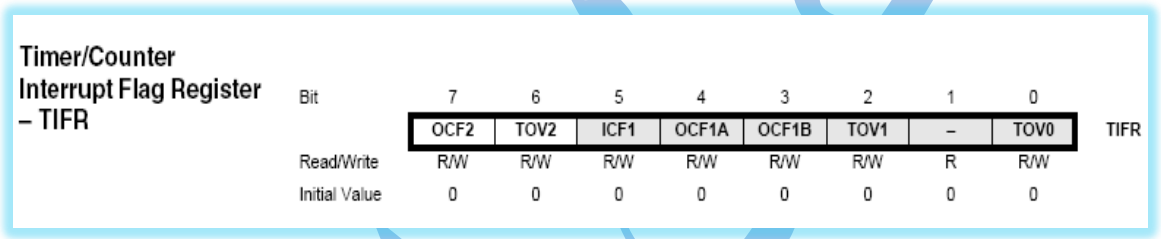
CK / 128	مؤقت	1	0	1
CK / 256	عداد	1	1	0
CK / 1024	عداد	1	1	1

المسجل TCNT2 :

وهو عبارة عن مسجل تحكم عرض 8 بت يتم فيه تخزين القيمة الحالية (الآنية) التي وصل إليها المؤقت / العداد 2 وباعتبار أنه 8 بت فإن أعظم قيمة يمكن تخزينها هي 256 = $FF



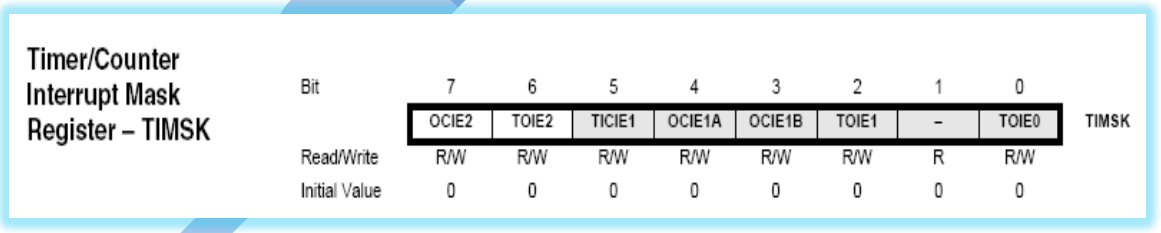
المسجل TIFR :



الخانة 6 :علم المقاطعة لطفحان المؤقت الثاني

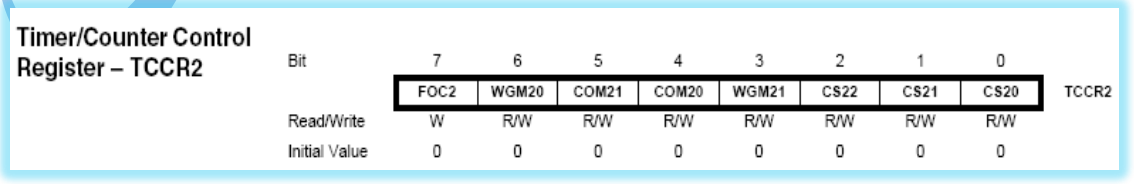
2 - عمله كمقارن

المسجل TIMSK :



1 - حيث يتم تفعيل مقاطعة المقارنة OCIE2

المسجل TCCR2 :



الخانتين 3 و6 : حيث يعمل كمقارن وفق الجدول التالي في النمط الثاني

النمط	WGM21 (CTC2)	WGM20 (PWM2)	OPERATION	TOP	UPDATE OCR2	TOV2 FLAG SET
0	0	0	NORMAL	0XFF	مباشر	MAX
1	0	1	PWM PHASE CORRET	0XFF	TOP	BOTTOM
2	1	0	CTC	OCR2	مباشر	MAX
3	1	1	FAST PWM	0XFF	BOTTOM	MAX

الخانتين 4 و5 :وذلك لتفعيل القطب الخارجي الخاص بالمقارنة OC2 لوضع القيم التالية عليه

العمل	COM21	COM20
لاشيء مفصول	0	**0**
تغيير OC2 عند حصول مقارنة	0	**1**
وضع صفر OC2 عند حصول مقارنة	1	**0**
وضع واحد OC2 عند حصول مقارنة	1	**1**

المسجل OCR2 :

وهو عبارة عن مسجل تحكم عرض 8 بت يتم فيه تخزين القيمة الي سوف يقارن معها

Output Compare Register – OCR2

Bit	7	6	5	4	3	2	1	0	
	OCR2[7:0]								OCR2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

المسجل TIFR :

Timer/Counter Interrupt Flag Register – TIFR

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	–	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

الخانة 7 :علم المقاطعة للمقارن

3 – عمله كمولد عرض النبضة :

عندما يتم وضع المؤقت / العداد في وضع الـ PWM فإن مسجل المؤقت / عداد 2 TCNT2 و مسجل مقارنة الخرج OCR2 يشكلان مولداً لنبضات معدلة .

حيث يوضع قيمة في المسجل TCNT2 ثم يوضع قيمة متغيرة في المسجل OCR2 ليتم تعديلها معها

المسجل TCCR2 :

Timer/Counter Control Register – TCCR2

Bit	7	6	5	4	3	2	1	0	
	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	TCCR2
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

الخانتين 3 و6 : حيث يعمل كمقارن وفق الجدول التالي في النمطين الأول والثالث

النمط	WGM21 (CTC2)	WGM20 (PWM2)	OPERATION	TOP	UPDATE OCR2	TOV2 FLAG SET
0	0	0	NORMAL	0XFF	مباشر	MAX
1	0	1	PWM PHASE CORRET	0XFF	TOP	BOTTOM
2	1	0	CTC	OCR2	مباشر	MAX
3	1	1	FAST PWM	0XFF	BOTTOM	MAX

الخانتين 4 و5 :وذلك لتفعيل القطب الخارجي الخاص OC2 نمط العمل FAST PWM MODE

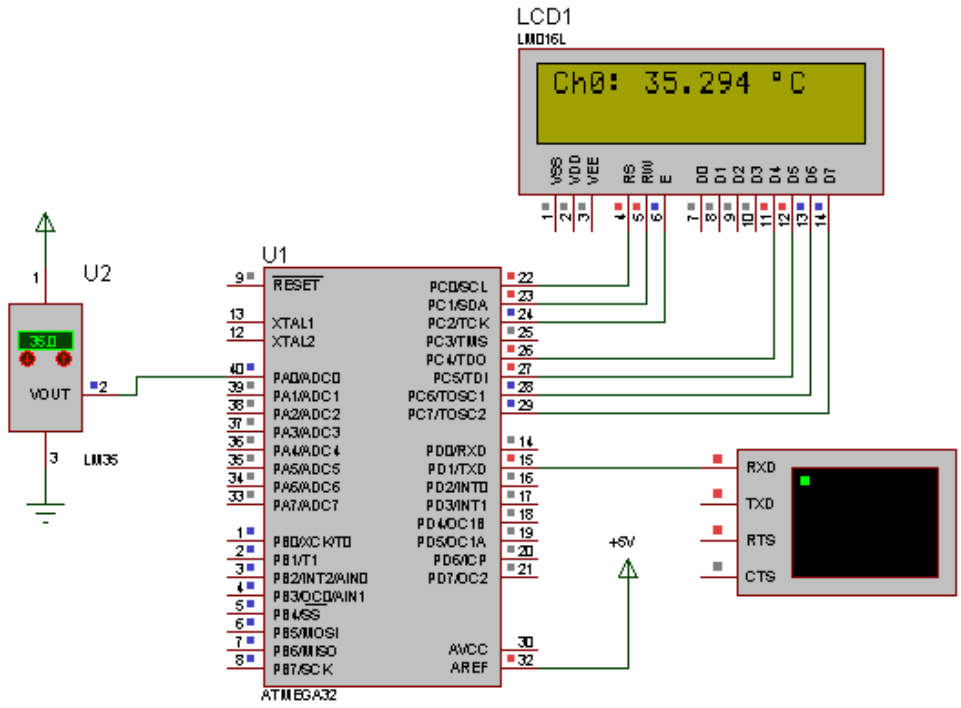
العمل	COM21	COM20
لاشيء مفصول	0	0
محجوز	0	1
وضع صفر OC2 عند حصول مقارنة ووضع واحد ل OC2 للأسفل (بدون قلب)	1	0
وضع واحد OC2 عند حصول مقارنة ووضع صفر OC2 (مع قلب)	1	1

الخانتين 4 و5 :وذلك لتفعيل القطب الخارجي الخاص OC2 نمط العمل PHASE CORRECT PWM MODE

العمل	COM21	COM20
لاشيء مفصول	0	0
محجوز	0	1
وضع صفر OC2 عند حصول مقارنة عندما يعد للأعلى ووضع واحد OC2 عندما المقارن يعد للأسفل	1	0
وضع واحد OC2 عند حصول مقارنة عندما يعد للأعلى ووضع صفر OC2 عندما المقارن يعد للأسفل	1	1

امثلة عامة

مثال 1 :حساس lm35 مع atmega32



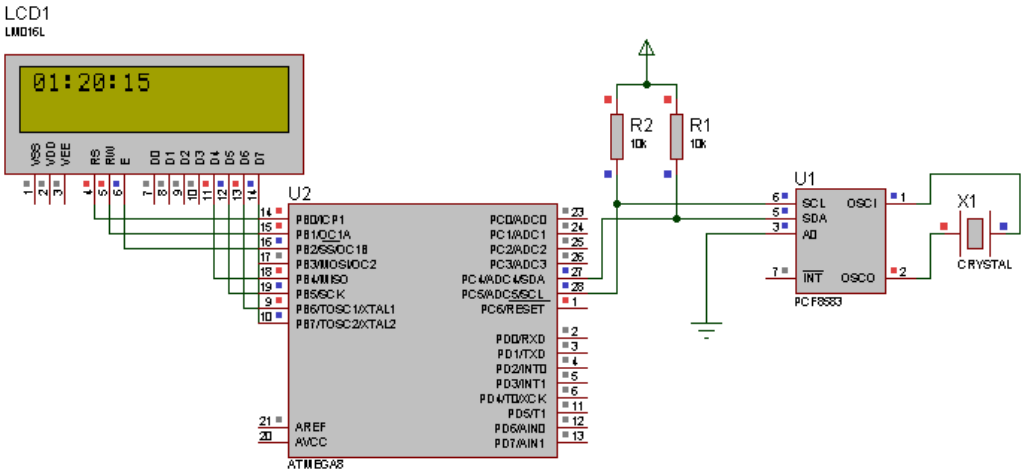
الكود :

```
#include <mega32.h>
#include <delay.h>
#include <math.h>
#include <stdio.h>
#define ADC_VREF_TYPE 0x20
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input|ADC_VREF_TYPE;
ADCSRA|=0x40;
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}
#asm
  .equ __lcd_port=0x15 ;PORTC
```

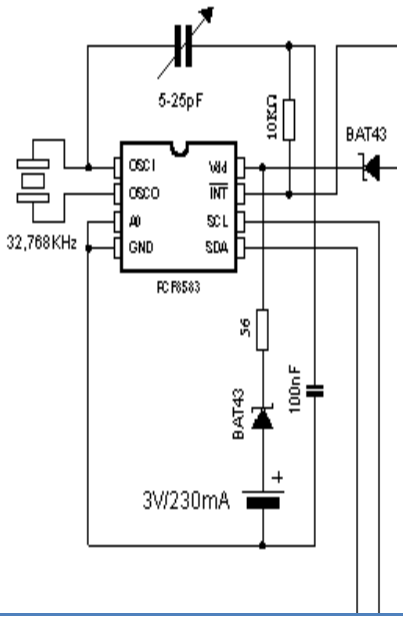
```
#endasm
#include <lcd.h>
void main(void)
{
char str[3];
float flt;
unsigned char res;
PORTB=0x00;
DDRB=0xFF;
// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: Off
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x08;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x33;
ACSR=0x80;
SFIOR=0x00;
// ADC initialization
// ADC Clock frequency: 125.000 kHz
// ADC Voltage Reference: AREF pin
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x86;

lcd_init(16);
ADCSRA = 0xFF;
```

```
while (1)
    {
      res = read_adc(0);
      flt = (float)res/51;
      flt=flt*100;
      sprintf(str,"Ch0: %2.3f %cC",flt,0xdf);
      printf("\n\r%s",str);
      delay_ms(500);
      lcd_clear();
      lcd_puts(str);
     };
}
```

مثال 2 : حساس PCF8583 للتاريخ واليوم بدقة عالية



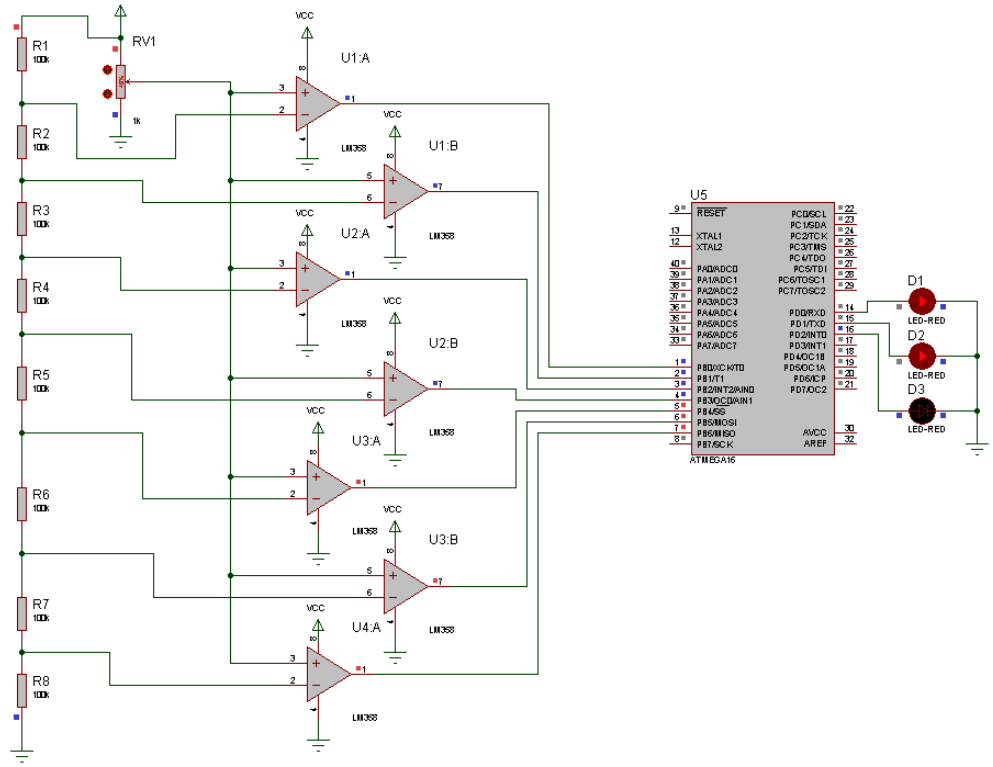
العناصر الواجب وصلها مع **PCF8583**
والبطارية لكي لا تتوقف الساعة أثناء انقطاع
الكهرباء يرجى الاطلاع على **SHEET DATA**




```
#include <mega8.h>
#include <math.h>
#include <stdio.h>
#include <delay.h>
#asm
  .equ __lcd_port=0x18;
#endasm
#include <lcd.h>
#asm
   .equ __i2c_port=0x15
   .equ __sda_bit=4
   .equ __scl_bit=5
#endasm
#include <pcf8583.h>
char lcd_buffer[33];
void main(void)
{
unsigned char h,m,s,hs;
unsigned char *date,*month;
unsigned  *year;
lcd_init(20);
```

```
i2c_init();
rtc_init(0,0);
for(;;)
{
    rtc_get_time(0,&h,&m,&s,&hs);
    rtc_get_date(0,  &date,  &month, &year);
    sprintf(lcd_buffer,"%02d:%02d:%02d  %02d/%02d/%02d",h, m, s, date, month,year);
    lcd_clear();
    lcd_puts(lcd_buffer);
    delay_ms(100);
}
}
```

مثال 3 :

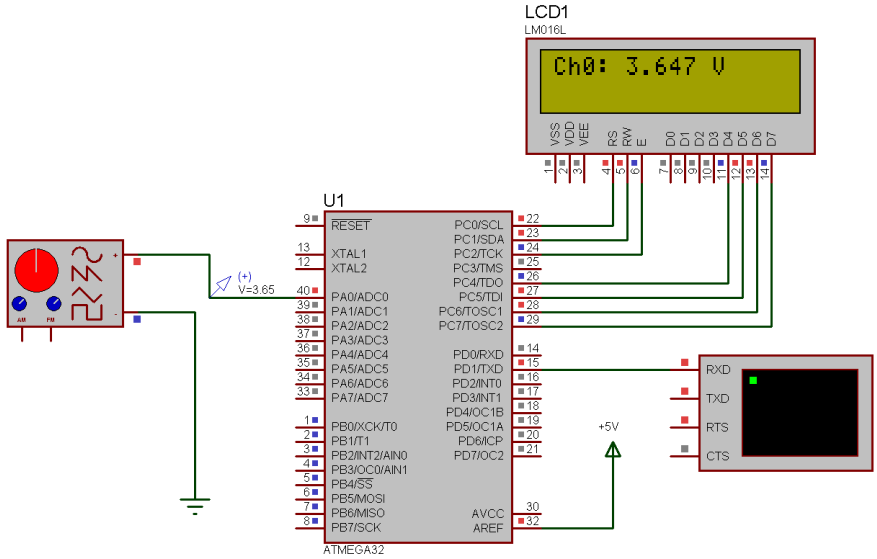


الكود :

```
#include <mega16.h>
```

```
void main(void)
{

char output;
PORTB=0x7F;
DDRB=0x00;
PORTD=0x00;
DDRD=0x07;
ACSR=0x80;
SFIOR=0x00;
while (1)
    {
    output = 0 ;
    if(PINB.0 == 1)
            output++;
    if(PINB.1 == 1)
            output++;
    if(PINB.2 == 1)
            output++;
    if(PINB.3 == 1)
            output++;
    if(PINB.4 == 1)
            output++;
    if(PINB.5 == 1)
            output++;
    if(PINB.6 == 1)
            output++;
    PORTD = output;
    };
}
```

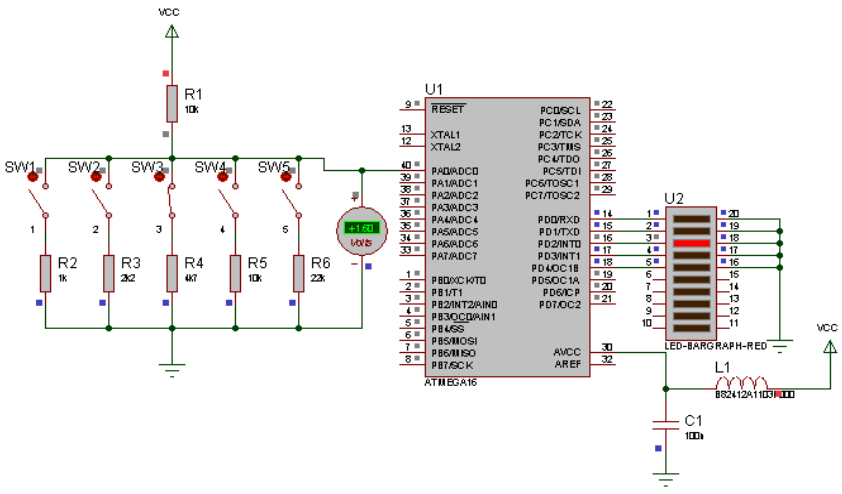
مثال 4 :




```
#include <mega16.h>
#include <stdio.h>
#include <delay.h>
#asm
  .equ __lcd_port=0x12 ;PORTD
#endasm
#include <lcd.h>
#define ADC_VREF_TYPE 0x40

unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input|ADC_VREF_TYPE;
ADCSRA|=0x40;
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}

void main(void)
{
```

```
char lcd_show[16];
float volt;
ACSR=0x80;
SFIOR=0x00;
// ADC initialization
// ADC Clock frequency: 125.000 kHz
// ADC Voltage Reference: AVCC pin
// ADC Auto Trigger Source: None
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x86;
lcd_init(16);
while (1)
    {
    volt = read_adc(0);
    volt = (volt / 1024) * 5;
    sprintf(lcd_show,"%.2f" , volt);
    lcd_clear();
    lcd_puts(lcd_show);
    delay_ms(100);
    };
}
```

مثال 4 : analog keys

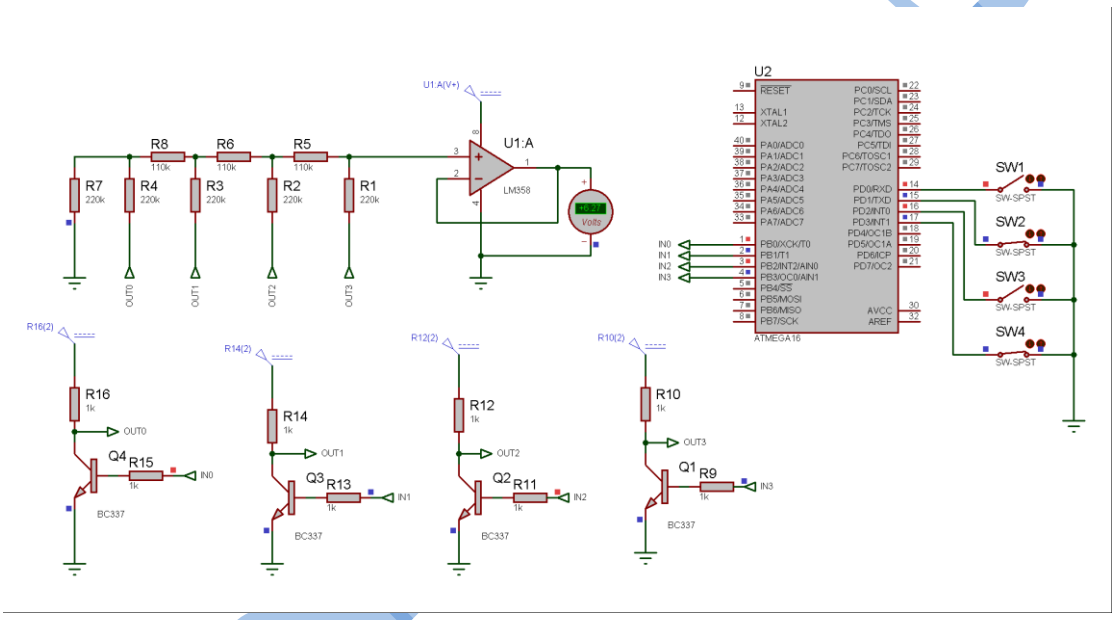


الكود

```
#include <mega16.h>
#include <delay.h>hg
#define ADC_VREF_TYPE 0x40
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input|ADC_VREF_TYPE;
ADCSRA|=0x40;
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}
void main(void)
{
float vkey;
PORTD=0x00;
DDRD=0x1F;
ACSR=0x80;
SFIOR=0x00;
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x86;
while (1)
    {
    vkey = read_adc(0);
    vkey = vkey * 5 /1024;
    if(vkey < 3.6 && vkey > 3.3)
       PORTD.0 = 1;
    else if(vkey < 2.6 && vkey > 2.4)
       PORTD.1 = 1;
    else if(vkey < 1.8 && vkey > 1.4)
       PORTD.2 = 1;
```

```
    else if(vkey < 1.01 && vkey > 0.81)
       PORTD.3 = 1;
    else if(vkey < 0.55 && vkey > 0.35)
       PORTD.4 = 1;
    else
       PORTD = 0;
    delay_ms(10);
    };
}
```

مثال 5: dac محول رقمي تشابهي

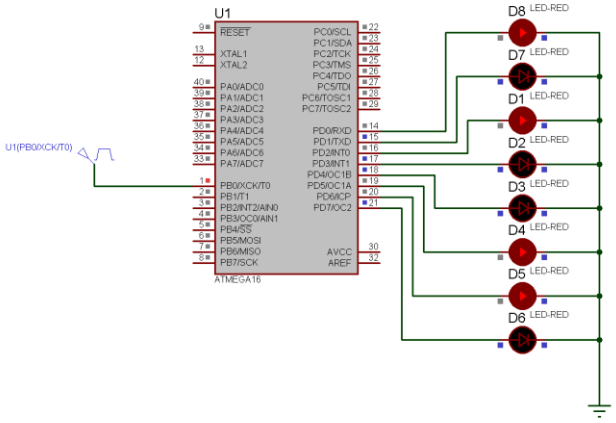


الكود :

```
#include <mega16.h>
void main(void)
{
PORTB=0x00;
DDRB=0x0F;
PORTD=0x0F;
DDRD=0x00;
ACSR=0x80;
```

```
SFIOR=0x00;


while (1)
    {
    PORTB = PIND ;
    };
}
```

مثال 6 : DIGITAL FREGUENCY

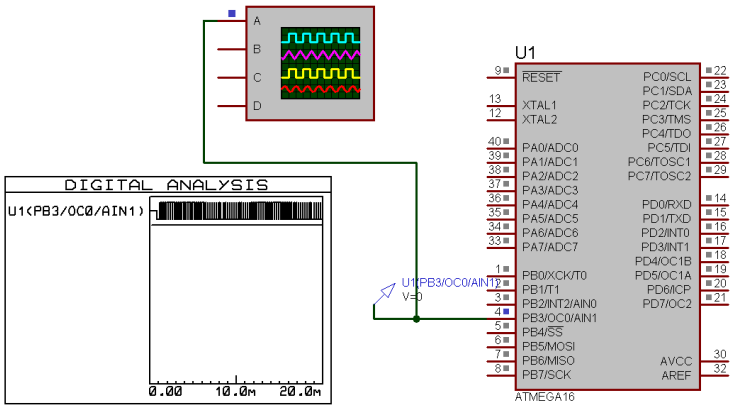


الكود :

```
#include <mega16.h>
#include <delay.h>
void main(void)
{
PORTD=0x00;
DDRD=0xFF;
TCCR0=0x07;
TCNT0=0x00;
OCR0=0x00;
TIMSK=0x01;
ACSR=0x80;
```

```
SFIOR=0x00;
#asm("sei")
while (1)
    {
    PORTD = TCNT0;
    };
}
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
int i;
for (i = 0 ; i < 3 ; i++){
            PORTD = 0X00;
            delay_ms(200);
            PORTD = 0xFF;
            delay_ms(200);
}
TCNT0 = 0;
}
```

مثال 7 : fast pwm

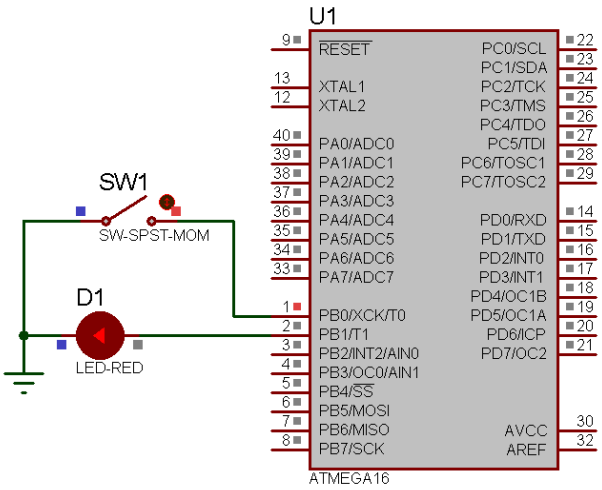


الكود :

```
#include <mega16.h>
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
static unsigned char counter = 0;
counter++;
if (counter < 20)
        OCR0 = 0x35;
else if(counter < 40)
        OCR0 = 0xF0;
else
        counter = 0;
}
void main(void)
{
PORTB=0x00;
DDRB=0x08;
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 1000.000 kHz
// Mode: Fast PWM top=FFh
// OC0 output: Inverted PWM
TCCR0=0x7A;
TCNT0=0x00;
OCR0=0x35;
ACSR=0x80;
SFIOR=0x00;
#asm("sei")

while (1)
    {
```

```
    };
}
```

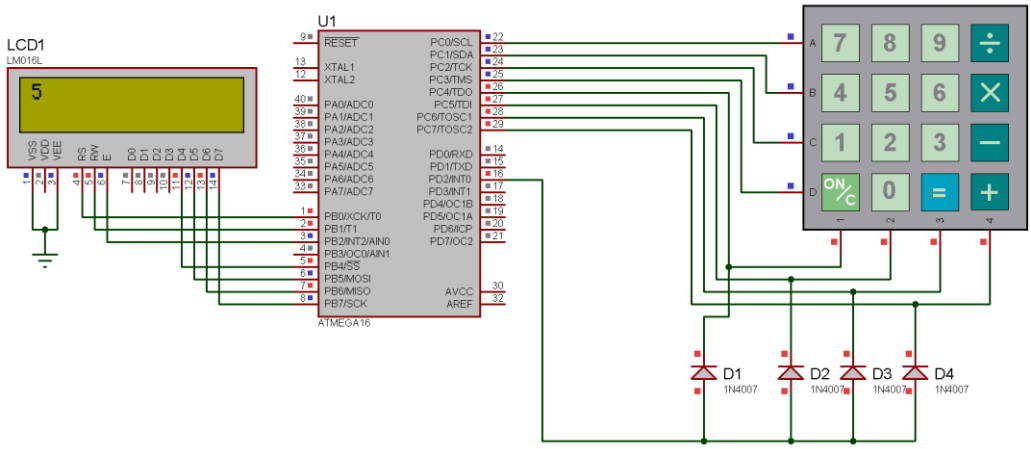
مثال 8 :LED



الكود :

```
#include <mega16.h>
void main(void)
{
PORTB=0x03;
DDRB=0x02;
ACSR=0x80;
SFIOR=0x00;
while (1)
    {
    if (PINB.0 == 0)
        PORTB.1 = 0;
    else
        PORTB.1 = 1;
    };
}
```

مثال 9 : KEYPAD

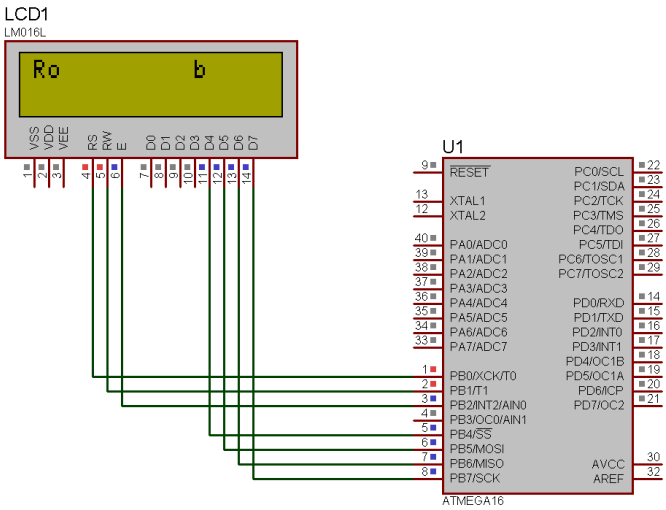



```
#include <mega16.h>
#include <delay.h>
#include <stdio.h>
#asm
   .equ __lcd_port=0x18 ;PORTB
#endasm
#include <lcd.h>
bit check;
char key_code[4][4] = {{'7' , '4' , '1' , 'C'} , { '8' , '5', '2' , '0'} , { '9' , '6' , '3' , '='} , {'/' , '*' , '-' , '+'}};
char lcd_show[16];
interrupt [EXT_INT0] void ext_int0_isr(void)
{
    char x , y;
    if ((PINC & 0xF0) != 0xF0){
      if(PINC.4 == 0)
          x = 0;
       else if(PINC.5 == 0)
          x = 1;
```

```
else if(PINC.6 == 0)
   x = 2;
else if(PINC.7 == 0)
   x = 3;
check = 1;
                  if (check){
   PORTC.0 = 1;
   if((PINC & 0xF0) == 0xF0){
      y = 0;
      check = 0;
   }
}

                  if (check){
   PORTC.1 = 1;
   if((PINC & 0xF0) == 0xF0){
      y = 1;
      check = 0;
   }
}
                  if (check){
   PORTC.2 = 1;
   if((PINC & 0xF0) == 0xF0){
      y = 2;
      check = 0;
   }
}
                  if (check){
   PORTC.3 = 1;
   if((PINC & 0xF0) == 0xF0){
      y = 3;
      check = 0;
```

```
            }


          }
      lcd_clear();
      sprintf(lcd_show , "%c" , key_code[x][y]);
      lcd_puts(lcd_show);
      delay_ms(100);
      PORTC = PORTC & 0xF0;
      }
}
void main(void)
{
PORTC=0xF0;
DDRC=0x0F;
GICR|=0x40;
MCUCR=0x02;
MCUCSR=0x00;
GIFR=0x40;
TIMSK=0x00;
ACSR=0x80;
SFIOR=0x00;
lcd_init(16);
#asm("sei")
while (1)
      {
      };
}
```

مثال 10 : LCD

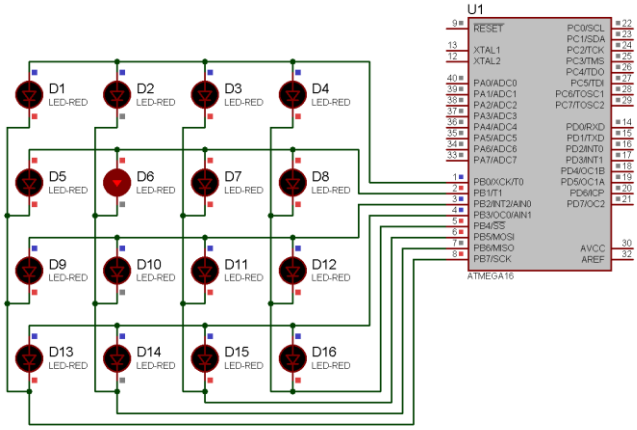


الكود :

```
#include <mega16.h>
#include <string.h>
#include <delay.h>
#asm
   .equ __lcd_port=0x18 ;PORTB
#endasm
#include <lcd.h>
void main(void)
{
char string[] = "Robotic Center";
int str_length;
int i , j;
ACSR=0x80;
SFIOR=0x00;
lcd_init(16);
str_length = strlen(string);
while (1)
    {
```

```
    for(i = 0 ; i < str_length ; i++)
    {
      if(string[i] != ' ')
      {
        for(j = 15 ; j >= i ; j--)
        {
          lcd_gotoxy(j , 0);
          lcd_putchar(string[i]);
          lcd_putchar(' ');
          delay_ms(100);
        }
      }
    }
    delay_ms(1000);
    lcd_clear();
    };
}
```

مثال 11 : LED

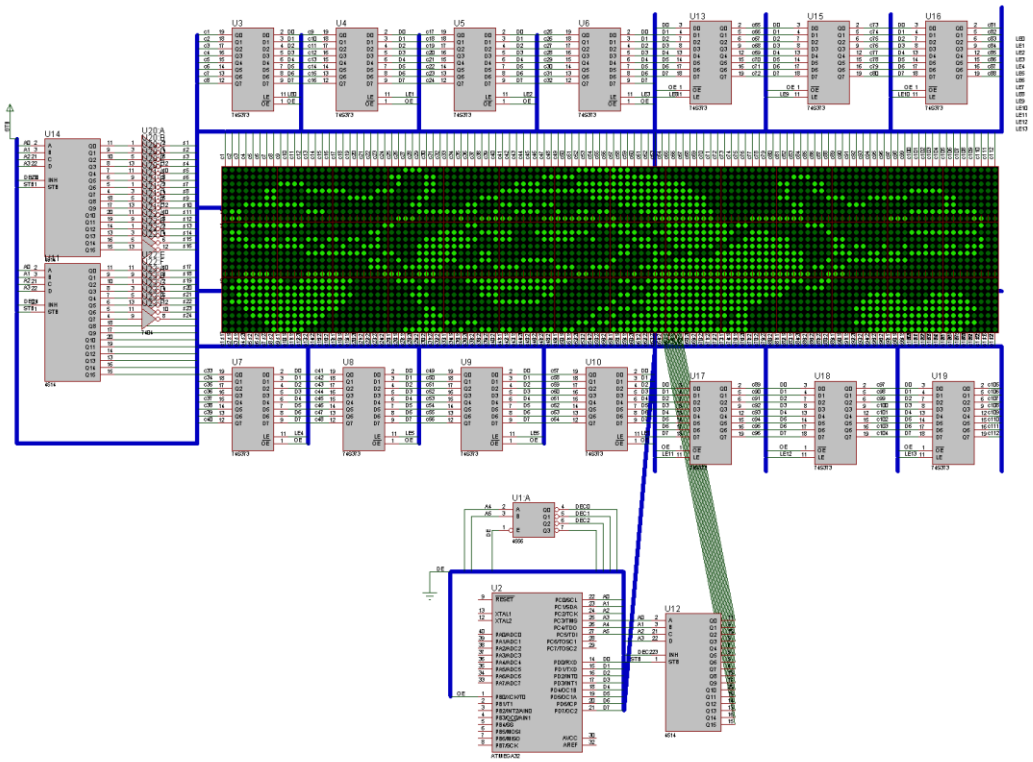


الكود :

```
#include <mega16.h>
#include <delay.h>
```

```
void main(void)
{
PORTB=0x00;
DDRB=0xFF;
ACSR=0x80;
SFIOR=0x00;
while (1)
    {
    PORTB = PORTB | 0xF0;
    PORTB.7 = 0;
    PORTB = PORTB & 0xF0;
    PORTB = PORTB | 0x01;
    delay_ms(1000);
    PORTB = PORTB | 0xF0;
    PORTB.6 = 0;
    PORTB = PORTB & 0xF0;
    PORTB = PORTB | 0x02;
    delay_ms(1000);
    PORTB = PORTB | 0xF0;
    PORTB.5 = 0;
    PORTB = PORTB & 0xF0;
    PORTB = PORTB | 0x04;
    delay_ms(1000);
    PORTB = PORTB | 0xF0;
    PORTB.4 = 0;
    PORTB = PORTB & 0xF0;
    PORTB = PORTB | 0x08;
    delay_ms(1000);
    };
}
```

مثال 12 : جريدة ثابتة أو متحركة حسب التطوير



الكود :

```
#include <mega32.h>

#include <delay.h>

flash unsigned char PicData[14][24]=

        {

        {0xFF,0xFF,0xFF,0x3F,0x3F,0x7,0xF9,0xF9,0x7,0xF9,0x31,0x7,0xFF,0xFF,0xFF,0x87,0x3,0x1,0x3,0x1,0x7,0xF,0x3F,0x7F},

        {0xFF,0xFF,0xF0,0xF,0xFF,0xF0,0x8F,0xFF,0xF0,0xFF,0x49,0xB0,0xF,0xFF,0xFF,0xF,0x6,0x0,0x0,0x0,0x0,0x0,0x80,0xC0,0xF0},

        {0xFF,0xFF,0xFF,0x3C,0xC3,0xFF,0xFF,0xFF,0x7F,0x9F,0xB2,0x8D,0x60,0x9F,0x1F,0x7E,0xF8,0xFC,0xF8,0xFC,0xFE,0xFF,0xFF,0xFF},

        {0xE3,0x9C,0x9D,0x9E,0xCF,0xE7,0xF,0xF1,0xFE,0x7F,0x1F,0xF,0xE,0xD,0x3,0x8C,0x91,0xEF,0xC7,0xEF,0x9F,0x1F,0x8F,0x3},

        {0xFF,0xFF,0xFF,0xFF,0x1F,0xE7,0x60,0x1B,0xE7,0x18,0x60,0xF0,0x9C,0x9C,0x63,0xFF,0x7,0x1F,0x1F,0x7F,0xFF,0xFF,0xE4,0x9B},

        {0xFF,0x3F,0xC3,0x0,0xFF,0x24,0xC3,0x3C,0xFF,0x1C,0x3C,0xFF,0xFF,0x7,0xF8,0xFF,0x0,0x0,0x3F,0xC0,0xF,0xFF,0xFF,0x24},
```

{0x7,0xF8,0x7,0x0,0x1F,0xFF,0x26,0xC0,0x3F,0xE0,0xD8,0xFF,0xFF,0xFF,0x18,0xE7,0xFC,0xFF,0xFF,0xFF,0x3F,0xDF,0x27,0xD9},

{0xF8,0x80,0x0,0x0,0x0,0x0,0x1,0xF,0x3E,0xC8,0x31,0xC7,0x89,0x39,0xBE,0xF,0x37,0x19,0x27,0x19,0x6,0x19,0x6,0x1,0x0},

{0xFF,0xFF,0xF8,0xF0,0xC0,0xC0,0x80,0x80,0x0,0x0,0x0,0x0,0x81,0x76,0x1,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x80,0x0},

{0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF,0xF2,0x8C,0x6,0x1,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0xC0,0xF0,0xF2,0xF1,0xF0},

{0xFF,0xE3,0x9C,0x9D,0x7C,0xF3,0xF3,0xF0,0x8F,0x7F,0xFC,0xFC,0xF0,0x70,0x90,0x60,0x30,0x8C,0xFE,0xFF,0xFF,0xFF,0xFF,0xFF},

{0xFF,0xFF,0xFF,0xFF,0x1C,0xE3,0xFF,0xFF,0xFF,0xFF,0xFC,0x64,0x98,0x3,0xFC,0xFC,0xFF,0xFF,0x1F,0xF,0x7,0xF,0x7,0x1F},

{0xFF,0xFF,0xFF,0x7,0xF8,0xFF,0x7,0xF8,0xFF,0x7,0xFF,0xDB,0x4,0xF8,0xFF,0xFF,0xFF,0xFF,0x3C,0x18,0x0,0x0,0x0,0x0,0x0},

{0xFF,0xFF,0xFF,0xFF,0xFC,0xFC,0xE0,0xCF,0xDF,0xE0,0xDF,0xC4,0xE0,0xFF,0xFF,0xFF,0xFF,0xFF,0xF8,0xF0,0xE0,0xF0,0xE0,0xF8}

};

unsigned char satr=0;/*ãÊÛíÑ ÔãÇÑäÅå ÓØÑ*/

unsigned char hour=12,minute=58,second=30,weekday=6,monthday=7,monthyear=5,timedigit[9];

unsigned int year=1386;

unsigned char char1;

unsigned char char2;

interrupt [TIM1_OVF] void timer1_ovf_isr(void)

{

  unsigned char i=0;

  TCNT1H=0xd1;

  TCNT1L=0x1f;

  PORTB.0=1;//OE inactive

  PORTD=0xff-PicData[0][satr];

  PORTC=0x20|i;

  PORTC=PORTC|0x30;

  i++;

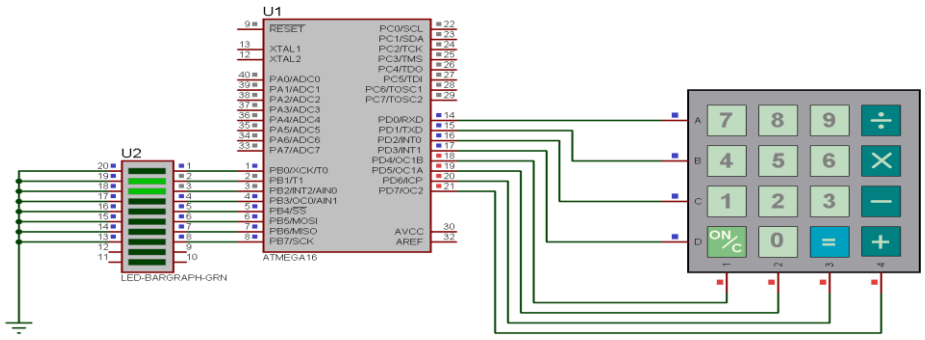
```
PORTD=0xff-PicData[1][satr];
PORTC=0x20|i;
PORTC=PORTC|0x30;
i++;
PORTD=0xff-PicData[2][satr];
PORTC=0x20|i;
PORTC=PORTC|0x30;
i++;
PORTD=0xff-PicData[3][satr];
PORTC=0x20|i;
PORTC=PORTC|0x30;
i++;
PORTD=0xff-PicData[4][satr];
PORTC=0x20|i;
PORTC=PORTC|0x30;
i++;
PORTD=0xff-PicData[5][satr];
PORTC=0x20|i;
PORTC=PORTC|0x30;
i++;
PORTD=0xff-PicData[6][satr];
PORTC=0x20|i;
PORTC=PORTC|0x30;
i++;
PORTD=0xff-PicData[7][satr];
PORTC=0x20|i;
PORTC=PORTC|0x30;
i++;
PORTD=0xff-PicData[8][satr];
PORTC=0x20|i;
PORTC=PORTC|0x30;
i++;
PORTD=0xff-PicData[9][satr];
```

```
  PORTC=0x20|i;
  PORTC=PORTC|0x30;
  i++;
  PORTD=0xff-PicData[10][satr];
  PORTC=0x20|i;
  PORTC=PORTC|0x30;
  i++;
  PORTD=0xff-PicData[11][satr];
  PORTC=0x20|i;
  PORTC=PORTC|0x30;
  i++;
  PORTD=0xff-PicData[12][satr];
  PORTC=0x20|i;
  PORTC=PORTC|0x30;
  i++;
  PORTD=0xff-PicData[13][satr];
  PORTC=0x20|i;
  PORTC=PORTC|0x30;
  i++;
  PORTB.0=0;
  PORTC=satr;
  satr++;
  if(satr==24)
    satr=0;
}
// Timer 2 overflow interrupt service routine
interrupt [TIM2_OVF] void timer2_ovf_isr(void)
{

}
// Declare your global variables here
void main(void)
{
```

```
PORTA=0x00;
DDRA=0xff;
PORTB=0x00;
DDRB=0x01;
PORTC=0x00;
DDRC=0xff;
PORTD=0x00;
DDRD=0xff;
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x81;
TCNT1H=0xa2;
TCNT1L=0x3f;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
// Timer/Counter 2 initialization
```

```
// Clock source: TOSC1 pin
// Clock value: PCK2/128
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x08;
TCCR2=0x05;
TCNT2=0x00;
OCR2=0x00;
// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x44;
// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;
// Global enable interrupts
#asm("sei")
while (1);
}
```

مثال 13 : KEYPAD

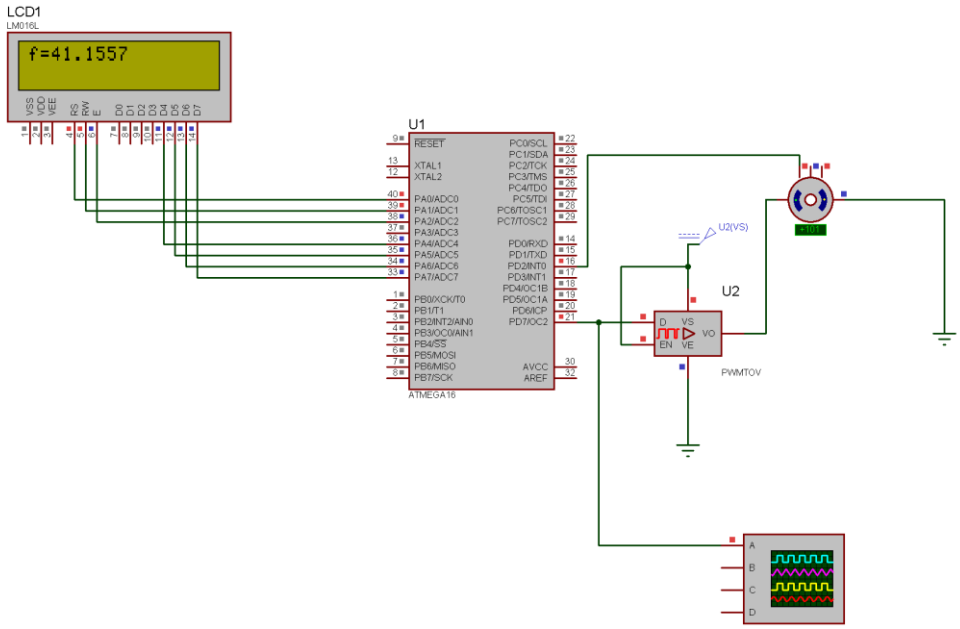


الكود :

```
#include <mega16.h>
bit check;
void main(void)
{
char x , y;
char key_code[4][4] = {{0 , 1 , 2 , 3} , { 4 , 5 , 6 , 7} , { 8 , 9 , 10 , 11} , {12 , 13 , 14 , 15}};
PORTB=0x00;
DDRB=0xFF;
PORTD=0xF0;
DDRD=0x0F;
ACSR=0x80;
SFIOR=0x00;
while (1)
    {
    PORTD = PORTD & 0xF0;
    if ((PIND & 0xF0) != 0xF0){
       if(PIND.4 == 0)
          x = 0;
       else if(PIND.5 == 0)
          x = 1;
       else if(PIND.6 == 0)
          x = 2;
       else if(PIND.7 == 0)
          x = 3;
       check = 1;
                    if (check){
       PORTD.0 = 1;
       if((PIND & 0xF0) == 0xF0){
          y = 0;
          check = 0;
```

```
    }
  }


                if (check){
  PORTD.1 = 1;
  if((PIND & 0xF0) == 0xF0){
    y = 1;
    check = 0;
  }
}
                if (check){
  PORTD.2 = 1;
  if((PIND & 0xF0) == 0xF0){
    y = 2;
    check = 0;
  }
}
                if (check){
  PORTD.3 = 1;
  if((PIND & 0xF0) == 0xF0){
    y = 3;
    check = 0;
  }

}
PORTB = key_code[x][y];
}
};
}
```

FREGUENCY: 14 مثال



الكود :

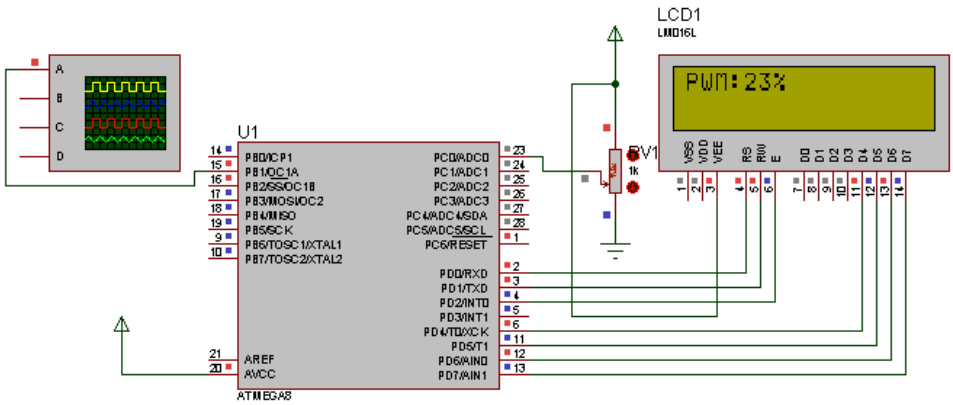
```
#include <mega16.h>
#asm
  .equ __lcd_port=0x1B ;PORTA
#endasm
#include <lcd.h>
#include <delay.h>
#include <stdio.h>
#include <math.h>
 float a,d=0,d1,f=0,i,de;
interrupt [EXT_INT0] void ext_int0_isr(void)
{
 a=TCNT1;
 TCNT1=0;
 }
interrupt [TIM2_OVF] void timer2_ovf_isr(void)
{
```

```
}
interrupt [TIM2_COMP] void timer2_comp_isr(void)
{
}
void main(void)
{
float t;
char lcd_buf[30],i;
PORTD=0x00;
DDRD=0x80;
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 500.000 kHz
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x02;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
```

```
// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: 500.000 kHz
// Mode: Fast PWM top=FFh
// OC2 output: Non-Inverted PWM
ASSR=0x00;
TCCR2=0x6A;
TCNT2=0x00;
OCR2=0x99;

// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Falling Edge
// INT1: Off
// INT2: Off
GICR|=0x40;
MCUCR=0x02;
MCUCSR=0x00;
GIFR=0x40;
ACSR=0x80;
SFIOR=0x00;
lcd_init(16);
#asm("sei")
while (1) {
  delay_ms(100);
  t=a*0.000002;
  f=1/t;
  sprintf(lcd_buf,"f=%5.4f ",f);
  lcd_clear();
  lcd_puts(lcd_buf);
  delay_ms(10);
```

```
  d1=d;
  //detect the error
  d=f-36;
  //i is the integral parameter
  i=d+d1;
  //de is the derive parameter
  de=d-d1;
  //and we assume the pwm by the combination of these three parameters
  OCR2-=((int)(d))-((int)(i*0.001))-((int)(de*0.1));
// Place your code here
    };
}
```

مثال 15: PWM AND ADC



الكود :

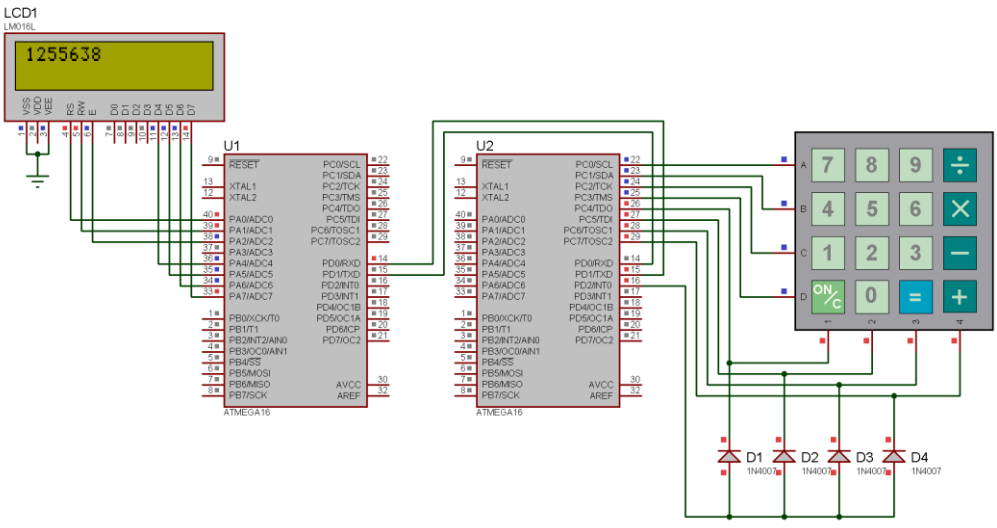
```
#include <mega8.h>
#asm
  .equ __lcd_port=0x12 ;PORTD
#endasm
#include <lcd.h>
#include <delay.h>
#include <math.h>
#include <stdio.h>
```

```
#define ADC_VREF_TYPE 0x60


unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input|ADC_VREF_TYPE;
ADCSRA|=0x40;
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}
char lcd_buffer[30];
unsigned char fes;
int pwm;
void main(void)
{
PORTB=0x00;
DDRB=0xFF;
PORTD=0x00;
DDRD=0xFF;
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 1000,000 kHz
// Mode: Fast PWM top=00FFh
// OC1A output: Non-Inv.
// OC1B output: Inverted
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0xB1;
```

```
TCCR1B=0x0A;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x06;
TCNT2=0x00;
OCR2=128;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
MCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;
```

```
// ADC initialization
// ADC Clock frequency: 125,000 kHz
// ADC Voltage Reference: AVCC pin
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x86;
lcd_init(16);
while (1)
    {
    fes=read_adc(0);
    OCR1A=fes;
    pwm=fes/2.55;
    sprintf(lcd_buffer,"PWM:%u%c",pwm,37);
    lcd_clear();
    lcd_puts(lcd_buffer);
    delay_ms(200);
    };
}
```

مثال 16 : USART



كود الاستقبال :

```
#include <mega16.h>
#asm
  .equ __lcd_port=0x1B ;PORTA
#endasm
#include <lcd.h>
#include <stdio.h>
void main(void)
{
char data;
// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: Off
// USART Mode: Asynchronous
// USART Baud rate: 19200
UCSRA=0x00;
UCSRB=0x10;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x19;
SFIOR=0x00;
lcd_init(16);
while (1)
    {
    data = getchar();
    lcd_putchar(data);
    };
}
```

كود الارسال :

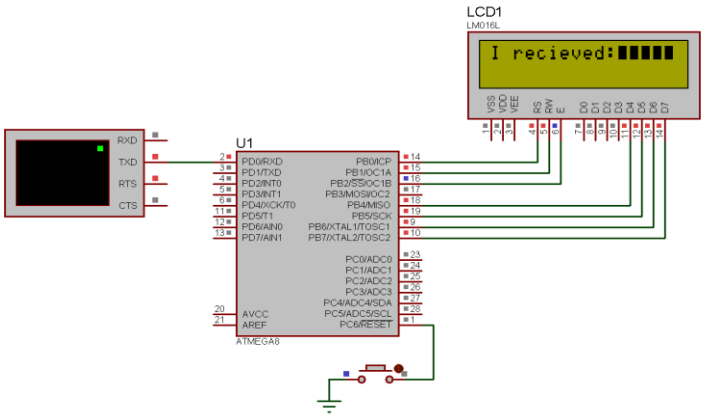
```
#include <mega16.h>
```

```
#include <delay.h>
#include <stdio.h>
bit check;
char key_code[4][4] = {{'7' , '4' , '1' , 'C'} , { '8' , '5', '2' , '0'} , { '9' , '6' , '3' , '='} , {'/' , '*' , '-' , '+'}};
char lcd_show[16];
interrupt [EXT_INT0] void ext_int0_isr(void)
{
char x , y;
if ((PINC & 0xF0) != 0xF0){

   if(PINC.4 == 0)
      x = 0;
   else if(PINC.5 == 0)
      x = 1;
   else if(PINC.6 == 0)
      x = 2;
   else if(PINC.7 == 0)
      x = 3;
   check = 1;
   if (check){
      PORTC.0 = 1;
   if((PINC & 0xF0) == 0xF0){
      y = 0;
   check = 0;
   }
}
if (check){
   PORTC.1 = 1;
   if((PINC & 0xF0) == 0xF0){
      y = 1;
      check = 0;
   }
```

```
}
if (check){
   PORTC.2 = 1;
   if((PINC & 0xF0) == 0xF0){
   y = 2;
   check = 0;
   }
}
if (check){
   PORTC.3 = 1;
   if((PINC & 0xF0) == 0xF0){
   y = 3;
   check = 0;
   }
}
putchar(key_code[x][y]);
delay_ms(100);
PORTC = PORTC & 0xF0;
}
}
void main(void)
{
PORTC=0xF0;
DDRC=0x0F;
// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Falling Edge
// INT1: Off
// INT2: Off
GICR|=0x40;
MCUCR=0x02;
MCUCSR=0x00;
```

```
GIFR=0x40;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;
// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: Off
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 19200
UCSRA=0x00;
UCSRB=0x08;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x19;
ACSR=0x80;
SFIOR=0x00;
#asm("sei")
while (1)
    {
    };
}
```

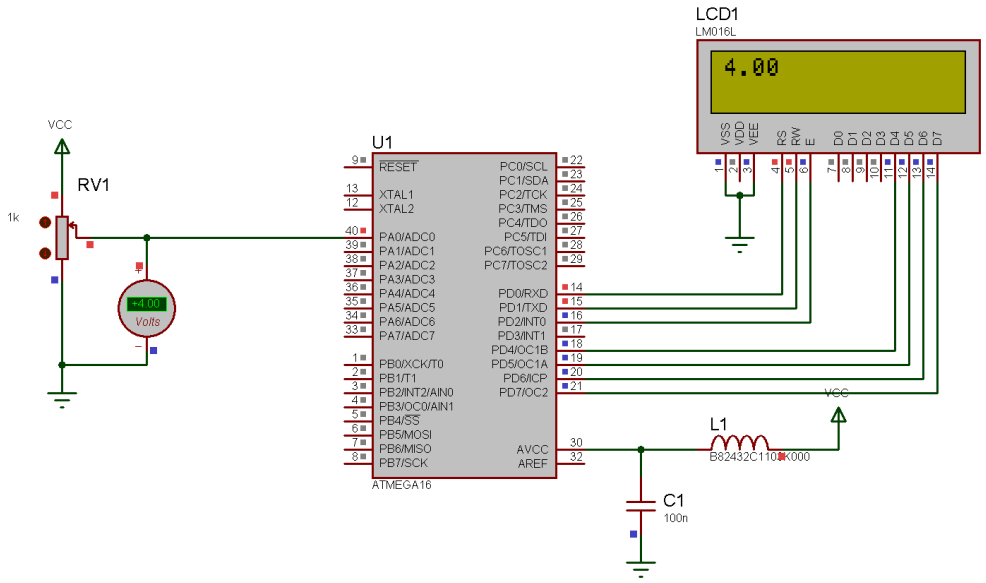
مثال 17 : استقبال ارقام



الكود :

```
#include <mega8.h>
#asm
  .equ __lcd_port=0x18 ;PORTB
#endasm
#include <lcd.h>
#include <stdio.h>
void main(void)
{
  char x;
// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: Off
// USART Mode: Asynchronous
// USART Baud rate: 300
UCSRA=0x00;
UCSRB=0x10;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0xCF;
ACSR=0x80;
SFIOR=0x00;
lcd_init(16);
lcd_putsf("I recieved:");
lcd_gotoxy(0,13);
while (1)
    {
    x=getchar();
    lcd_putchar(x);
    };}
```

مثال 18 : ADC

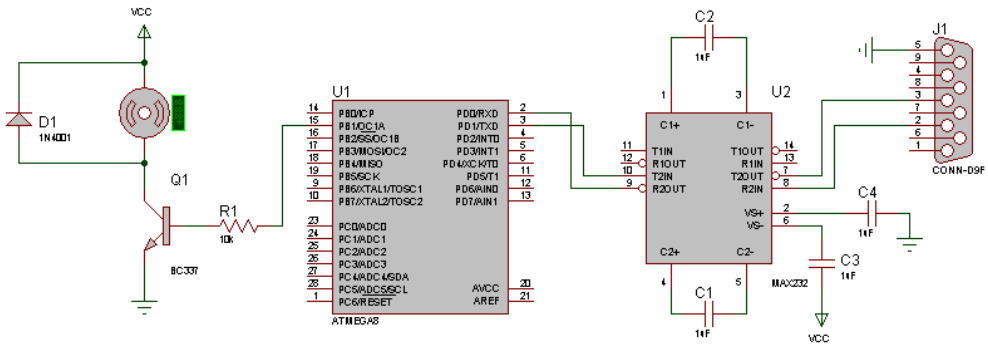


الكود :

```
#include <mega16.h>
#include <stdio.h>
#include <delay.h>
#asm
  .equ __lcd_port=0x12 ;PORTD
#endasm
#include <lcd.h>
#define ADC_VREF_TYPE 0x40
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input|ADC_VREF_TYPE;
ADCSRA|=0x40;
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}
void main(void)
{
```

```
char lcd_show[16];
float volt;
ACSR=0x80;
SFIOR=0x00;
// ADC initialization
// ADC Clock frequency: 125.000 kHz
// ADC Voltage Reference: AVCC pin
// ADC Auto Trigger Source: None
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x86;
lcd_init(16);
while (1)
    {
    volt = read_adc(0);
    volt = (volt / 1024) * 5;
    sprintf(lcd_show,"%.2f" , volt);
    lcd_clear();
    lcd_puts(lcd_show);
    delay_ms(100);
    };
}
```

مثال 19 : توصيل برنامج الماتلاب مع المايكرو بواسطة منفذ ال COM ويمكن تطويره بسهولة الى منفذ ال USB لكن في الكتاب الثاني مايكرو بلغة ال C بإذن الله



كود المايكرو :

```
#include <mega8.h>
#include <stdio.h>
void main(void)
{
char x;
PORTB=0x00;
DDRB=0x02;
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 500.000 kHz
// Mode: Fast PWM top=00FFh
// OC1A output: Non-Inv.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x81;
TCCR1B=0x0A;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x01;
OCR1BH=0x00;
OCR1BL=0x00;
// USART initialization
```

```
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x19;
ACSR=0x80;
SFIOR=0x00;
putsf("Hello!");
while (1)
    {
    x=getchar();
    switch(x)
    {
      case '0':OCR1A=0;
           break;
      case '1':OCR1A=75;
           break;
      case '2':OCR1A=90;
           break;
      case '3':OCR1A=110;
           break;
      case '4':OCR1A=135;
           break;
      case '5':OCR1A=150;
           break;
      case '6':OCR1A=175;
           break;
```

```
        case '7':OCR1A=200;
            break;
        case '8':OCR1A=225;
            break;
        case '9':OCR1A=250;
            break;
    }
    };
}
```

كود الماتلاب على command window :

تعريف للمنفذ

ضبط البود الرات

فتح المنفذ للارسال

ارسال قيمة 1

اغلاق المنفذ

```
s=serial('com1')
set(s,'baudrate',9600)
fopen(s)
fprintf(s,'1')
fclose(s)
```

اسماعيل الحجي

طالب سنة ثالثة هندسة الكهرباء والالكترون

قسم التحكم الآلي

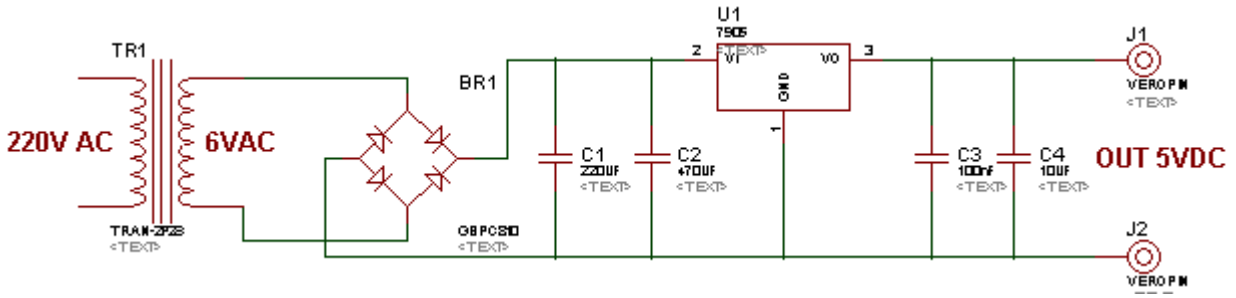
والحمد الله رب العالمين

هذه أجمل ملف وشيء عملته بيدي والفضل والمن لله تعالى الذي وفقني لجمعه و أتمنى من كل إنسان أن تكون لديه معلومات كثيرة أن يفيد حتى وإن لم يفيديه الآخرين لأنك ستموت وستدفن هذه المعلومات فلنورث هذه المعلومات من بعضنا ونطورها لنفيد أمتنا ونرضي ربنا ولا يصبح العلم تجارة أيضا وهذا مايفعله بعض الناس هذا الملف للتعلم فقط

لغة ال C لغة رائعة لبرمجة المايكرو وهي ليست لغة صعبة كما يظن البعض إنما تحتاج للفهم ليست كود بصم تطبقه على كافة الدارات

ملحق :

دارة تغذية :



الشكل (2.1) شكل دارة تغذية المتحكم

- إن دارة التنظيم الكاملة تتم بدون المكثفات C3,C4 ولكن بسبب الضجيج الناتج في الخرج توضع هاتين المكثفتين بقيم لاعلى التعيين ، وتم هنا اختيار القيم : C4 = 10 UF و C3 = 100UF .
- الضجيج ناتج عن كون الأسلاك تشكل هوائيات تستقبل إشارات الضجيج المنتشرة.
- مكثفات الدخل تمثل مرشح تمرير منخفض حيث يقوم بقصر الترددات العالية ولا نقوم عادة بمكافئة المكثفين C1 ,C2 بمكثف واحد وذلك لكي يقوم كل منهما بالترشيح عند تردد مختلف. C1 = 220 UF و C2 = 470 UF .
- بالنسبة لمنظم الجهد والذي يكون بنوعين 78XX منظم جهد موجب و 79XX منظم جهد سالب وفي كلا الحالتين لمعرفة الدخل والخرج و الأرضي : ضع المنظم بوضعية بحيث تكون الكتابة باتجاهك فيكون دائماً الخرج على اليمين وأصغر جهد على اليسار ، فإذا كان منظم موجب فالدخل هو على اليسار أم إذا كان منظم جهد سالب فالدخل هو في الوسط لأن الأرضي أكبر منه جهداً وهو يكون على اليسار.
- مع العلم أن المنظم لا يقوم بالتنظيم إذا كان جهد الدخل أقل من جهد التنظيم ،وإذا كان جهد الدخل أكبر بكثير من جهد التنظيم فإنه يجب وضع مبرد ، فلو كان جهد الدخل لمنظم 7805 هو 12 فولت فالمنظم يحتجز على دخله 7 فولت وباعتبار أن تياره 0.5 أمبير فإن الاستطاعة المبددة هي :

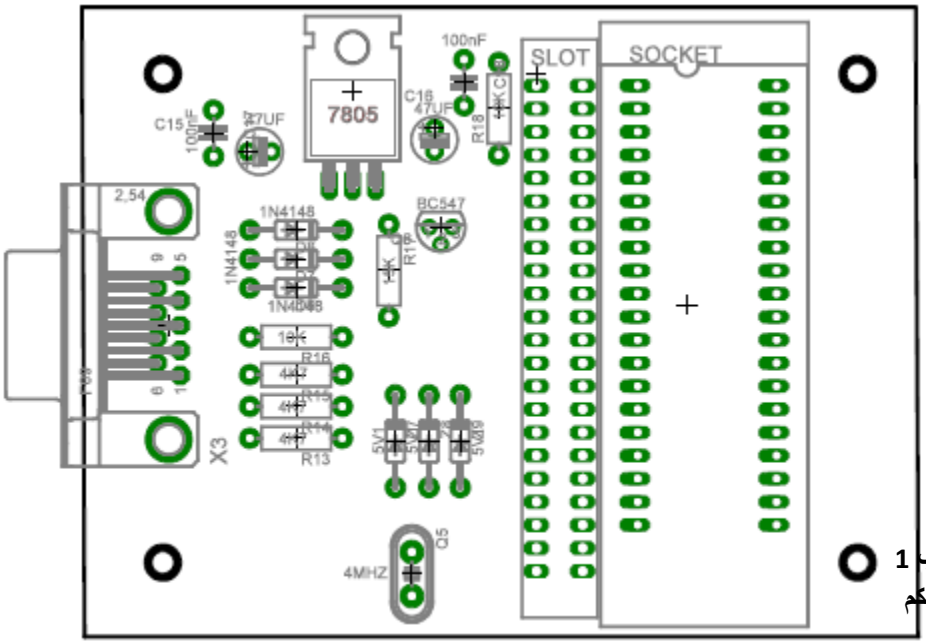
$$P = I*V \quad \rightarrow \quad P = 7V * 0.5\ A = 3.5\ WATT$$

وبالتالي يجب وضع مبرد على المنظم أو معرفة الاستطاعة التي يستهلكها واتخاذ اللازم (قد يكون غير قادر على تحمل هذه الاستطاعة حتى بوجود مبرد وذلك حسب نوع المنظم)

دارة المبرمجة :

وهي عبارة عن دارة الملائمة التي توصل بين الحاسب و المتحكم لنقل البرنامج المكتوب على الحاسب إلى المتحكم، كما يمكن للمبرمجة أن تقرأ البرنامج من الميكرو إلى الحاسب

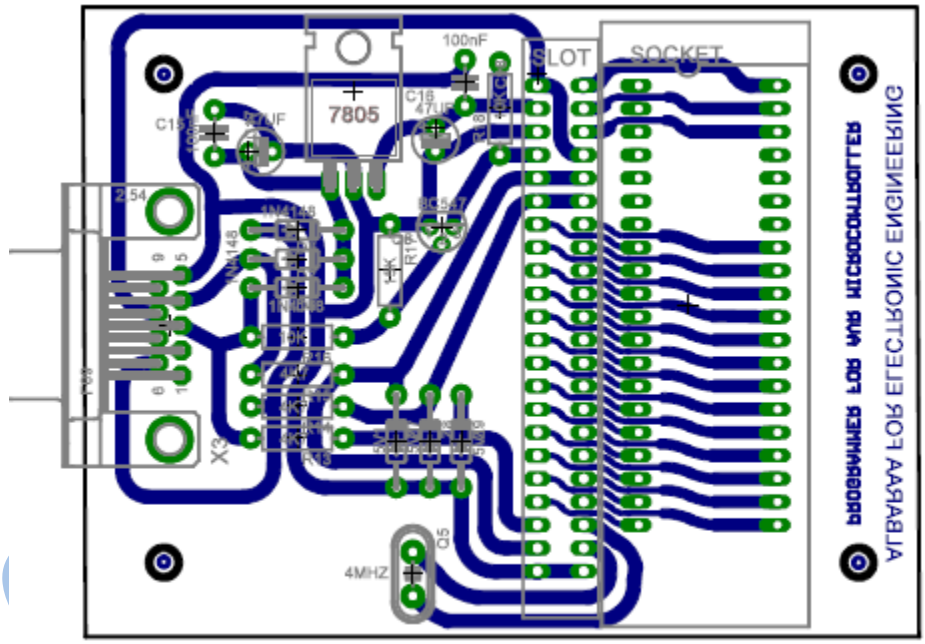
تتكون من دارة اساسية ومن دارات (كروت) الغاية منها تبديل أقطاب البرمجة للمتحكمات المختلفة لكي نستطيع عبر مبرمجة واحدة أن نبرمج كافة أنواع مبرمجات AVR حيث يعطى مخطط الدارة الساسية على الشكل:

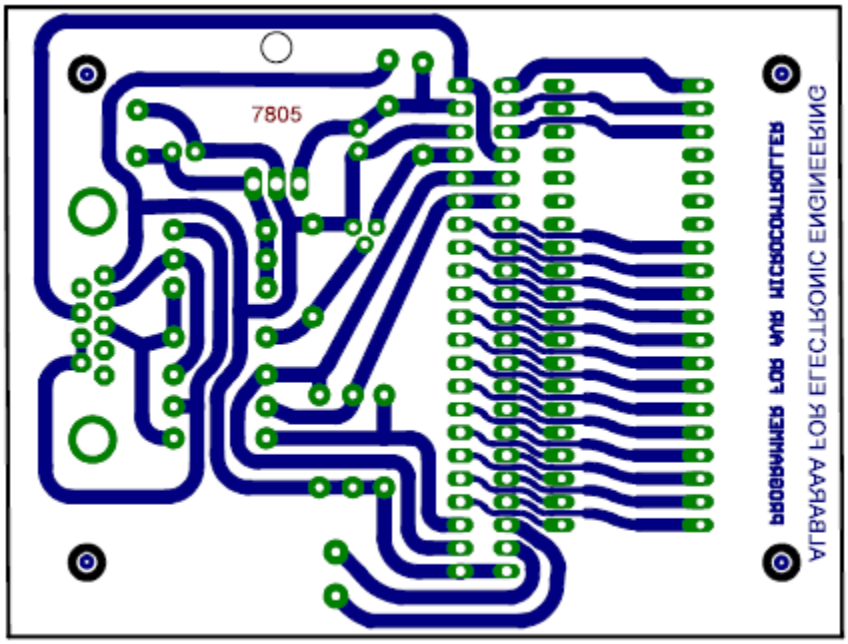


الوجه B للكرت

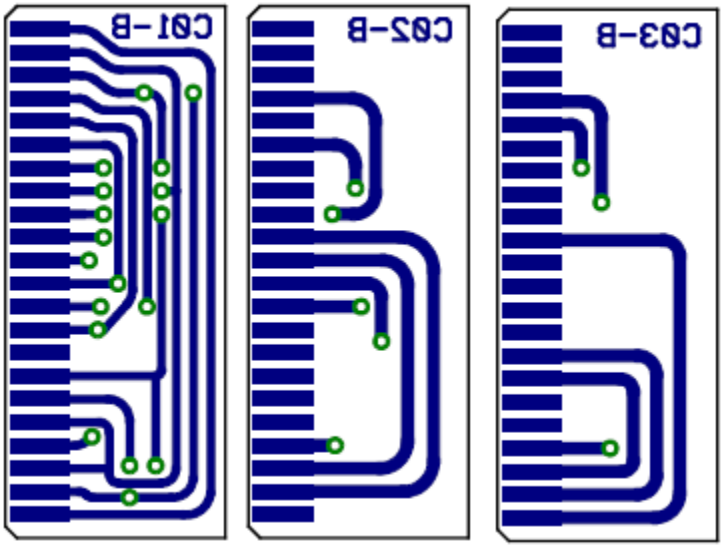
القطب 1 للمتحكم

الدارة الرئيسية مع توصيلاتها تكون بالشكل:

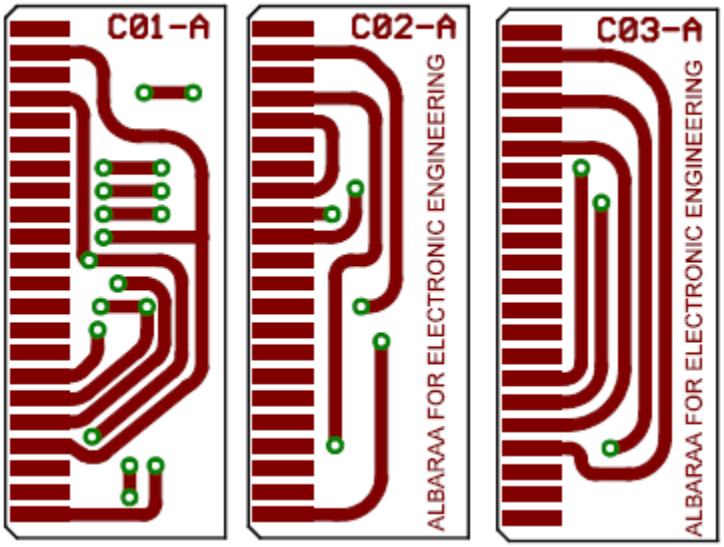




أما دارات الكروت فتعطى على الشكل: (الوجه السفلي):



أما الوجه العلوي فيعطى على الشكل:



حيث الكرت الثاني لل atmega8 والباقي لباقي المعالجات

قيادة الريليه

مبدأ عمل الريليه : عند تغذية الملف بجهد مستمر ← ينشأ حقل مغناطيسي ← تنجذب صفيحة الريليه ← تتغير وضعية التماسات .

عند قطع التغذية يقوم النابض بإرجاع صفيحة الريليه إلى وضعها الطبيعي .

يوجد لدينا نوعان من الريليهات من ناحية التغذية :

ريليه تعمل بالتيار المستمر DC: بعض القيم العملية : 1.5 , 3 , 5 , 6 , 7.5 , 9 , 12 , 18 , 24 , 36 , 110 , 220

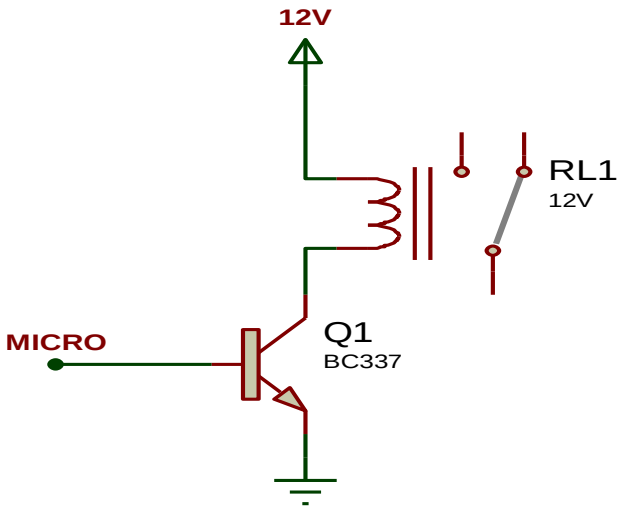
ريليه تعمل بالتيار المتناوب AC:بعض القيم العملية: 6 , 9 , 12 , 24 , 48 , 110 , 220 , 380

جميع القيم السابقة بالفولت ، وهذا البارامتر لا يكفي لاختيار نوع الريليه بل يجب أيضاً معرفة قيمة التيار الذي يتحمله تماس الريليه اي قيمة التيار الذي يستجره الحمل.

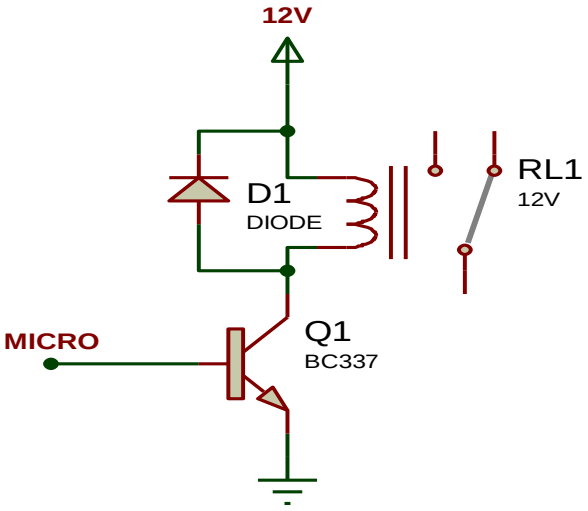
عند وصل ريليه مع ميكرو سنتعرض للمشاكل التالية:

1- إن أغلب الريليهات المستعملة هي ذات جهد 12 فولت وتيار 10 أمبير لأنها متوفرة بكثرة وتتحمل تيار عالي وبالتالي لدينا مشكلة ، وهي أن المتحكم يعطي على خرجه جهد 5 فولت لا يمكن من خلاله قيادة ريليه 12 فولت وبالتالي

الحل : نضع ترانزستور يقوم بقيادة بوبين الريليه كما يلي:



2- عند تغذية ملف الوشيعة فإنه سيختزن حقل كهربائي بحيث انه عند قطع التغذية عن الوشيعة فإن هذا الملف سيفرغ الحقل باتجاه معاكس لاتجاه التغذية وهذا التفريغ يولد نبضات جعد شوكية تصل إلى 1000 فولت تؤدي إلى عطب الترانزستور وقد تؤدي إلى عطب المتحكم ولذلك نضيف ديود يسمى ديود المسار الحر



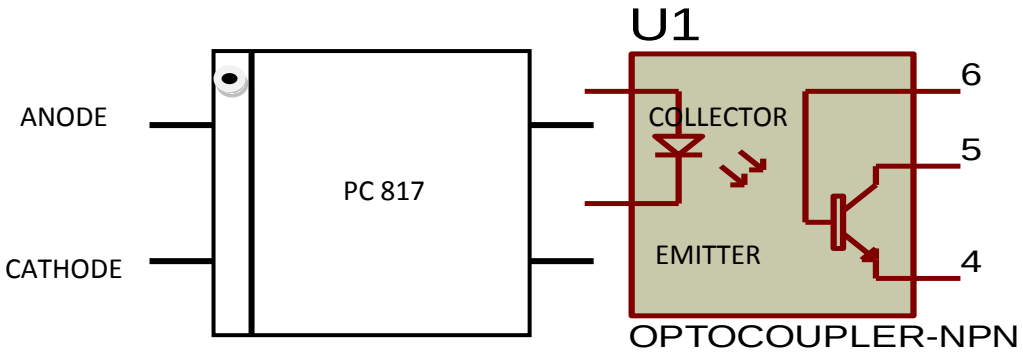
3- إن الملف عندما يمر به تيار يؤدي إلى توليد ترددات (تشويش) تنتشر بشكل قوي في النواقل المحيطة به وتنتشر في الفراغ وبالتالي لدينا مشكلة هنا هي أن الترددات التشويشية قد تنتقل من الملف عبر الترانزستور لتؤثر على المتحكم وبالتالي وقد تنتقل هذه الترددات عبر الأرضي المشترك إلى المتحكم وكما نعلم فإن المتحكم حساس جداً للضجيج وبالتالي الحل هو بعزل دارة القيادة تماماً عن دارة التحكم وبالتالي عزل هذا الضجيج عن المتحكم ويتم ذلك باستخدام الرابط الضوئي PHOTO COUPLER كعازل كهربائي.

مبدأ العازل الضوئي:

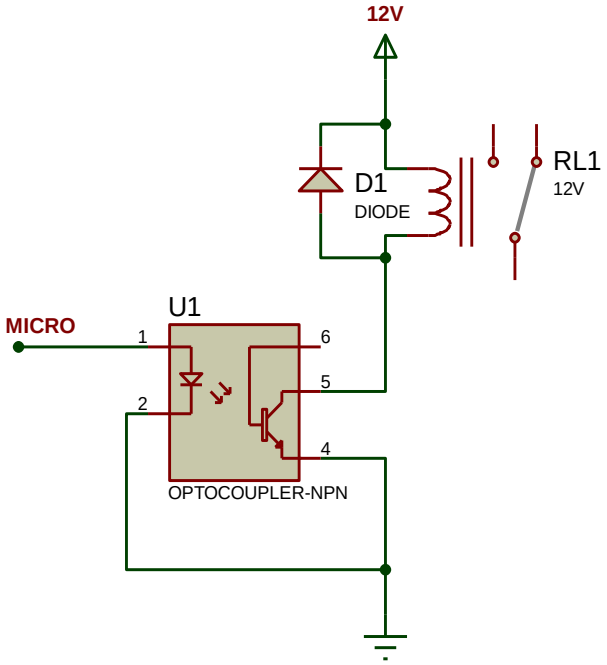
هو عبارة عن عنصرين ، ديود مرسل وترانزستور مستقبل ذو قاعدة ضوئية ، حيث أن الترانزستور الضوئي يعمل في مجالي القطع والإشباع

عند تغذية الديود الضوئي ← يعمل الترانزستور الضوئي ويصل المجمع مع الباعث.

يعطى الرابط الضوئي نظرياً وعملياً بالشكل



وبالتالي تصبح الدارة بالكامل بالشكل:



4- المشكلة الرابعة هي أن الأرضي لكل من دارة التحكم ودارة القدرة مشترك وهذه مشكلة لذلك نقوم بتصميم دارة تغذية خاصة بالمتحكم مع ديود الإرسال بقيمة 5 فولت مع أرضي منفصل وأخرى خاصة بالترانزستور الضوئي و الريليه بقيمة 12 فولت بأرضي منفصل أيضاً مع العلم أن الريليه لا تحتاج إلى منظم جهد 7812 لأنها عنصر كهربائي ليس بحاجة إلى جهد دقيق.

ملاحظات :

- العازل الضوئي موجود في السوق كدارة متكاملة بسعر رخيص 5 ل.س .
- يفضل الـ plc على الميكرو لأنه مستقر في أداءه مع العناصر التي تولد ضجيج حيث أن الـ plc يؤمن جد خاص للمداخل وأخر خاص للمخارج وأخر خاص للمعالج والدارات الرقمية حيث أن المداخل والمخارج تكون مزودة بعوازل ضوئية أو مغناطيسية، وفي درسنا اليوم قمنا بتأمين هذه الميزة للميكرو ويستطيع الميدرو بذلك القيام بأداء جيد مع الأحمال الضجيجة.
- تعتبر المحولة عازل كهربائي ولكنها ناقل مغناطيسي وبالتالي لا يمكن استخدام عوازل كهرطيسية.

- يفضل وضع مقاومة تصل الميكرو بالديود R حيث أن تحديد قيمة هذه المقاومة يتم من الدرارة السابقة كما يلي:

نقيس بالأفومتر مقاومة ملف الريليه ولتكن R2 وبالتالي تيار مجمع الترانزستور : Ic = 12V/R2وبالتالي تيار القاعدة(دخل الديود) :

$$Ib = Ic/B$$

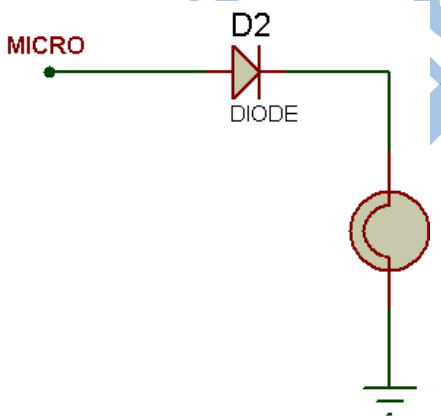
من حلقة الدخل نكتب

$$Ib = (5-0.7)/R$$

لدينا معادلتين بمجهولين R و Ib يمكن حساب R .

قيادة الديود

يفتح الديود أي يمرر عندما يكون جهد مصعده أكبر من جهد مهبطه وبالتالي يمكن الاستفادة منه بالشكل التالي:



ولكن هنا الحمل يجب أن يكون صغير وغير استطاعي ولكن لقيادة الأحمال الاستطاعية تم تطوير الديود إلى الثايرستور.

قيادة الثايرستور

الثايرستور : عبارة عن ديود موجه (قابل للتحكم) .

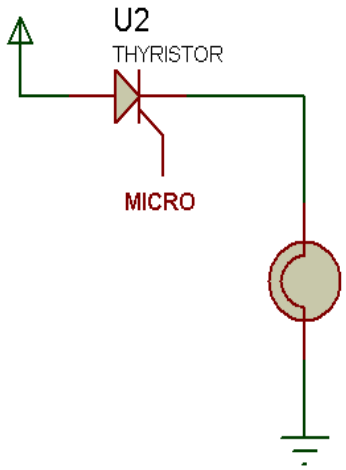
يعمل الثايرستور (يوصل)عندما يتحقق الشرطين التاليين:

1- جهد مصعده أكبر من جهد مهبطه .

2- تأمين نبضة فتح على بوابته ، مع العلم أن هذه النبضة ليس بالضرورة أن تكون مستمرة طيلة فترة العمل حيث ان نبضة واحدة تكون كافية لوضع الثايرستور في حالة العمل.

يغلق الثايرستور عند تحقق إنعدام التيار المار فيه وهذا يتحقق عندما تفتح دارة الدخل أو عندما يصبح جهد مهبطه اكبر من جهد مصعده لأنه لا يسمح للتيار بالمرور بالاتجاه العكسي.

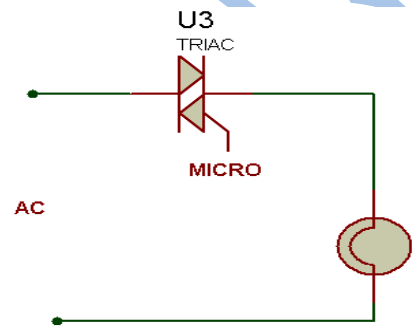
- انطلاقاً من عدم سماح الثايرستور للتيارات العكسية بالمرور أي انه لا يمرر الجهود السالبة فإنه يستخدم لقيادة الأحمال المستمرة فقط DC .



قيادة التريـاك

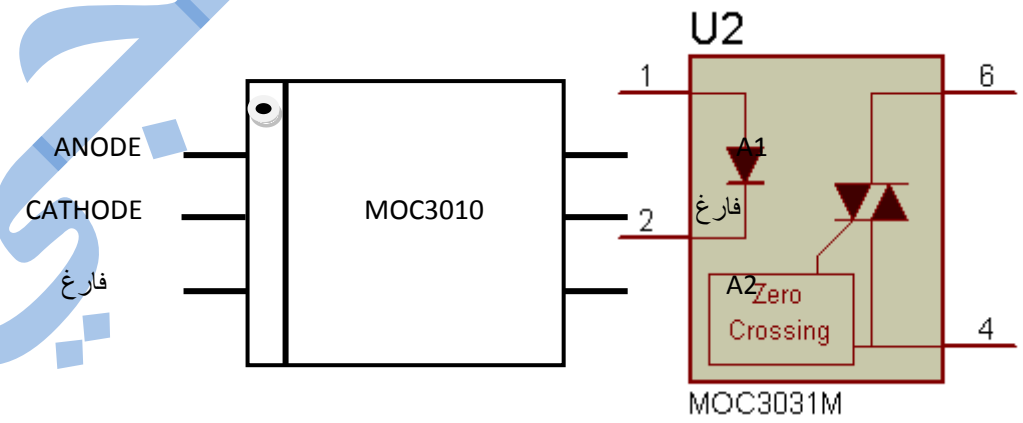
هو عبارة عن ثايرستورين على التضاد و التفرع وبالتالي فإنه يمرر كلا النبضتين الموجبة والسالبة وبالتالي يمكن استخدامه لقيادة الأحمال المتناوبة.

بعض أنواع التريـاكات الاستطاعية : BT139 – BT138 – BT 137 - BT136 والاختلاف فما بينها هو في التيار.



- كما راينا فإن العازل الضوئي PC817 يستخدم الترانزستور كمستقبل مع العلم أنه يوجد لدينا عازل ضوئي يستخدم التريـاك ، وهو عبارة عن دارة متكاملة لها الرقم MOC3061 حيث يكون على دخله ديود وعلى خرجه تريـاك ببوابة ضوئية.

يعطى شكله عملياً



سبب وضع أقطاب فارغة هي للحماية في حال تركيبه بشكل معاكس لأن الجهود التي نتعامل بها هنا هي 220 فولت وبالتالي قد يحدث احتراق في حال الخطأ .

- عند قيادة ريليه استطاعية نستخدم العازل الضوئي PC817 أما قيادة الترياك الاستطاعي تستوجب استخدام العازل الضوئي الترياكي MOC والذي يكون له عدة أنواع :

MOC3010 : جهد أعظمي 250 فولت

MOC3041 : جهد أعظمي 400 فولت

MOC3062 : جهد أعظمي 600 فولت

ملاحظة عملية :

- يعتبر الترياك افضل من الحاكمة (الريليه) من حيث أنه يتحمل تيارات عالية ولا يصدر اي ضجيج لأنه عنصر الكتروني ، كما ان تماس الريليه معرض للاهتراء بينما الترياك يكون اهتراءه أقل.