برمجة المتحكم باستخدام لغة C Compiler

٤-١- هيكلة البرنامج :

```
#include < اسم المتحكم .h>
#use delay(clock=تردد الكريستالة)
#include < اسم مكتبة ما.h >
التصريح عن متغيرات عامة ( تظهر في البرنامج الرئيسي و الفرعي )

Void  اسم البرنامج الفرعي ( متغيرات يتم تمريرها إلى البرنامج الفرعي )  // إجرائية البرنامج الفرعي
{
التصريح عن متغيرات تظهر في البرنامج الفرعي فقط
        جسم البرنامج الفرعي
}

Void main( )                                                    // البرنامج الرئيسي
{
التصريح عن متغيرات تظهر في البرنامج الرئيسي فقط
        جسم البرنامج الرئيسي
}
```

٤-٢- أهم المكتبات القياسية التي يتم استدعاؤها

- #include <math.h>

يتم استدعاء هذه المكتبة عند الحاجة إلى استخدام بعض التعليمات الرياضية مثل Sin ( ) , log ( ) , floor ()

- #include <string.h>

يتم استدعاء هذه المكتبة عند استخدام بعض التعليمات التي تتعامل مع السلاسل المحرفية.

- #include <stdlib.h>

يتم استدعاء هذه المكتبة عند استخدام بعض التعليمات مثل Atoi( ) .....

٤-٣- التصريح عن المتغيرات

الشكل العام للتصريح عن المتغيرات في لغة C هو :

نمط المتغير اسم المتغير ;

الأنماط الأساسية التي يمكن تعريف المتغيرات من خلالها يمكن إيضاحها بالجدول التالي :

نمط (نوع) المتغير	الأنماط المكافئة	السعة	المجال	
			بدون إشارة Unsigned	مع إشارة Signed
int1	short , Boolean	1 bit	عدد صحيح 0 to 1	N/A
int8	int , Byte	8 bit	عدد صحيح 0 to 255	عدد صحيح -128 to 127
int16	long	16 bit	عدد صحيح 0 to 65535	عدد صحيح -32768 to 32767
int32	long long	32 bit	عدد صحيح 0 to 4294967295	عدد صحيح -2147483648 to 2147483647
float32	float	32 bit	عدد حقيقي -1.5×10^{45} to 3.4×10^{38}	
char		8 bit	محرف يرمز برموز أسكي وله مجال 0 to 255	

الأنماط السابقة هي بشكل افتراضي بدون إشارة unsigned ما عدا float32. و لكي تصبح بإشارة فقط عندما نضع قبل النمط عبارة signed.

أمثلة

- **Int1 x;** : يتم حجز بت واحد من الذاكرة RAM اسمه x. يأخذ x إما 0 أو 1.
- **short x;** : يتم حجز بت واحد من الذاكرة RAM اسمه x. (تكافئ العبارة السابقة).
- **Int8 x;** : يتم حجز بايت واحد من الذاكرة RAM اسمه x. مجال x من 0 و حتى 255 بأعداد صحيحة.
- **Signed int8 x;** : يتم حجز بايت واحد من الذاكرة RAM اسمه x. مجال x من 128- و حتى 127 بأعداد صحيحة.
- **float x;** : يتم حجز 32 bit أي 4 بايت من الذاكرة RAM اسمه x. مجال x من الجدول السابق.

٤-٤- الحلقات : تهدف إلى تكرار تنفيذ مجموعة من الأوامر

أ- الحلقات غير الشرطية :

- for (x = قيمة ابتدائية ; x<= قيمة نهائية ; x = x + قفزة)

{ أوامر }

- for (;;) // تنفيذ لا نهائي

{ أوامر }

ب- الحلقات الشرطية :

- while (شرط منطقي)

{ أوامر }

- while (true أو 1) // تنفيذ لا نهائي

{ أوامر }

يتم كسر الحلقة (أي الخروج منها) من خلال تعليمة break.

٤-٥- استخدام تعليمة IF الشرطية : تهدف إلى تقييد تنفيذ أوامر بتحقق شرط معين

If (شرط منطقي)

{ أوامر }

If (شرط منطقي)

{ أوامر }

Else

{ أوامر }

ما هو الشرط المنطقي ؟

هو عملية منطقية تكون نتيجتها True أو False ، فعندما تكون النتيجة True (1) يتم تنفيذ أوامر while أو if.....، أما False (0) فلا يتم تنفيذها . و أهم العمليات المنطقية :

A== ٣ , A!=٣ , A<٣ , A>3 , A<=٣ A>=3

من الممكن أيضاً دمج عمليتين منطقيتين باستخدام AND : A==3 && B!=8

من الممكن أيضاً دمج عمليتين منطقيتين باستخدام OR : A==3 || B!=8

٤-٦- عمليات الإدخال و الإخراج الرقمية على البوابات :

١ – عملية الإخراج :

أ – عملية الإخراج على بوابة كاملة :

Output_اسم البوابة(value);

Output_a(value); Output_b(value); ……

الأنماط الأساسية التي يمكن تعريف المتغيرات من خلالها يمكن إيضاحها بالجدول التالي :

نمط (نوع) المتغير	الأنماط المكافئة	السعة	المجال	
			بدون إشارة Unsigned	مع إشارة Signed
int1	short , Boolean	1 bit	عدد صحيح 0 to 1	N/A
int8	int , Byte	8 bit	عدد صحيح 0 to 255	عدد صحيح -128 to 127
int16	long	16 bit	عدد صحيح 0 to 65535	عدد صحيح -32768 to 32767
int32	long long	32 bit	عدد صحيح 0 to 4294967295	عدد صحيح -2147483648 to 2147483647
float32	float	32 bit	عدد حقيقي -1.5×10^{45} to 3.4×10^{38}	
char		8 bit	محرف يرمز برموز أسكي وله مجال 0 to 255	

الأنماط السابقة هي بشكل افتراضي بدون إشارة unsigned ما عدا float32. و لكي تصبح بإشارة فقط عندما نضع قبل النمط عبارة signed.

أمثلة

- **Int1 x;** : يتم حجز بت واحد من الذاكرة RAM اسمه x. يأخذ x إما 0 أو 1.
- **short x;** : يتم حجز بت واحد من الذاكرة RAM اسمه x. (تكافئ العبارة السابقة).
- **Int8 x;** : يتم حجز بايت واحد من الذاكرة RAM اسمه x. مجال x من 0 و حتى 255 بأعداد صحيحة.
- **Signed int8 x;** : يتم حجز بايت واحد من الذاكرة RAM اسمه x. مجال x من 128- و حتى 127 بأعداد صحيحة.
- **float x;** : يتم حجز 32 bit أي 4 بايت من الذاكرة RAM اسمه x. مجال x من الجدول السابق.

٤-٤- الحلقات : تهدف إلى تكرار تنفيذ مجموعة من الأوامر

أ- الحلقات غير الشرطية :

- for (x = قيمة ابتدائية ; x<= قيمة نهائية ; x = x + قفزة)

{ أوامر }

- for (;;) // تنفيذ لا نهائي

{ أوامر }

ب- الحلقات الشرطية :

- while (شرط منطقي)

{ أوامر }

- while (true أو 1) // تنفيذ لا نهائي

{ أوامر }

يتم كسر الحلقة (أي الخروج منها) من خلال تعليمة break.

٤-٥- استخدام تعليمة IF الشرطية : تهدف إلى تقييد تنفيذ أوامر بتحقق شرط معين

If (شرط منطقي)

{ أوامر }

If (شرط منطقي)

{ أوامر }

Else

{ أوامر }

ما هو الشرط المنطقي ؟

هو عملية منطقية تكون نتيجتها True أو False ، فعندما تكون النتيجة True (1) يتم تنفيذ أوامر while أو if.....، أما False (0) فلا يتم تنفيذها . و أهم العمليات المنطقية :

A== ٣ , A!=٣ , A<٣ , A>3 , A<=٣ A>=3

من الممكن أيضاً دمج عمليتين منطقيتين باستخدام AND : A==3 && B!=8

من الممكن أيضاً دمج عمليتين منطقيتين باستخدام OR : A==3 || B!=8

٤-٦- عمليات الإدخال و الإخراج الرقمية على البوابات :

١ – عملية الإخراج :

أ – عملية الإخراج على بوابة كاملة :

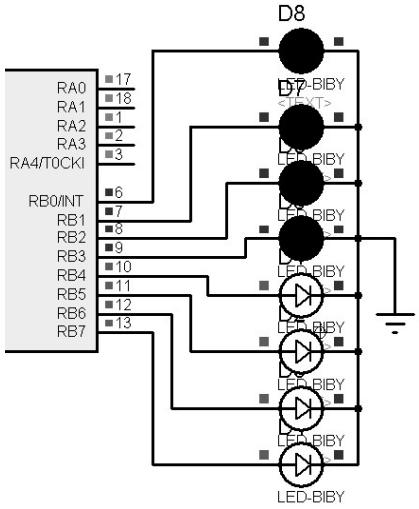
Output_اسم البوابة(value);

Output_a(value); Output_b(value); ……

حيث value قيمة ما بـ 8 bit.

مثال

Output_b(0xF0); أو Output_b(0b11110000);



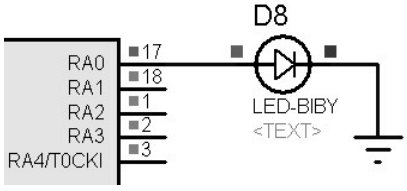
ب – عملية الإخراج على رجل أو قطب معينة :

Output_high(pin_اسم الرجل); // إخراج واحد منطقي

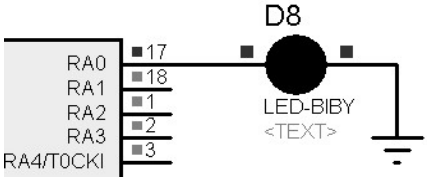
Output_low(pin_اسم الرجل); // إخراج صفر منطقي

مثال :

Output_high(pin_A0);



Output_low(pin_A0);



٢- عملية الإدخال

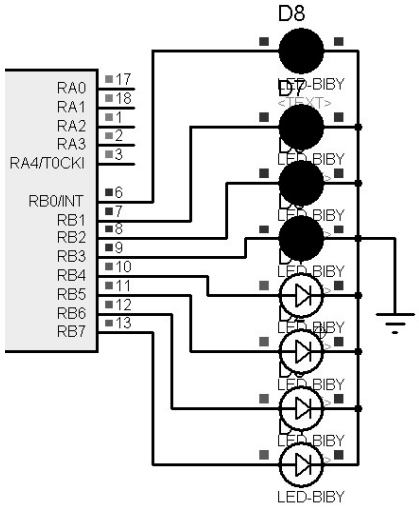
أ- عملية الإدخال لبوابة كاملة :

Int value;

حيث value قيمة ما بـ 8 bit.

مثال

Output_b(0xF0); أو Output_b(0b11110000);



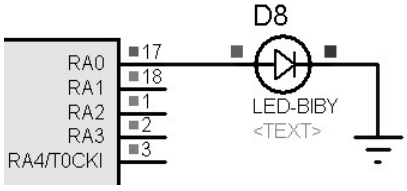
ب – عملية الإخراج على رجل أو قطب معينة :

Output_high(pin_اسم الرجل); // إخراج واحد منطقي

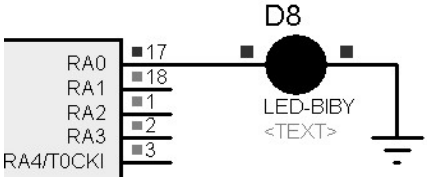
Output_low(pin_اسم الرجل); // إخراج صفر منطقي

مثال :

Output_high(pin_A0);



Output_low(pin_A0);



٢– عملية الإدخال

أ– عملية الإدخال لبوابة كاملة :

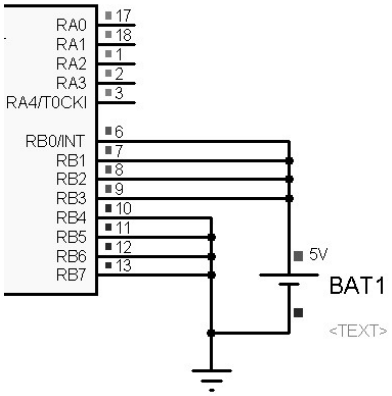
Int value;

Value = input_ اسم البوابة( );

Value هو متغير بـ 8bit أي يتم التصريح عنه من نوع int value;

مثال :

Value = input_b( );



إن قيمة Value نتيجة التوصيل السابق : value = 0x0F

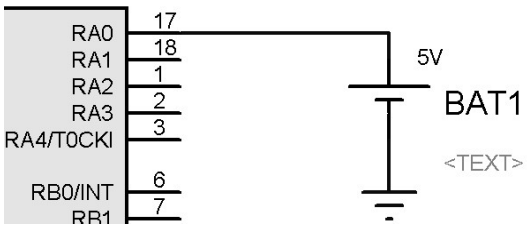
ب- عملية الإدخال لرجل معينة :

Value = input(pin_اسم الرجل);

Value هو متغير بـ 1bit أي يتم التصريح عنه من نوع int1 value; . يأخذ القيمة 1 في حال تطبيق واحد منطقي (+5v) على الرجل ، و 0 في حال تطبيق صفر منطقي (0V) على الرجل.

مثال :

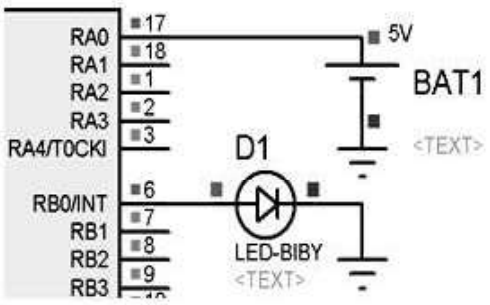
```
int1 value;
value=input(pin_A0);
```

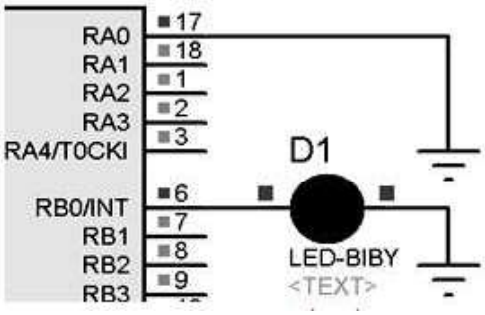


إن قيمة Value نتيجة التوصيل السابق : value = 1

مثال : اكتب برنامج يقوم بتفحص الرجل A0. إذا كان مطبق عليها 1 منطقي (+5V) يتم إخراج 1 منطقي (+5V) على الرجل B0 و إلا يتم إخراج 0 منطقي.

```
int1 value;
value=input(pin_a0);
IF ( value==1) {output_high(pin_b0);}
ELSE {output_low(pin_b0);}
```





٤-٧- تعليمة التأخير الزمني :

- Delay_ms(count);

count ثابت أو متغير من 0 و حتى 65535

- Delay_us(count);

count ثابت أو متغير من 0 و حتى 65535

- Dalay_cycles(count);

count ثابت من 1 و حتى 255.

٤-٨- البرنامج الفرعي

١- **إجرائية Procedure** : يبدأ بكلمة void و لا يعيد أية قيمة

```
Void    اسم البرنامج الفرعي ( نمط المتغير  اسم المتغير ……..)  // البرنامج الفرعي
{
التصريح عن متغيرات تظهر في البرنامج الفرعي فقط
        جسم البرنامج الفرعي
}
```

استدعاء البرنامج الفرعي // ……..; (قيمة يراد تمريرها للبرنامج الفرعي) اسم البرنامج الفرعي

٢- **تابع function** : يبدأ بنمط القيمة المستعادة ، وينتهي بـ Return

```
نمط القيمة المستعادة  اسم البرنامج الفرعي ( نوع المتغير  اسم المتغير ……..)
{
```

التصريح عن متغيرات تظهر في البرنامج الفرعي فقط

جسم البرنامج الفرعي

Return ((القيمة المستعادة (متغير مثلا)

}

استدعاء البرنامج الفرعي // ...; (قيمة يراد تمريرها للبرنامج الفرعي) اسم البرنامج الفرعي = Value

٤-٩- المصفوفات

التصريح عن المصفوفة ذات بعد واحد :

نوع عناصر المصفوفة اسم المصفوفة [عدد عناصر المصفوفة];

```
Int x[3];
char y[3];
```

التعامل مع عناصر المصفوفة :

قيمة ما = اسم المصفوفة [رقم العنصر]

```
x[0]=1;     x[1]=10;     x[2]=75;
y[0]='a';   y[1]='A';    y[2]='3';
```

التصريح عن متغيرات تظهر في البرنامج الفرعي فقط

جسم البرنامج الفرعي

Return ((القيمة المستعادة (متغير مثلا)

}

استدعاء البرنامج الفرعي // ...; (قيمة يراد تمريرها للبرنامج الفرعي) اسم البرنامج الفرعي = Value

٤-٩- المصفوفات

التصريح عن المصفوفة ذات بعد واحد :

نوع عناصر المصفوفة اسم المصفوفة [عدد عناصر المصفوفة];

```
Int x[3];
char y[3];
```

التعامل مع عناصر المصفوفة :

قيمة ما = اسم المصفوفة [رقم العنصر]

```
x[0]=1;     x[1]=10;     x[2]=75;
y[0]='a';   y[1]='A';    y[2]='3';
```

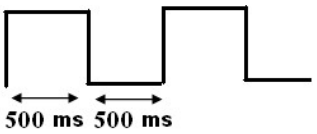
أمثلة متنوعة لبرامج بلغة C-Compiler

١- برنامج يقوم بقراءة أرجل البوابة A وإخراج النتيجة على البوابة B

```
#include <16F84A.h>
#use delay(clock=4M)
 void main()
{int value;
up:
value=input_a();
output_b(value);
goto up;}
```

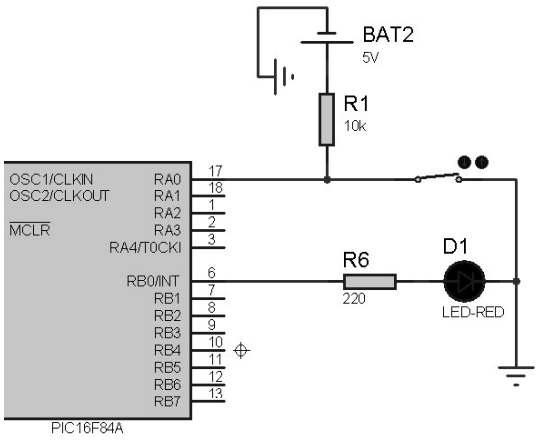
٢- برنامج يقوم بتوليد موجة مربعة ذات تردد 1Hz و نحصل على تلك الموجة على الرجل RB0

```
#include <16F84A.h>
#use delay(clock=4M)
void main()
{up:
output_high(pin_b0);
delay_ms(500);
output_low(pin_b0);
delay_ms(500);
goto up;}
```



٣- برنامج يقوم بتوليد موجة مربعة ذات تردد 1Hz عندما يتم تطبيق 0 منطقي (صفر فولت) على الرجل RA0 ، و توليد موجة مربعة ذات تردد 10Hz عندما يتم تطبيق 1 منطقي (5+ فولت) على الرجل RA0 ، و نحصل على تلك الموجة على الرجل RB0.

```
#include <16F84A.h>
#use delay(clock=4M)
void main()
{up:
if (input(pin_a0)==0)
{output_high(pin_b0);
delay_ms(500);
output_low(pin_b0);
delay_ms(500);}
if (input(pin_a0)==1)
{output_high(pin_b0);
delay_ms(50);
output_low(pin_b0);
delay_ms(50);}
goto up;}
```



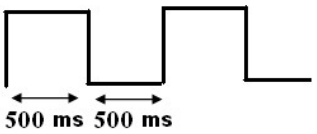
أمثلة متنوعة لبرامج بلغة C-Compiler

١- برنامج يقوم بقراءة أرجل البوابة A وإخراج النتيجة على البوابة B

```
#include <16F84A.h>
#use delay(clock=4M)
 void main()
{int value;
up:
value=input_a();
output_b(value);
goto up;}
```

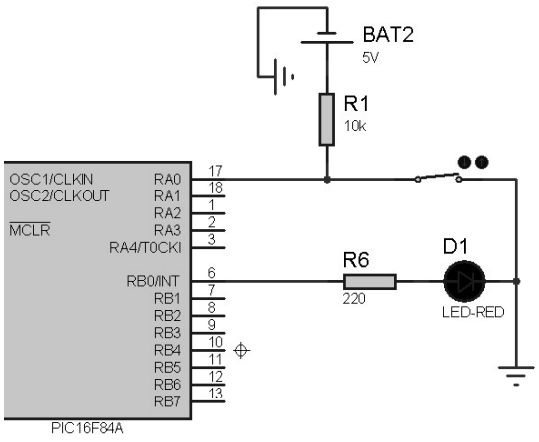
٢- برنامج يقوم بتوليد موجة مربعة ذات تردد 1Hz و نحصل على تلك الموجة على الرجل RB0

```
#include <16F84A.h>
#use delay(clock=4M)
void main()
{up:
output_high(pin_b0);
delay_ms(500);
output_low(pin_b0);
delay_ms(500);
goto up;}
```



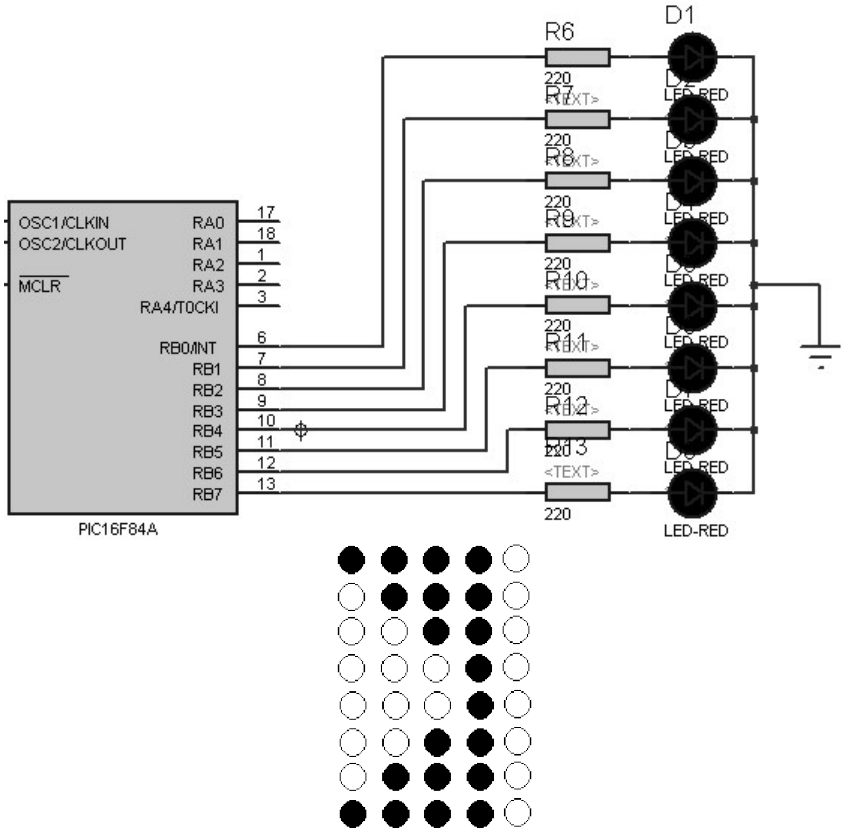
٣- برنامج يقوم بتوليد موجة مربعة ذات تردد 1Hz عندما يتم تطبيق 0 منطقي (صفر فولت) على الرجل RA0 ، و توليد موجة مربعة ذات تردد 10Hz عندما يتم تطبيق 1 منطقي (5+ فولت) على الرجل RA0 ، و نحصل على تلك الموجة على الرجل RB0.

```
#include <16F84A.h>
#use delay(clock=4M)
void main()
{up:
if (input(pin_a0)==0)
{output_high(pin_b0);
delay_ms(500);
output_low(pin_b0);
delay_ms(500);}
if (input(pin_a0)==1)
{output_high(pin_b0);
delay_ms(50);
output_low(pin_b0);
delay_ms(50);}
goto up;}
```



٤- الثنائيات الضوئية

في هذا التطبيق سنقوم بوصل ثمانية ثنائيات ضوئية على النافذة B. سنضيء الثنائيين الطرفيين بداية ثم الثنائيين اللذين بجوارهما و هكذا حتى تضيء كل الثنائيات ونعاود ذلك من جديد كما هو مبين بالشكل التالي من اليسار لليمين ، بين كل مرحلة و أخرى هناك تأخير زمني لكي تلاحظ العين ذلك.



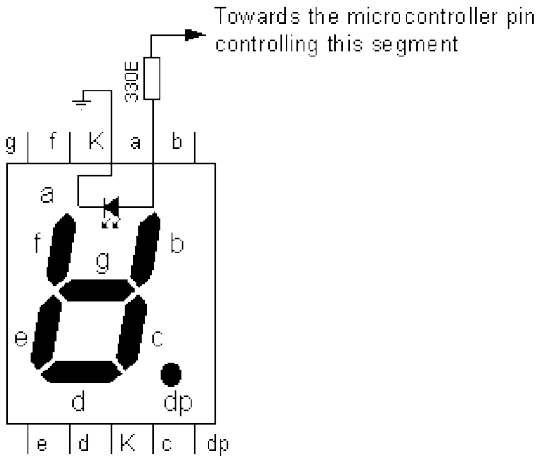

```
#include <16f84A.h>
#use delay(clock=4M)

void main()
{
while (1){
output_b(0b10000001);
delay_ms(500);
output_b(0b11000011);
delay_ms(500);
output_b(0b11100111);
delay_ms(500);
output_b(0b11111111);
delay_ms(500);
output_b(0b00000000);
                                    delay_ms(500);}}
```

عداد باستخدام السبع قطع الضوئية 7-Segment Display

في هذا التطبيق سنستخدم السبع قطع الضوئية لتعمل كعداد تعد الأرقام من 0 و حتى 9.

دارة السبع قطع ضوئية 7-Segment ما هي إلا عبارة عن ثنائيات ضوئية من الممكن أن تتصل معاً من خلال مصاعدها وتسمى عندئذ Common Anode أو قد أن تتصل معاً من خلال مهابطها وتسمى عندئذ Common Cathode

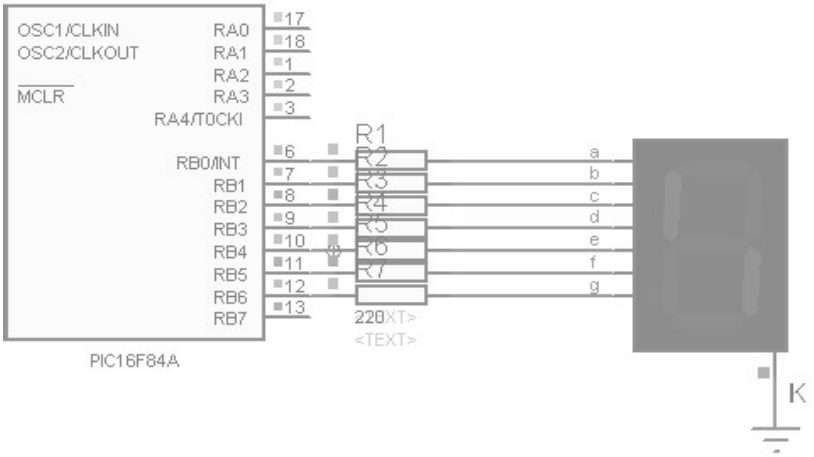
- Common Anode : لابد لكي تعمل من أن نصل الرجل المشتركة K إلى 5V+ (أي سيتم تطبيق 5V+ على مصاعد الثنائيات) ، ويتم إضاءة كل ثنائي من خلال تطبيق ٠ منطقي (0V) على الرجل الخاصة به a,b,c,d,e,f,g
- Common Cathode : لابد لكي تعمل من أن نصل الرجل المشتركة K إلى الأرضي (أي سيتم تطبيق 0V على مهابط الثنائيات) ، ويتم إضاءة كل ثنائي من خلال تطبيق 1 منطقي (5V) على الرجل الخاصة به a,b,c,d,e,f,g كما هو موضح بالشكل في الأسفل.



يوضح الجدول التالي القيم التي لابد من تطبيقها على الأرجل a,b,c,d,e,f,g لكي تظهر الأرقام من 0 وحتى 9 وذلك بأخذ النمط Common Cathode

Digit	Seg. h	Seg. g	Seg. f	Seg. e	Seg. d	Seg. c	Seg. b	Seg. a	HEX
0	0	0	1	1	1	1	1	1	$3F
1	0	0	0	0	0	1	1	0	$06
2	0	1	0	1	1	0	1	1	$5B
3	0	1	0	0	1	1	1	1	$4F
4	0	1	1	0	0	1	1	0	$66
5	0	1	1	0	1	1	0	1	$6D
6	0	1	1	1	1	1	0	1	$7D
7	0	0	0	0	0	1	1	1	$07
8	0	1	1	1	1	1	1	1	$7F
9	0	1	1	0	1	1	1	1	$6F

سنربط الأرجل a,b,c,d,e,f,g على الأرجل B0,B1,B2,B3,B4,B5,B6 على الترتيب وسنعمل على إخراج القيم من الجدول السابق على أرجل البوابة B وهذا ما يسمح لدارة السبع قطع ضوئية أن تعد من 0 و حتى 9.

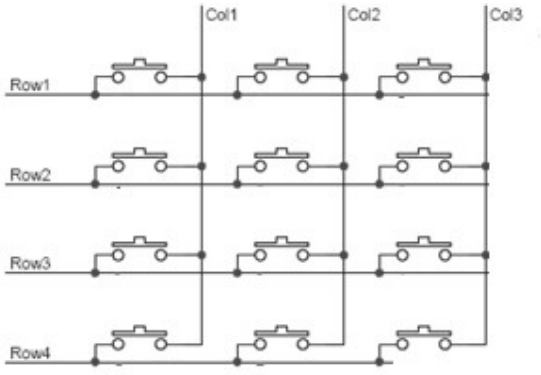



```
#include <16f84A.h>
#use delay(clock=4M)

void delay()
{delay_ms(1000);}

void main(){
up:
output_B(0b00111111);
delay();
output_B(0b00000110);
delay();
output_B(0b01011011);
delay();
output_B(0b01001111);
delay();
output_B(0b01100110);
delay();
output_B(0b01101101);
delay();
output_B(0b01111101);
delay();
output_B(0b00000111);
delay();
output_B(0b01111111);
delay();
output_B(0b01101111);
delay();
goto up;}
```

لوحة المفاتيح Keypad

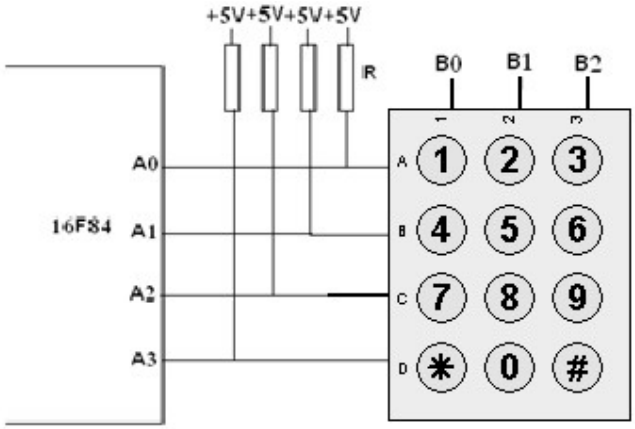
ما هي لوحة المفاتيح Keypad ؟

عبارة عن مصفوفة من الأسطر والأعمدة على شكل أسلاك غير متقاطعة. عند الضغط على أحد المفاتيح يتقاطع السطر الذي يتواجد فيه الزر مع العمود الذي يتواجد فيه الزر.تخرج أسلاك الأسطر و الأعمدة للخارج على شكل أقطاب pin ، كل قطب يعبر عن أحد الأسطر أو أحد الأعمدة.



١- معرفة المفتاح المضغوط :

لنقم بوصل الدارة المبينة جانباً:



❖ لنطبق على الأعمدة القيم التالية : B2 = +5V B1 = +5V B0 = 0V

١. عند الضغط على المفتاح (١) A0=0 ، A1=A2=A3=5V

٢. عند الضغط على المفتاح (٤) A1=0 ، A0=A2=A3=5V

٣. عند الضغط على المفتاح (٧) A2=0 ، A0=A1=A3=5V

٤. عند الضغط على المفتاح (*) A3=0 ، A0=A1=A2=5V

٥. عند عدم الضغط على أي مفتاح : A0=A1=A2=A3=5V

وبالتالي فإذا قمنا في برنامجنا بعملية فحص للأرجل A0,A1,A2,A3 سنعلم ما هو الزر الذي تم الضغط عليه.

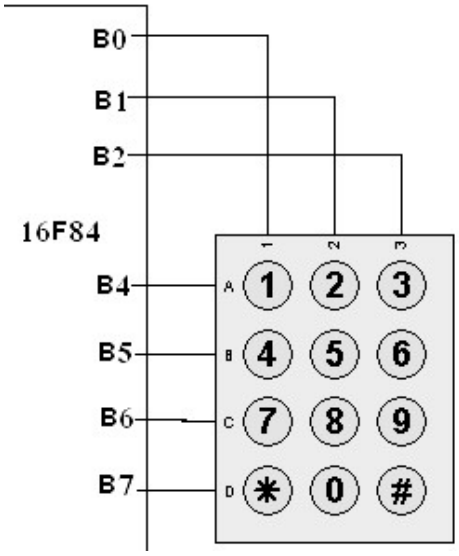
❖ لنقم الآن بتطبيق القيمة التالية على الأعمدة : B2 = +5V B1 = 0V B0 = +5V

عندئذ سنعلم الأزرار (٢)،(٥)،(٨)،(٠) بنفس الطريقة السابقة.

❖ لنقم الآن بتطبيق القيمة التالية على الأعمدة: B2 = 0V B1 = +5V B0 = +5V

عندئذ سنعلم الأزرار (٣)،(٦)،(٩)،(#) بنفس الطريقة السابقة

مقاومات السحب المشار إليها سابقاً سيتم تنفيذها بواسطة المتحكم لأن البوابة PORTB كما نعلم تمتلك هذه الخاصية وذلك بتفعيل الخانة RBPU الموجود في مسجل الاختيار OPTION-REG. وبالتالي سيتم وصل لوحة المفاتيح مباشرة مع المتحكم كما هو مبين بالشكل التالي حيث ستفعل مقاومات السحب من داخل المتحكم الصغري.



B0,B1,B2 سيتم تطبيق 011 ،ثم 101،ثم 110 .....(خرج)

B7,B6,B5,B4 سنقوم بعملية تفحص للقيم المطبقة عليها.....(دخل)

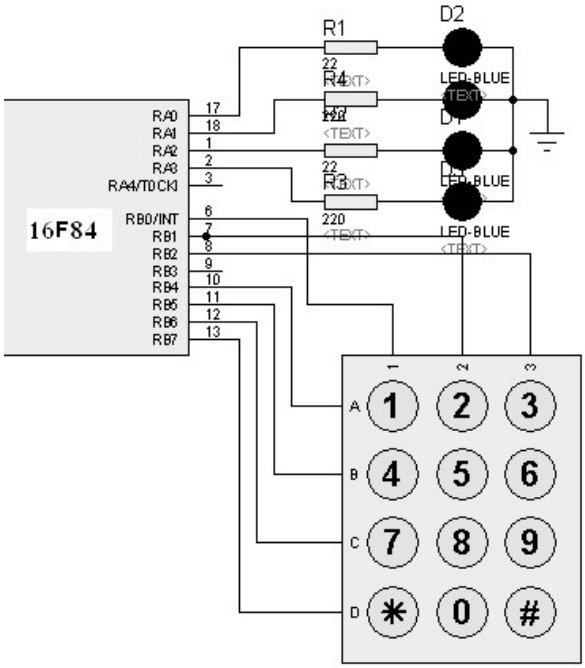
٢- البرنامج الفرعي المستخدم لمعرفة المفتاح المضغوط

char keypad_read()

يقوم هذا البرنامج بعملية المسح التي شرحت سابقاً حتى يتم الضغط على أحد الأزرار عندئذ يعيد البرنامج ذلك الزر المكبوس و على شكل محرف char (..... '3' , '2' , '1').

٣- برنامج معرفة المفتاح المضغوط من لوحة المفاتيح

سنكتب في هذه الفقرة برنامج مسح لوحة المفاتيح ، سيقوم المستخدم بالضغط على أحد المفاتيح ليقوم المتحكم الصغري بتشغيل الثنائي الضوئي المناسب : المفتاح '1' يعمل الثنائي على الرجل A0 ، المفتاح '٢' يتوقف الثنائي على الرجل A0 عن العمل ، المفتاح '٣' يعمل الثنائي على الرجل A١ ، وهكذا




```
#include <16F84A.h>
#use delay(clock=4M)

char keypad_read(){          برنامج فرعي الخاص بمسح KEYPAD
char DATA;
#use fast_io(B)
set_tris_b( 0xF0 );
port_b_pullups(TRUE);                تفعيل مقاومات السحب

WHILE (true){  هذه الحلقة تستمر إذا لم يتم الضغط على أي مفتاح ، ويتم كسرها من خلال تعليمة break
// مسح العمود الأول
OUTPUT_B(0b00001110);     وضع صفر منطقي على العمود الأول و بقية الأعمدة واحد منطقي
                          الأسطر التالية يتم من خلالها معرفة الزر المكبوس
if (input(pin_b4)==0){DATA='1';break;}
if (input(pin_b5)==0){DATA='4';break;}
if (input(pin_b6)==0){DATA='7';break;}
if (input(pin_b7)==0){DATA='*';break;}
```

```
// مسح العمود الثاني
OUTPUT_B(0b00001101);
if (input(pin_b4)==0){DATA='2';break;}
if (input(pin_b5)==0){DATA='5';break;}
if (input(pin_b6)==0){DATA='8';break;}
if (input(pin_b7)==0){DATA='0';break;}
// مسح العمود الثالث
OUTPUT_B(0b00001011);
if (input(pin_b4)==0){DATA='3';break;}
if (input(pin_b5)==0){DATA='6';break;}
if (input(pin_b6)==0){DATA='9';break;}
if (input(pin_b7)==0){DATA='#';break;}
//--------------------------------
                    }
```

هذه الحلقة تستمر ما دام المفتاح مكبوس عليه من قبل المستخدم

```
WHILE (input(pin_b4)==0 || input(pin_b5)==0 || input(pin_b6)==0 || input(pin_b7)==0)
{}
return(DATA); }

void main(){
char A;
start:
A= keypad_read ();
```

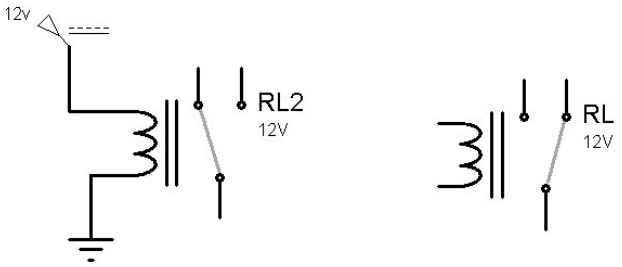
هنا يتم استدعاء برنامج KEYPAD

```
if (A=='1') output_high(pin_a0);
if (A=='2') output_low(pin_a0);
if (A=='3') output_high(pin_a1);
if (A=='4') output_low(pin_a1);
if (A=='5') output_high(pin_a2);
if (A=='6') output_low(pin_a2);
if (A=='7') output_high(pin_a3);
if (A=='8') output_low(pin_a3);
goto start;}
```

تشغيل الأجهزة التي تعمل بجهد 220V متناوب من خلال المتحكم الصغري

يعمل المتحكم الصغري بجهود منخفضة 0-5V ، ومع ذلك فإنه يستطيع التحكم بالأجهزة الكهربائية التي تعمل على جهود 220V ، وهذا يتم من خلال أداة كهربائية تعرف بالحاكمة Relay .

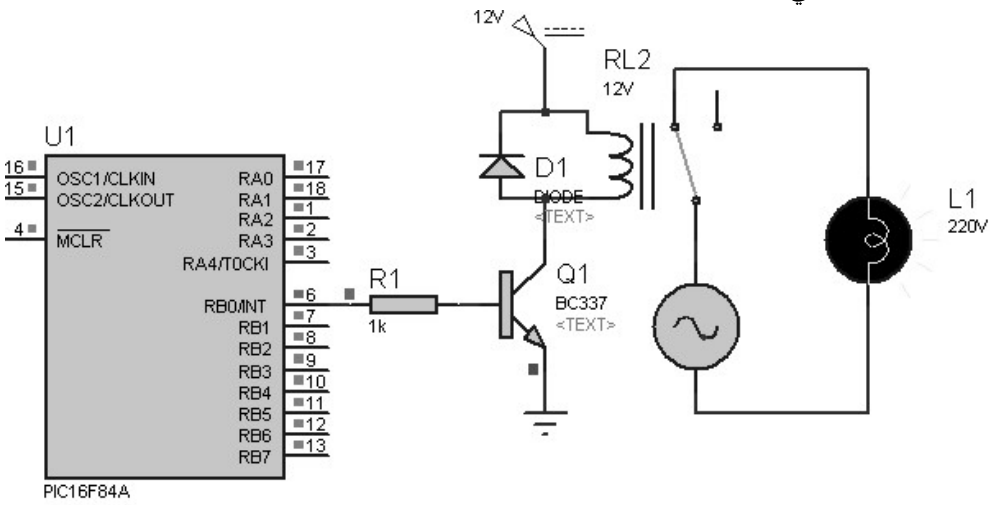
الحاكمة هي عبارة عن أداة الكتروميكانيكية تحول الإشارة الكهربائية إلى حركة ميكانيكية . تتكون بشكل رئيسي من ملف و ذراع مغناطيسي معدني يتحرك في حال مرور تيار في الملف . أبسط أنواع الحاكمة ذات خمسة أقطاب : اثنان طرفا ملف ، واحدة طرف الذراع ، واحدة موصولة مع الذراع في حال عدم مرور تيار في الملف ، واحدة يتصل معها الذراع في حال مرور تيار. يوضح الشكل التالي حاكمة في حالتين : عدم مرور تيار في الملف ، مرور تيار في الملف



يكتب على الحاكمة الجهد اللازم تطبيقه على الملف لكي يجذب الذارع ، ويكتب أيضاً ما يتحمله تماس الذارع من تيار و جهد.

لا يمكن وصل ملف الحاكمة مباشرة مع المتحكم الصغري لأنه لا يؤمن تيار كافي ، لهذا يمكن الاستعانة بترانزيستور يعمل مفتاح On/Off ويسمح بمرور تيار لا بأس به يكفي لملف الحاكمة. في الشكل التالي يعمل الترانزيستور كمصرف لتيار الملف.

عند إزالة الجهد عن ملف الحاكمة فإن الفيض المغناطيسي سينهار و سيتولد جهد عال في الاتجاه المعاكس الذي قد يؤذي الترانزيستور لهذا يتم وضع ثنائي موصول بانحياز عكسي على أطراف الملف الذي يقصر الملف في حال نشوء هذا الجهد العالي.



التحكم بالمحركات باستخدام المتحكم الصغري

كل التطبيقات التي درسناها سابقاً تعتبر تطبيقات ساكنة . إذا أردنا أن نجري عملية تحريك ديناميكية فإن المحركات الكهربائية ستفي لدينا بالغرض. المزج ما بين التطبيقات الساكنة و التحريك سيجعل من المشروع أكثر فاعلية و إثارة و مثال على ذلك الروبوتات التي أخذت شهرة واسعة النطاق. سنتعلم في هذه الفقرة كيفية ربط و قيادة المحركات من خلال المتحكمات الصغرية.

يوجد أنواع عديدة للمحركات سنهتم هنا بالأكثر شهرة استخدامه مع المتحكمات الصغرية :

١- محركات التيار المستمر DC motor

٢- محركات خطوية Stepping motor

١- محركات التيار المستمر DC motor

تعتبر أبسط أنواع المحركات الكهربائية ، مثل المحركات الموجودة في المسجلة و ألعاب السيارات. تحتاج إلى تغذية بجهد مستمر حيث تعمل على تحويل الطاقة الكهربائية إلى طاقة ميكانيكية التي يمكن الاستفادة منها في العديد من التطبيقات. لهذه المحركات سلكان فقط . عندما يتم توصيل فرق جهد مستمر إلى هذين السلكين فإن المحرك سيدور باتجاه معين ، عند عكس قطبية التغذية فإن المحرك سيتحرك بالاتجاه الآخر. جهد التغذية يتراوح ما بين 6V إلى 12V.

تستجر هذه المحركات تياراً لا بأس به لا يستطيع المتحكم الصغري تأمينه (25 mA) لهذا لا بد من استخدام دارة عزل ما بين المتحكم و المحرك تؤمن له هذا التيار. هناك طرق عديدة يمكن بها تنفيذ دارة العزل : ترانزستورات – ريليهات – دارات متكاملة مثل l293 . تعتبر الريليهات من أبسط دارات العزل و أكثرها تأميناً للتيار و لكن تبقى مشكلة الحجم و الشكل يجعلها غير مرغوبة. و مادمنا في عالم الدارات المتكاملة فإنني أفضل العمل مع دارة l293 التي سنشرحها في الفقرة التالية.

- دارة l293

تحتوي l293 على أربع بوابات قيادة دفع – جذب push-pull تسمح بمرور تيار مقداره ما بين 600mA للنوع l293D إلى 1A للأنواع المتبقية. يمتاز النوع l293D بأنه يحتوي على ديودات حماية داخلية و بالتالي ليس هناك حاجة لوضعها أثناء وصلها مع المحركات ، في حين يتطلب وضع الديودات في الأنواع الأخرى كما سنرى لاحقاً.

الشكلان التاليان يوضحان أرجل الدارة و مداخل و مخارج البوابات الأربعة.