الفصل الخامس 5

إدخال وإخراج الإشارات الرقمية

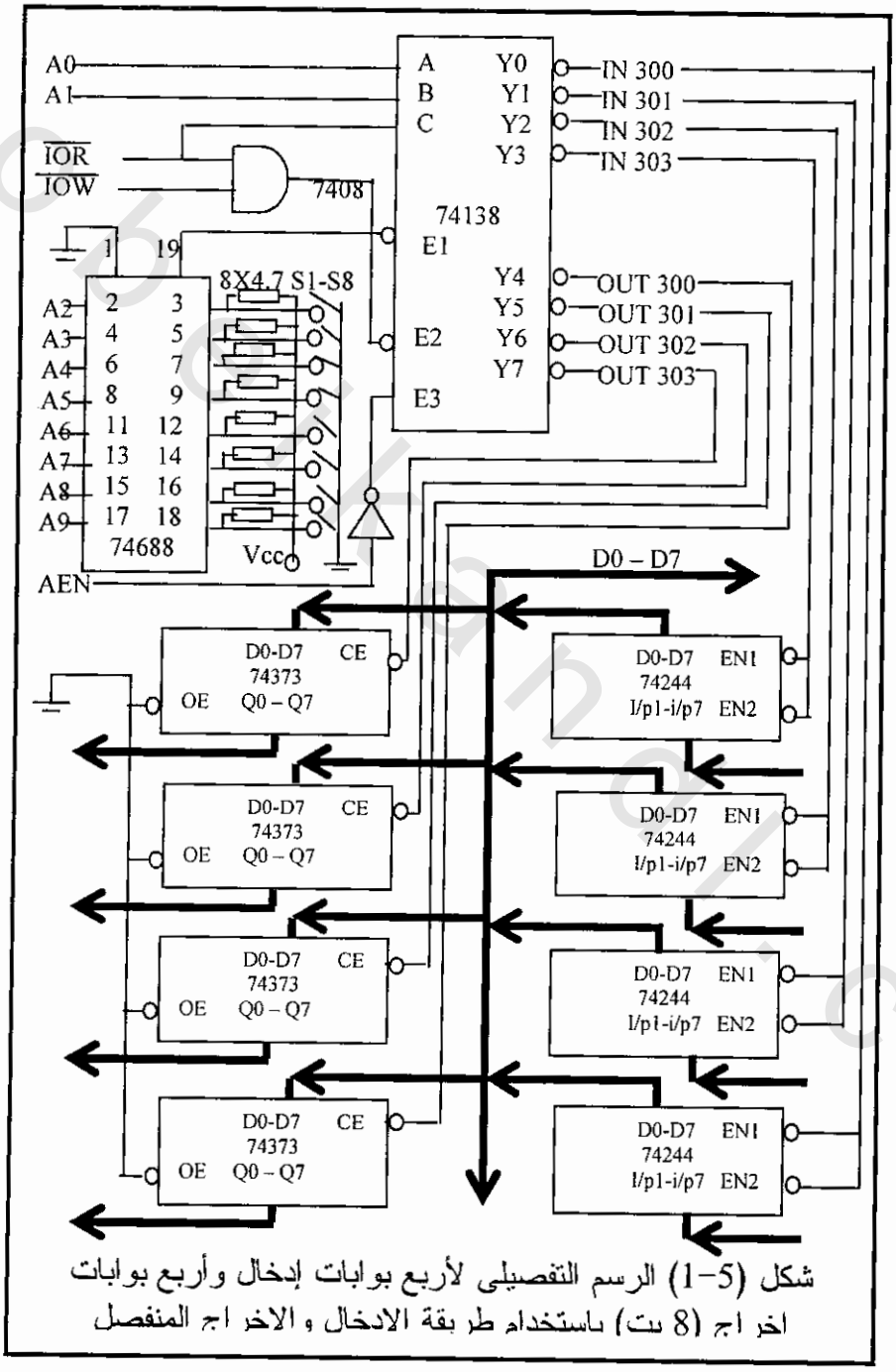
5-1 مقدمة

لقد رأينا فى الفصل الثانى الطرق المختلفة للإدخال والإخراج وهى إما باستخدام خرائط الذاكرة أو باستخدام خرائط الإدخال والإخراج ، ولقد رأينا رسما صندوقيا لمثال لبوابة من كل نوع .
سنرى فى هذا الفصل شرحا مفصلا لبناء كارت إدخال وإخراج يحتوى 4 بوابات إدخال و 4 بوابات إخراج كل منها 8 بت . أى أننا سنبنى كارت إدخال وإخراج يحتوى 32 خط إدخال و 32 خط إخراج . سنرى أيضا كيفية عزل هذه الخطوط بحيث يمكنها التعامل مع أحمال كبيرة نسبيا .
الكارت الذى سنقدمه هنا كما ذكرنا يحتوى 4 بوابات إدخال و 4 إخراج ، ولكن يمكن تعديله ليكون 8 إخراج أو 8 إدخال على حسب الرغبة وعلى حسب التطبيق المستخدم .

5-2 كارت الإدخال والإخراج الرقمى

شكل (5-1) يبين رسما تخطيطيا لهذا الكارت . الشريحة 74688 عبارة عن مقارن رقمى ذى 8 بت بحيث عندما يتطابق الرقم الموضوع بالمفاتيح S1 – S8 مع الرقم الموجود على خطوط العناوين A2 – A9 فإن خرج هذا المقارن (الطرف 19) يصبح فعالا (0) . العنوان الموضوع بواسطة المفاتيح S1 – S8 يمثل عنوان القاعدة base address لهذا الكارت . فى شكل (5-1) يتم وضع المفاتيح S1-S8=11000000 بحيث يكون عنوان القاعدة للكارت هو 300H كما فى الشكل . عندما ينشط المقارن 74688 (الطرف 19 يساوى صفر) فإن خط التنشيط $\overline{E1}$ فى منتقى العناوين 74138 يصبح فعالا هو الآخر ، لاحظ أن الطرف $\overline{E1}$ له فعالية منخفضة active low . الطرف $\overline{E2}$ منخفض الفعالية هو الآخر ، ويتم تنشيطه من خرج بوابة AND الذى يكون صفر فى حالة فعالية أى واحدة من الإشارتين $\overline{IOR}$ أو $\overline{IOW}$ حيث كل منهما منخفض الفعالية أيضا . الطرف E3 مرتفع الفعالية active high وهذا يتم تنشيطه من الإشارة AEN . الإشارة AEN تكون صفرا فى حالة التعامل مع عناوين حقيقية قادمة من المعالج سواء فى الذاكرة أو الإدخال والإخراج وتكون واحد عند التعامل المباشر مع الذاكرة ، لذلك فقد تم عكس هذه الإشارة واستخدامها لتنشيط الطرف E3 . بعد تنشيط الخطوط $\overline{E1}$ و $\overline{E2}$ و E3 تصبح الشريحة 74138 فعالة ، ويتم تنشيط الخرج المناسب $\overline{Y0}$ إلى $\overline{Y7}$ باستخدام أطراف الدخل A و B و C . الطرفان A و B موصلان على خطى العناوين A0 و A1 حيث يمكن بهما عنونة واحد من الأربع

خروج $\overline{Y0}$ إلى $\overline{Y3}$ عندما يكون الطرف $C=\overline{IOR}=0$ وبذلك تكون هذه الخروج فعالة فقط فى حالة الإدخال .

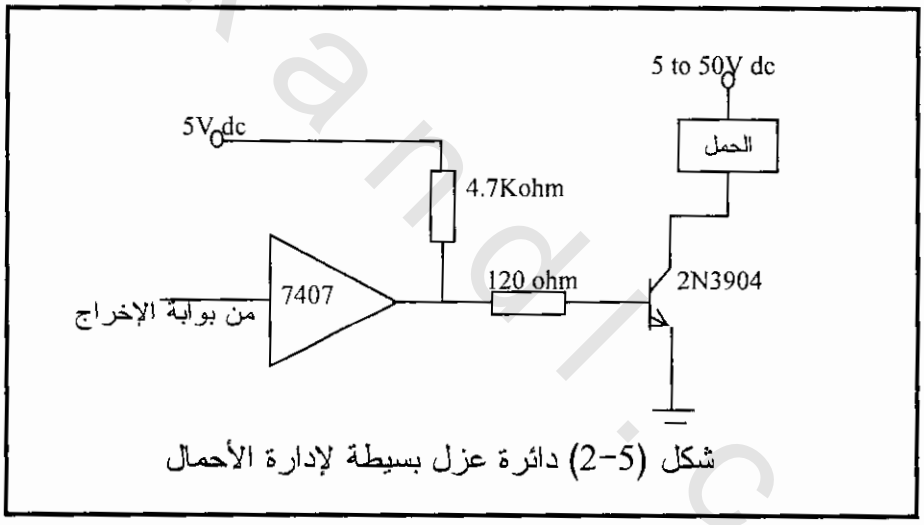


شكل (5–1) الرسم التفصيلى لأربع بوابات إدخال وأربع بوابات اخراج (8 بت) باستخدام طريقة الادخال والاخراج المنفصل

أما عندما C=$\overline{IOR}$=1 ، ففى هذه الحالة يكون $\overline{IOW}$=0 وفى هذه الحالة يتم عنونة واحد من الخروج $\overline{Y4}$ إلى $\overline{Y7}$ على حسب الإشارة الموجودة على الخطين A0 و A1 ، وسيتم تنشيطها فقط فى حالة الإخراج .

بعد ذلك تم استخدام كل من الخطوط $\overline{Y0}$ إلى $\overline{Y3}$ لتنشيط 4 بوابات إدخال من خلال الشرائح 74244 ، وكل بوابة لها الرقم الموضح فى الشكل (5-1) . أما الخطوط $\overline{Y4}$ إلى $\overline{Y7}$ فتم استخدامها لتنشيط 4 بوابات إخراج من خلال الشرائح 74373 كما فى نفس الشكل .

الإشارة الخارجة من بوابات الإخراج تكون فى أغلب التطبيقات غير قادرة على إدارة الحمل الموجود مثل إنارة لمبات البيان التى تحتاج للكثير من التيار أو إدارة موتور خطوة أو غير ذلك من التطبيقات الكثيرة . فى هذه الحالة يتم وضع عازل buffer يكون قادرا على إدارة هذه الأحمال . بالطبع ستختلف طبيعة العازل باختلاف طبيعة الحمل . شكل (5-2) يبين دائرة عازل مناسبة للكثير من الأحمال الشائعة الاستخدام .



شكل (5-2) دائرة عزل بسيطة لإدارة الأحمال

الشريحة 7407 عبارة عن عازل سداسى مفتوح المجمع . جميع الشرائح التى تكون مفتوحة المجمع لابد لتشغيلها أن يوصل هذا المجمع على مصدر القدرة Vcc من خلال مقاومة 4,7 كيلوأوم كما فى شكل (5-2) . هذا العازل موضوع لإدارة الترانزستور ويمكن الاستغناء عنه إذا كان خرج بوابة الإخراج قادرا على إدارة الترانزستور مباشرة . لاحظ أيضا أن العازل 7407 لا يعكس الإشارة . معنى ذلك إنه عندما يكون خرج بوابة الإخراج صفر ، فإن هذا الصفر ينتقل إلى قاعدة الترانزستور حيث يكون الترانزستور فى هذه الحالة فاتح أى لا يعمل .

وبذلك لن يمر تيار فى الحمل الموجود على مجمع الترانزستور . عندما يكون خرج بوابة الإخراج واحد ، فإن هذا الواحد ينتقل إلى قاعدة الترانزستور فيعمل الترانزستور ويمر تيار من مجمعه إلى الباعث ، وبذلك يمر تيار فى الحمل ويديره بالطريقة المطلوبة . لاحظ أن الجهد على مجمع الترانزستور يمكن أن يصل إلى 50 فولت ، وهذه فى الغالب تكون كافية لإدارة الكثير من الأحمال فى الكثير من التطبيقات . لاحظ أن دائرة العزل هذه ستركب على جميع الخروج من كل بوابات الإخراج .

5-3 تطبيقات على الإدخال والإخراج الرقمى

5-3-1 إخراج بيانات رقمية

أول تطبيق سنقوم بتنفيذه هو إخراج بيانات رقمية ثنائية على لمبات بيان . سنفترض أن الدائرة الموجودة فى شكل (5-1) قد تم تنفيذها حيث سنقوم باستخدام بوابة الإخراج رقم 300 لإخراج التمثيل الثنائى للأرقام من صفر حتى 255 على ثمان لمبات بيان مركبة على خرج هذه البوابة . سيتم توصيل اللمبات باستخدام عوازل كالموضحة فى شكل (5-2) . البرنامج التالى مكتوبا بلغة C++ سيقوم بهذه المهمة :

```
#include<dos.h>
#include<conio.h>
#include<stdio.h>
int main(void)
{ int i;
   do {  outportb(0x300H,i);
         sleep(1);
      } while(!kbhit() && (i!=255));
return (0);
}
```

هذا البرنامج سيخرج محتويات المتغير i على بوابة الإخراج رقم 300H طالما أن هذه القيمة أقل من 255 ، أو أن المستخدم لم يضرب أى مفتاح على لوحة المفاتيح . الفارق الزمنى بين كل رقم والتالى له هو ثانية واحدة . نلاحظ فى هذا البرنامج 3 دوال مهمة وهى :

1- دالة الإخراج outportb() ، والصورة العامة لها هى :

```
outportb(port_no, value);
```

حيث port_no هو رقم البوابة المراد الإخراج عليها ، و value هى القيمة المراد إخراجها . هذا الأمر يخرج بايت واحدة وهذا هو السر فى وجود b فى

نهاية لفظ الدالة outportb . هذه الدالة معرفة فى مكتبة dos ولذلك لابد من تضمين مكتبة dos باستخدام الأمر التوجيهى التالى :

```
#include <dos.h>
```

يمكن إخراج كلمة كاملة (2 بايت) باستخدام الدالة التالية :

```
outport(port_no,value);
```

حيث يتم إخراج البايت ذات القيمة الصغرى من value على البوابة رقم port_no والبايت ذات القيمة العظمى من value على البوابة التى رقمها port_no+1 .

2- الدالة الثانية هى الدالة sleep() ، والصورة العامة لهذه الدالة هى :

```
sleep(n);
```

حيث تتسبب هذه الدالة فى تأخير عملية التنفيذ delay عند هذا الموضع زمن مقداره n ثانية . يجب الرجوع إلى المساعد الخاص بإصدار لغة C الذى تتعامل معه لأن بعض هذه الإصدارات يعتبر أن الرقم n يجب أن يكون بالملليثانية وليس بالثانية .

3- الدالة الثالثة هى الدالة kbhit() ، وهذه الدالة ليس لها معاملات وتعود بالقيمة واحد true عند ضرب أى مفتاح من لوحة المفاتيح ، وطالما أنه لم يتم ضرب أى مفتاح فإن هذه الدالة تكون صفر .

5-3-2 إدخال وإخراج بيانات رقمية

سنفترض هنا أنه تم بناء أحد بوابات الإدخال ولتكن البوابة رقم 300H ، وتم توصيل ثمانية مفاتيح على دخلها حيث سيقوم البرنامج التالى بقراءة دخل هذه البوابة وإخراج محتوياتها على بوابة الإخراج رقم 300H أيضا التى تم تصميمها فى البرنامج السابق . هنا سنلاحظ أنه مع تغيير أى مفتاح من مفاتيح الدخل سنرى هذا التغيير منعكسا على اللمبة المقابلة له على بوابة الإخراج ، إلى أن يكون كل مفاتيح الدخل وحايد (255) حيث عندها ينتهى البرنامج . البرنامج سيكون كما يلى :

```
#include<dos.h>
#include<stdio.h>
int main(void)
{ int x;
   do {  x=inportb(0x300H);
         outportb(0x300H, x);
      } while(x!=255);
return (0);
}
```

الجديد هنا هو استخدام الدالة inportb() حيث الصورة العامة لها هى :

```
x=inportb(port_no);
```

حيث يتم قراءة محتويات بوابة الإدخال التى رقمها port_no ووضع هذه المحتويات فى المتغير x . لاحظ أنه يتم قراءة بايت واحدة فى هذه الحالة . يمكن قراءة كلمة (2 بايت) باستخدام الأمر :

```
x=inport(port_no);
```

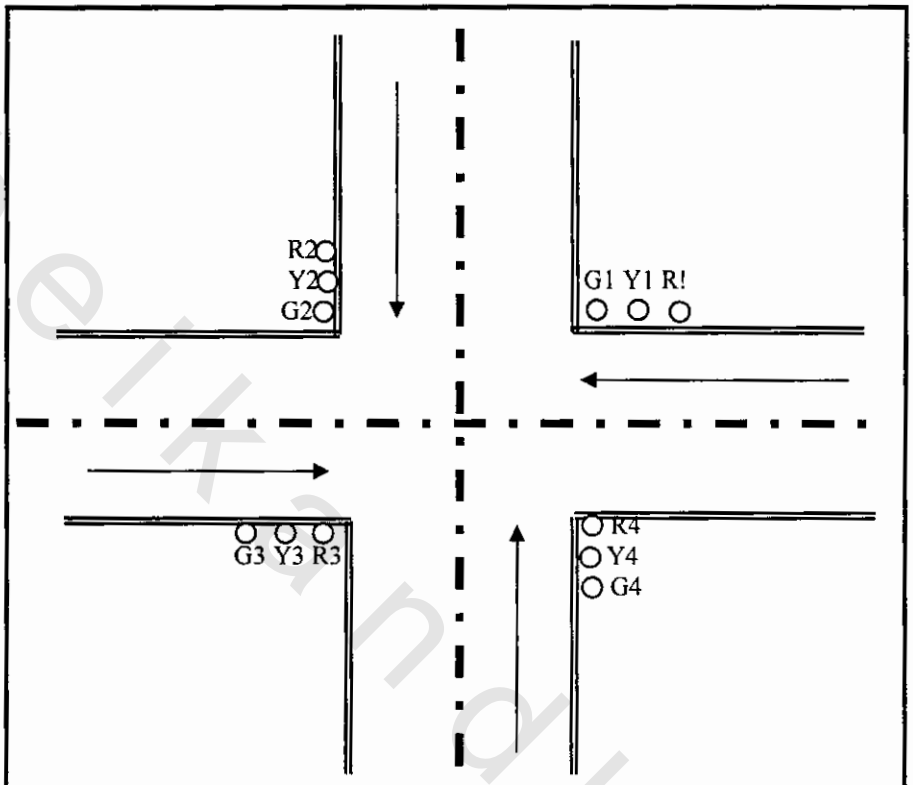
حيث يتم قراءة محتويات البوابة port_no ووضعها فى البايت الصغرى للمتغير x ، ومحتويات البوابة port_no+1 ووضعها فى البايت العظمى للمتغير x .

5-3-3 إشارات المرور فى تقاطع رباعى

البرنامج الذى سنقدمه هنا سيتحكم فى إشارة مرور فى تقاطع رباعى كالمبين فى شكل (5-3) . هذا النظام يتكون من 12 لمبة إضاءة موزعة بحيث يكون هناك 3 لمبات عند كل ركن من أركان التقاطع . اللمبة الأولى من هذه الثلاث لمبات تكون خضراء G والثانية تكون حمراء R والثالثة تكون صفراء Y . هذا النظام يسمح بمرور واحد فقط من الأربع اتجاهات ويمنع الثلاثة الباقين من المرور . أى أن واحد فقط من الاتجاهات الأربعة سيضىء اللون الأخضر G1=1 وباقى الاتجاهات ستضىء اللون الأحمر ، ويستمر ذلك لمدة 60 ثانية . بعد ذلك يضىء اللون الأصفر فى هذا الاتجاه Y1=1 لمدة 10 ثوان ويظل اللون الأحمر مضيئا فى الاتجاهات الأخرى أيضا . بعد ذلك يضىء اللون الأحمر فى جميع الاتجاهات بما فيها الاتجاه الأول R1=1 لمدة خمس ثوان كعامل أمان . بعد ذلك ينتقل هذا التتابع إلى الاتجاه الثانى ، حيث يكون G2=1 لمدة 60 ثانية ، ثم Y1=1 لمدة 10 ثوان ، ثم R1=1 لمدة خمس ثوان والأحمر فى أثناء ذلك يكون مضىء فى باقى الاتجاهات . بعدها ننتقل إلى الاتجاه الثالث ثم الرابع ، ثم نعيد الكرة من الاتجاه الأول مرة ثانية وذلك إلى ما لانهاية . فى بعض الأحيان يراد التحكم فى الإشارة يدويا نتيجة مرور أحد الشخصيات المهمة ، لكى يتم ذلك سنضع مفتاح S فى أحد الأركان يقوم رجل المرور بجعله يساوى واحد S=1 حيث فى هذه الحالة يضىء اللون الأصفر إضاءة ترددية flashing بفارق زمنى مقداره ثانية واحدة ، وفى هذه الحالة يقوم رجل المرور بالتحكم فى الإشارة يدويا وكما يشاء .

من ذلك نرى أننا سنحتاج لبوابتى إخراج 300H و 301H حيث سنستخدم الأولى بالكامل وسنستخدم من الثانية نصفها فقط (أربع خطوط) لأن كل خطوط الخرج التى نحتاجها هى 12 خط بما يقابل 12 لمبة . سنحتاج أيضا لبوابة إدخال واحدة 300H سنستغل منها الخط الأول فقط الذى سيركب عليه المفتاح S ، وباقى

خطوط هذه البوابة يمكن أن تترك للاستخدامات المستقبلية . جدول 5-1 يبين حالة جميع اللمبات والزمن المطلوب لكل حالة والقيمة المخرجة على كل بوابة فى هذه الحالة وذلك إلى أن تبدأ عملية تكرار الحالات .



شكل (5-3) شكل توضيحى لإشارة المرور فى تقاطع رباعى

سنضع أزمنة التأخير بالثانية فى مصفوفة array من 12 عنصر حيث يمثل كل عنصر فيها زمن حالة من الحالات بالترتيب ، هذه المصفوفة هى t[] . كذلك سنضع القيم المفروض إخراجها على البوابة 300H فى المصفوفة p1[] ، والقيم المفروض إخراجها على البوابة 301H فى المصفوفة p2[] ، وكل منهما مكونة من 12 عنصر كذلك كما فى الجدول 5-1 .

البرنامج الذى سيقوم بالتحكم فى إشارة المرور فى هذا التقاطع الرباعى سيكون كما يلى :

جدول 5-1 الحالات المختلفة لإشارة التقاطع الرباعي

الزمن بالثانية	البوابة 301H	300H	G4Y4R4	G3Y3R3	G2Y2R2	G1Y1R1
60	02	4C	0 0 1	0 0 1	0 0 1	1 0 0
10	02	4E	0 0 1	0 0 1	0 0 1	1 1 0
5	02	49	0 0 1	0 0 1	0 0 1	0 0 1
60	02	61	0 0 1	0 0 1	1 0 0	0 0 1
10	02	71	0 0 1	0 0 1	1 1 0	0 0 1
5	02	49	0 0 1	0 0 1	0 0 1	0 0 1
60	03	09	0 0 1	1 0 0	0 0 1	0 0 1
10	03	89	0 0 1	1 1 0	0 0 1	0 0 1
5	02	49	0 0 1	0 0 1	0 0 1	0 0 1
60	08	49	1 0 0	0 0 1	0 0 1	0 0 1
10	0C	49	1 1 0	0 0 1	0 0 1	0 0 1
5	02	49	0 0 1	0 0 1	0 0 1	0 0 1
60	02	4C	من هنا يبدأ تكرار الحالات			1 0 0

```
#include<dos.h>
#include<stdio.h>
int main(void)
{ int i,x;
  int t[12]={60,10,5,60,10,5,60,10,5,60,10,5};
  int p1[12]={0x4C,0x4E,0x49,0x61,0x71,0x49,0x09,0x89,0x49,
              0x49,0x49,0x49};
   int p2[12]={0x02,0x02,0x02,0x02,0x02,0x02,0x03,0x03,0x02,
               0x08,0x0C,0x02};
 for(;;)
 { i=1;
   x=inportb(0x300H);
   x=x&&01H;
   if(x= =1)
   { do { outportb(0x300H, 92H);
          outportb(0x301H,04H); // إضاءة الأصفر فى جميع الاتجاهات
          sleep(1); //  زمن تأخير ثانية واحدة
          outportb(0x300H,00H);
          outportb(0x301H,00H); // إطفاء الأصفر فى جميع الاتجاهات
          sleep(1);
          x=inportb(0x300H);
```

```
        }while(x= =01H);
    }
    for(i=0; i<=11;i++)
    { outportb(0x300H, p1[i]);
      outportb(0x301H, p2[i]);
      sleep(t[i]);
    }
  }
}
```

فى هذا البرنامج يمكن تغيير أزمنة التأخير وتتابع الاتجاهات أيضا عن طريق تغيير محتويات المصفوفات المعلنة فى بداية البرنامج p1[] و p2[] و t[] .

5-3-4 التحكم فى موتور الخطوة Stepper motor

من مميزات موتور الخطوة على باقى أنواع الموتورات أنه يمكن إدارته باستخدام إشارة رقمية ذات قدرة مناسبة . وعلى ذلك يمكن إدارته باستخدام معالج أو حاسب بسهولة من خلال برنامج ينفذ بأى لغة من لغات البرمجة .
هناك أنواعا عديدة من موتورات الخطوة ، ولكن فكرة عملها كلها سهلة وبسيطة ويمكن مراجعتها بالتفصيل فى الفصل الخاص بموتور الخطوة (الفصل الثامن) .
فى العادة يكون الموتور مجهزا بدائرة مقابلة تأخذ التتابع الثنائى لتوزيعه على الملفات بالطريقة المناسبة كما فى الفصل الخاص بذلك . جدول 5-2 وجدول 5-3 يبينان التابع الثنائى المطلوب فى حالة الدوران عكس أو مع عقارب الساعة .
برنامج إدارة موتور الخطوة السابق عكس عقارب الساعة على أن يظل الموتور يدور إلى أن يتم ضرب أى زرار من لوحة المفاتيح ، بلغة C من الممكن أن يكون كما يلى :

جدول 5-3 التتابع الثنائى المطلوب للدوران مع عقارب الساعة

D	C	B	A
1	0	0	1
0	1	0	1
0	1	1	0
1	0	1	0
بداية تكرار التتابع			

جدول 5-2 التتابع الثنائى المطلوب للدوران عكس عقارب الساعة

D	C	B	A
1	0	1	0
0	1	1	0
0	1	0	1
1	0	0	1
بداية تكرار التتابع			

```
#define delay 0.005
int main(void)
{  do{  outportb(0x300H,0AH);
        sleep(delay);
        outportb(0x300H,06H);
        sleep(delay);
        outportb(0x300H, 05H);
        sleep(delay);
        outportb(0x300H, 09);
     }while(!kbhit());
}
```

حاول تنفيذ هذا البرنامج مع استخدام أزمنة تأخير مختلفة قصيرة بشكل ملحوظ ، وطويلة بشكل ملحوظ أيضا لترى استجابة الموتور لسرعة تتابعات التغذية .

كل هذه التطبيقات يمكن تشغيلها من خلال أى وسيلة أخرى من وسائل التقابل مع الحاسب مثل المخرج المتوازى parallel port أو المخرج المتتـــالى serial port كما سنرى .