* The name of the clk_div register is PWMP and cycle_high is PWM1.

STEPPER MOTOR CONTROLLER :-

❖ **Explain how a stepper motor is controlled using driver. Gice relevant hardware and software details.**

Jan-07,8M

* A stepper motor is an electric motor that rotates fixed number of degrees whenever we apply a 'step' signal.

* Stepper motor can rotate 1.8° (Full step) or 0.9°(half step) per step. If the motor rotates 1.8° per step, then to move 360°, the number of steps required is 200° ie (1.8×200 steps = 360°)

* Internally, a stepper motor typically has four coils. To rotate the motor one step, we pass current through one or two of the coils. Thus rotating the motor 860° requires current to the coils in a specified sequence. Applying the sequence is reverse causes reversed rotation.

Stepper motor can be controlled in 2 ways :

1) Using a stepper motor driver

2) Controlling a stepper motor directly

Application of Stepper motor :-

1) Dist drivers
2) Printers
3) Photocopy
4) Fax machines
5) Robots
6) Camcorders
7) VCR's.

Stepper Motor Control using Driver :-

* Controlling a stepper motor requires applying a series of voltages to the four coils of the stepper motor. The coils are energised one or two at a time causing the motor to rotate one step.

* In this example, we are using a 9-volts, 2-phase bipal - ar stepper motor. The table indicating the input sequence required to rotate the motor. The entire sequence must be applied to get the motor to rotate 7.5 degrees.

To rotate the motor in the opposite direction, we simply apply the sequence in reverse order.

sequence	A	B	A'	B'
1	+	+	-	-
2	-	+	+	-
3	-	-	+	+
4	+	-	-	+
5	+	+	-	-

* We can use an 8051 μc and MC3479P chip to control the stepper motor. We need only worry about setting the direction on the clockwise / counter clock wise pin ($\overline{CW}$/CCW) and pulsing the clock pin (clk $_0\sqcap\!\!\sqcup$) on the stepper motor driver chip using the 8051 microcontroller.

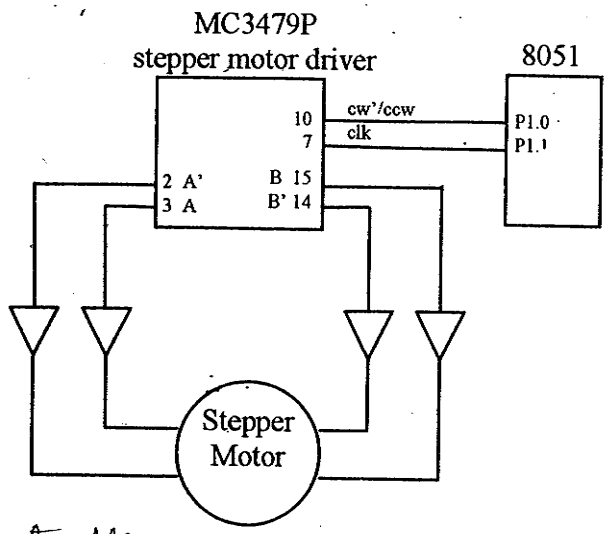

Fig 1@: Controlling a stepper motor using a driver

```
/* main.c */
sbit clk = P1^1;
sbit CW = P1^0;
void delay (void)
{
 int i, j;
 for( i = 0 ; i< 1000; i++)
  for (j= 0; j<50; j++)
   i = i + 0;
}
void main (void)
/* turn the motor forward */
CW = 0;     /* set direction*/
```

```
clk = 0;  /* pulse clock */
delay ();
clk = 1;
/* turn the motor backwards */
CW = 1;     /* set direction */
clk = 0;    /* pulse clock */
delay = ();
clk = 1;
```

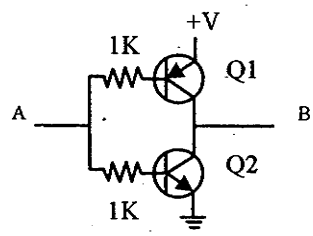

Fig ① b · Buffer

* The o/p pins on the stepper motor driver do not provide enough current to drive the stepper motor To amplify the current, a buffer is needed and is show in fig 1 ⓑ.

* Q_1 is an PNP transistor & Q_2 is an NPN transistor. 'A' is connected to the 8051 microcontroller and 'B' is connected to the stepper motor.

CONTROLLING STEPPER MOTOR DIRECTLY :-

(Without Using DRIVER)

In this example, the stepper motor driver is eliminated. The stepper motor is connected directly to the 8051 µc as shown in fig 1 ⓐ.

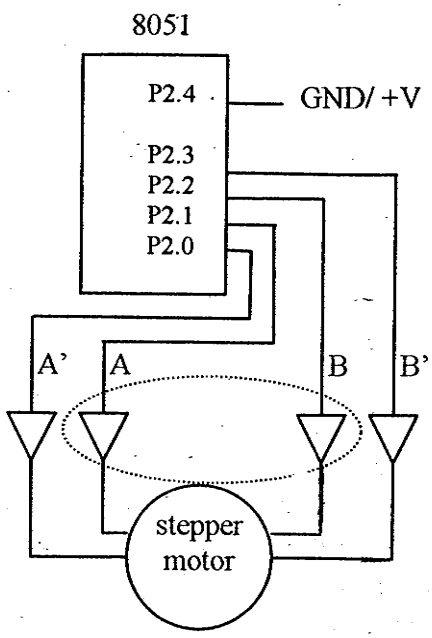

Fig ①a : Controlling a stepper motor directly

* The direction of the stepper motor is controlled manually.

If P2.4 is grounded, the motor rotates counter clockwise, otherwise the motor rotates clockwise

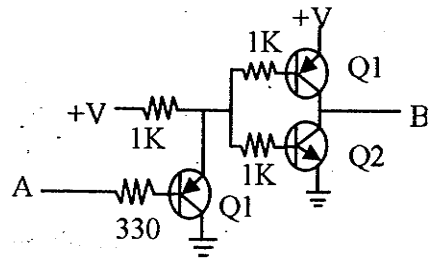

Fig ① b . Buffer.

* The 8051 ports are unable to directly supply the current needed to drive the motor . To amplify the current, a buffer is needed and is shown in fig 1⑥.

* Q_1 are PNP transistor and Q_2 is an NPN transistor A is connected to the 8051 microcontroller and B is

connected to the stepper motor.

The sample code to run the stepper motor is shown below

```
sbit notA = P2^0;
sbit isA  = P2^1;
sbit notB = P2^2;
sbit isB  = P2^3;
sbit dir  = P2^4;
void delay ( )
{
   int a, b;
   for (a=0; a<5000; a++)
   for (b=0; b<1000; b++);
}
void move (int dir, int steps)
{
   int Y, Z;
   if (dir==1)
   { for (Y=0; Y<=steps; Y++)
     { for (z=0; Z<=19; Z+=4)
       {  isA = lookup[z];
          isB = lookup[z+1];
          notA = lookup[z+2];
          notB = lookup[z+3];
          delay();
       }
     }
   }
  if (dir==0)
  {
    for (Y=0; Y<=step; Y++)
```

```
    {
      for (z = 19; z >= 0; z -= 4)
      {
          isA = lookup[z];
          isB = lookup[z-1];
          notA = lookup[z-2];
          notB = lookup[z-3];
          delay();
      }
    }
  }
}

int lookup[20] = { 1,1,0,0,0,1,1,0,0,0,
                   1,1,1,0,0,1,1,1,0,0 };

void main()
{
  while(1)
  { /* move forward 15 degrees */
    move(1,2);
   /* move backward 7.5 degrees */
    move(0,1);
  }
}
```

Note :- int lookup[20] = { 1,1,0,0, [z] [z+1] [z+2] [z+3]
0,1,1,0
0,0,1,1
1,0,0,1
1,1,0,0 };