Chapter 7: Motor Control

*Motor control which needs PWM, such as DC motor speed control and Servo Control, is discussed in a separate volume.

The most common motors in microcontroller control for robots and other motion/movement control are DC motors. They are cheap and easy to control (rotational direction, start/stop, and speed). However, for robot arm or tracking camera movement control, stepper motor is the only way to satisfy the requirement since its position is precisely controlled for a desired position. In this chapter we explore the world of DC motors and stepper motors. A similar one, servo motor, will be discussed in Chapter 13 while we discuss about the PWM (Pulse Width Modulation) module of 16F877. Therefore, any control of motors which requires PWM is not discussed here but later in Chapter 13. So the main theme of control is consistent of: (1) DC motor rotational direction control and (2) Stepper motor position control.

1. Motors

DC Motor

A DC motor usually means a permanent-magnet, direct-current (DC) motor of the sort used in toys, models, cordless tools, and robots. These motors are particularly versatile because both their speed and direction can be readily controlled; speed by the voltage or duty cycle of their power supply, and direction by its polarity.

Fig. 32 DC motor

Torque is a measurement of the motors power. The higher the torque of the motor the more weight it can move. DC motors provide different amounts of torque depending on their running speed, which is measured in RPM (revolutions per minute). At low RPM DC motors produce poor torque, and generally the higher the RPM, the better the motors torque. However, in high torque, the speed may be too high for an application. That's why we have to use gears (or geared motor) to reduce the overall speed of the motor and running at the top speed to get the most power to, say, a wheel attached to the shaft of the motor.

Stepper Motor

Stepper motors work in a similar way to dc motors, but where dc motors have 1 electromagnetic coil to produce movement, stepper motors contain many. Stepper motors are controlled by turning each coil on and off in a sequence. Every time a new coil is energized, the motor rotates a few degrees, called the step angle. Repeating the sequence causes the motor to move a few more degrees and so on, resulting in a constant rotation of the motor shaft. For example, a stepper motor with a step angle of 7.5 degrees requires 48 pulses for a complete revolution. The diagram below shows how a stepper motor works. The magnet in the middle of the arrangement is connected to the motor shaft and produces the rotation. The 4 magnets around the outside represent each coil of the stepper motor. As different coils are energized the central

magnet is pulled in different directions. By applying the correct sequence of pulses to the coils the motor can be made to rotate.

Fig. 33 Stepper Motor

This design gives stepper motors the upper hand over DC motors. Varying the speed of the input sequence can exactly control the speed of the motor. Also, by keeping count of the sequence the motor can be made to rotate any number of times to any position. The way to activate the coils determines two classes of the stepper motor: unipolar stepper motor and bipolar stepper motor. In a unipolar stepper motor the current direction in a coil is unidirectional. In other words, the polarity of the voltage across a coil is always the same or there is no voltage developed across the coil. Therefore in unipolar scheme, the motor power's positive polarity must be connected to the terminals 1 and 2 of the illustration shown below. The control of a unipolar stepper motor is, thus, to turn on and off the current flow through the four coils in a sequence for the step direction and speed. As we see a unipolar stepper motors has at least 5 external wires: 5 when terminals 1 and 2 are jointed together into one terminal, and 6 when two are left as separate terminals.

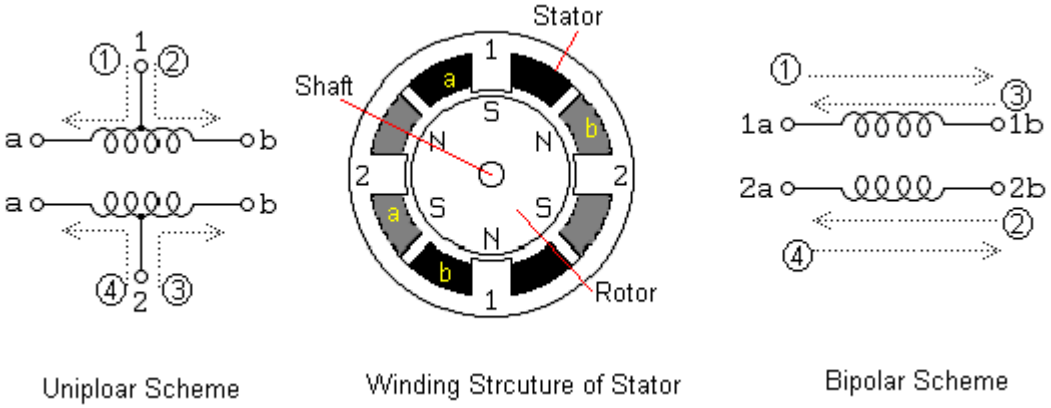


Fig. 34 How a Stepper Motor Works

In bipolar scheme, the coils can be excited by both directions. In other words, in a coil current can flow both ways. Therefore, the bipolar scheme must provide a way to apply either positive voltage to one of the coil and ground to the other end, vice versa, at a given sequence. A bipolar stepper motor therefore has only 4 external wires to excite.

Servo Motor

A Servo is a small device that has an output shaft. This shaft can be positioned to specific angular positions by sending the servo a coded signal. As long as the coded signal exists on the

input line, the servo will maintain the angular position of the shaft. As the coded signal changes, the angular position of the shaft changes. In practice, servos are used in radio controlled (RC) airplanes to position control surfaces like the elevators and rudders. They are also used in radio controlled cars, puppets, and of course, robots.

Fig. 35 Servo Motor

Servos are extremely useful in robotic application. The motors are small, and have built with a motor, control circuitry, a set of gears, and the case, and are extremely powerful for their size. A standard servo has 42 oz/inches of torque, which is pretty strong for its size. It also draws power proportional to the mechanical load. A lightly loaded servo, therefore, doesn't consume much energy. There are 3 wires in it: power, ground, and control.

The control circuitry includes a potentiometer that is connected to the output shaft, which allows the control circuitry to monitor the current angle of the servo motor. If the shaft is at the correct angle, then the motor shuts off. If the circuit finds that the angle is not correct, it will turn the motor the correct direction until the angle is correct. A normal servo is mechanically not capable of turning any farther due to a mechanical stop built on to the main output gear, and usually set to control an angular motion of between 0 and 180 degrees.

The amount of power applied to the motor is proportional to the distance it needs to travel. So, if the shaft needs to turn a large distance, the motor will run at full speed. If it needs to turn only a small amount, the motor will run at a slower speed.

The control wire is used to communicate the angle. The angle is determined by the duration of a pulse that is applied to the control wire. In other words, the duration of the pulse dictates the angle of the output shaft.

The details on servo motor and its control is discussed in Chapter 13, in which we introduce the PWM module of 16F877.

2. DC Motor Control

Control by Relay (or Electronic Switch)

A relay (or magnetic relay or magnetic switch) is a switch operated by an electromagnetic action. The relatively small current flowing through a coil of an electromagnet inside pulls (or pushes) a lead contact to make (or break)a circuit. No current would push (or pull) back the contact by the mechanical spring attached to the mechanism. The contact lead is made of very low resistive material it does not consume much power. The major problem with using a relay is that, since the contact is physically moving from a position to the other, the response time is longer that

electronic device. It takes 1 few milliseconds. Also, numerous makes and breaks of the contact would eventually wear out the contact leads. The electrical life of a relay is about 1,000,000 times of switch action.

The minimum current to energize the electromagnet and thus to have switch operation is not the same. Different relays and relay manufactures have different specifications. The maximum current 16F877 can provide at its output pin is only 25 mA. Therefore, we can directly operate a relay whose energizing current is below 25 mA. The energizing current is usually, indirectly, indicated by the coil voltage across the electromagnet and power consumption. In other words, if the power consumption by the electromagnet is 100mW with rated coil voltage is 5V, then we can calculate the nominal current through the coil is 20 mA. But if you get or buy a relay, the relay comes alone, without a specification. If it does come with a specification, the specification would be scant and not helping. The best way to test your relay's minimum energizing current is to apply a DC source, from low voltage to higher, and measure the minimum current when you hear a tick sound of contact making or breaking. You do not have to apply the rated voltage across the coil. Staring from 1V you gradually increase the voltage and read the reading of the current, and get the number when you hear the contact sound. That current is the minimum driving current we have to supply. For example, for a relay made by American Zellter, AZ831-2C-5DSE had rated voltage of 5V and 125ohm of coil resistance. A simple calculation says the nominal coil current is 40mA. But it does mean that the relay does not work with coil current of under 40mA. The test with the same procedure described above, the minimum current for coil is found to be only 17mA. So this relay is good for our 16F877 application.

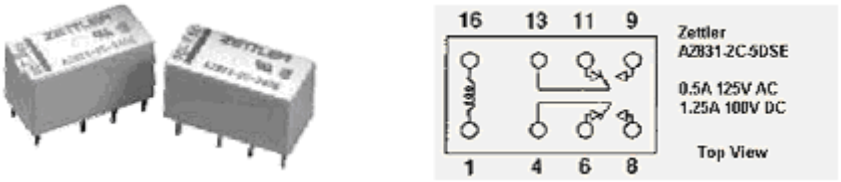


Fig. 36 American Zellter AZ831-2C-5DSE relay

However, there also is the limitation of the maximum current we flow to the coil: above the maximum coil current would damage the electromagnet coil. However, for our case of using a microprocessor, this maximum coil current is of no concern except in very rare cases. Another limitation we have to consider is how much current the contact lead, which would be a part of a circuit when it makes a circuit, can flow without damaging the contact. This value is usually called the "maximum contact current" or "maximum carrying current" in the specification. This value must be higher than your motor current, if you want to use a relay for motor control.

Let's further look at the internal schematic of the AZ831 relay described above. First, we see the coil between pins 1 and 16. There are two contacts, however, connected to pins 4 and 13, respectively. So when the coil is energized each contact (or pole) will move to pins 8 and 9, respectively. When the coil is not energized they will move back to the pins 6 and 11 respectively. In other words, if we use terms used for mechanical switch, the above relay is of Double Pole Double Throw (DPDT): two contacts and two switching positions.

On the other hand, a normal push button we use for temporary contact, which bounces back when it is released, is a Single Pole Single Throw (SPST) switch, with 1 contact and 1 position.

Usual On/Off switch is of Single Pole Double Throw (SPDT) with two positions but with one contact.

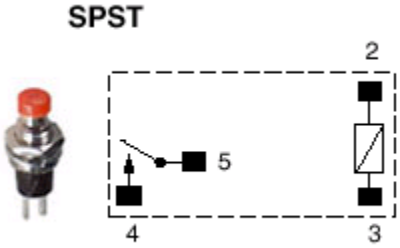


Fig. 37(a) Single Pole Single Throw (SPST) switch

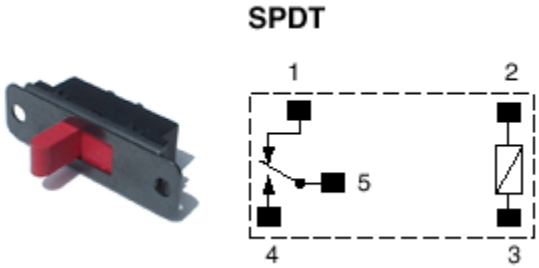


Fig. 37(b) Single Pole Double Throw (SPDT) switch

Before applying relays to control a DC motor, let's examine the principle of DC motor operational control, namely, rotational direction and start/stop. Start/Stop control is rather a simple matter. Turn on or off a SPST switch or a relay of a DC voltage source. However, changing the polarity of a motor terminal is not that simple and straight. The schematic below is to control the direction of a DC motor. As you see the motor power supply is connected to pins 9 and 8 for positive polarity, and to pins 8 and 11 for negative polarity. Therefore, when the DPDT relay is not energized (i.e., no current through the coil between pins 16 and 1, the current from the motor power supply flows through the pins, in the order of, 6 – 4 – Motor's Positive terminal – Motor's negative terminal – 13 – 11 – power supply. Therefore, motor runs in forward motion. When the coil is energized the two contacts move forward to pins 8 and 9, respectively. In this case, the current from the motor power supply flows, in the order of, pin 9 – pin 13 – Motor's negative terminal – motor's positive terminal – pin4 – pin 8 – power supply. Then, the motor turns in reverse motion.

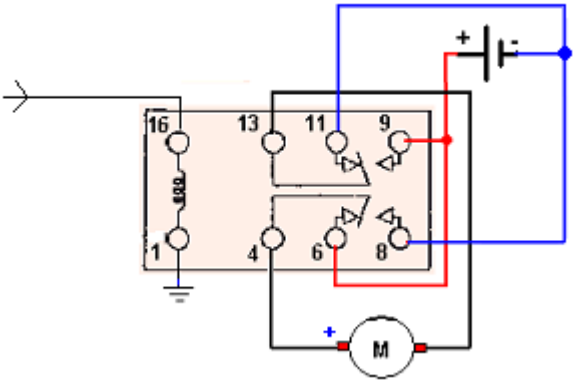


Fig. 38 DC motor operational control

If we add one more DPDT relay (DPDT is not a must. SPDT does well too) for start/stop operation, we have the following final schematic for motor control. Using two of AZ831-2C 5DSE (by American Zellter) relays, we need only two output pins from 16F877 for Forward, Reverse, and Stop motion control for a DC motor.

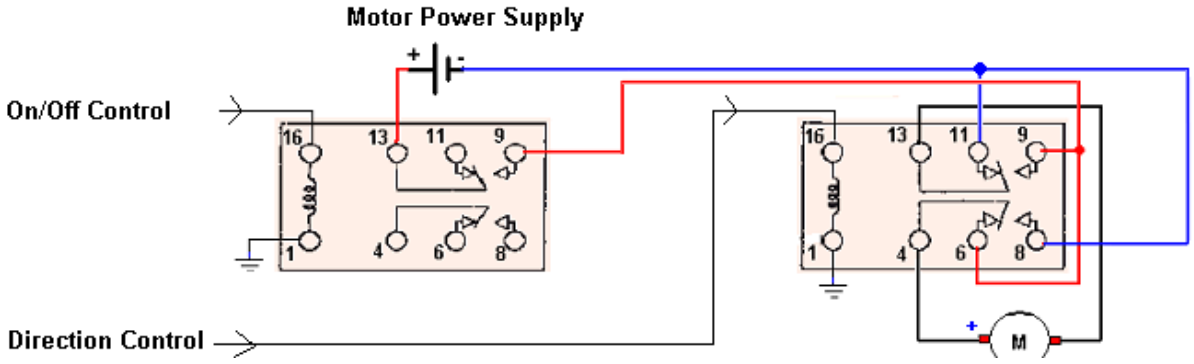


Fig. 39 Motor Power Supply

We choose RD7 (PORTD<7>) and RD6 (PORTD<6>) for On/Off and direction control pins. We added two LEDs (one green at RD0 and the other amber at RD1) as indicators of the DC motor status: Lit green LED for reverse running and the lit amber for forward running. When motor stops, the two LEDs turn off too. Therefore we have the following output table for the following 3 conditions:

Motor Action	PORTD								Remarks
	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	
Forward Motion	1	1	X	X	X	X	1	0	LED1 On /LED0 Off
Reverse Motion	1	0	X	X	X	X	0	1	LED0 On /LED1 Off
Stop	0	X	X	X	X	X	0	0	LED0 Off /LED1 Off

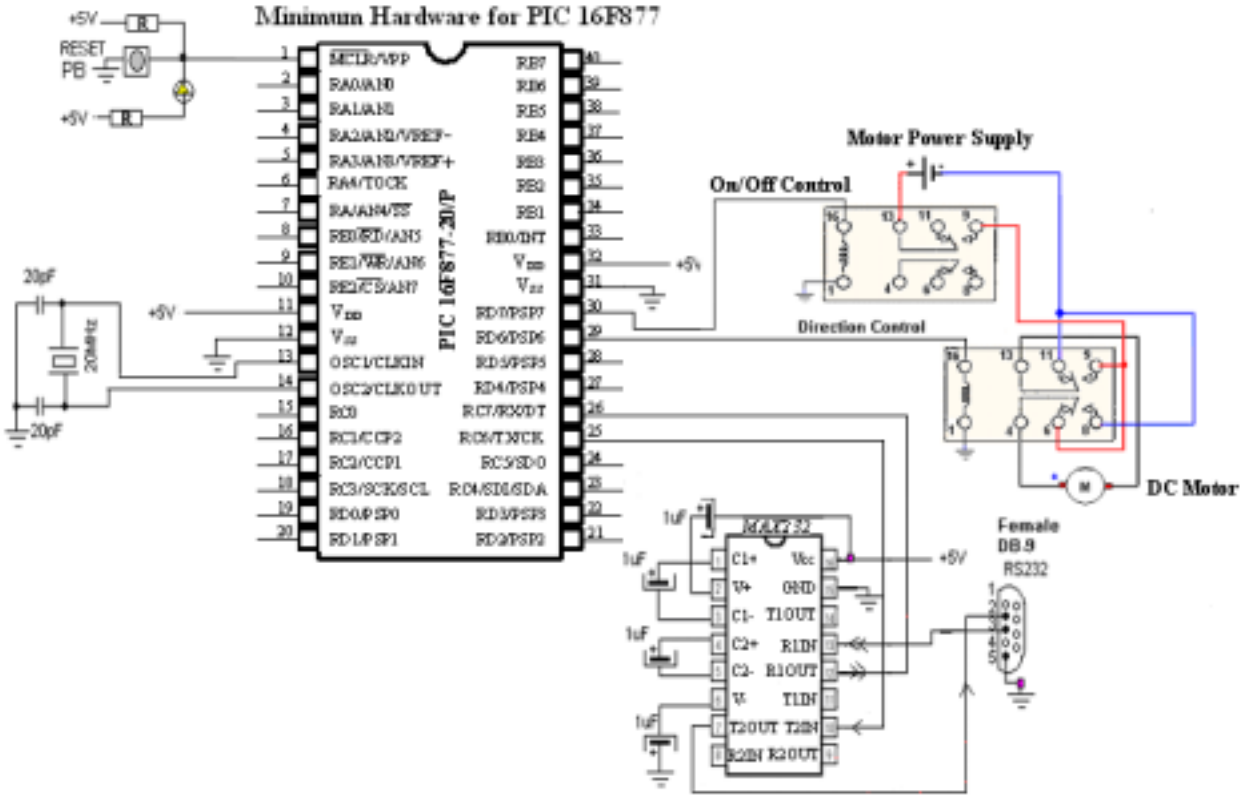


Fig. 40 PIC 16F877 connection to Motor Power Supply

An example code for the above scheme goes like below. The code goes with a sequence of motor motions: 3 seconds of forward, and 3 seconds of reverse, followed by 1 second of stop motions. For forward and reverse motions, 1 second of stop is inserted before running the motor, to give time to stop the on-going motion and its inertia (and reverse electromotive force development and attenuation). The code contains only the main part: subroutine for 1 second delay has been covered before.

```
;DC-RLY.asm
;
;Two DPDT AZ831 relays to control a DC motor
;
; ON/OFF control is connected to RD7
; DIRECTIOn control is connected to RD6
; LED1 is connected to RD1   (FWD motion indication)
; LED0 is connected to RD0   (RVS motion indication)
;
;
;     ACTION            PORTD
;  ============    7 6 5 4 3 2 1 0 =====
; STOP             0 X X X X X 0 0   (00000000)
; ON/FORWARD       1 1 X X X X 1 0   (11000010)
; ON/REVERSE       1 0 X X X X 0 1   (10000001)

        list P = 16F877

STATUS      EQU   0x03
TRISD       EQU   0x88
PORTD       EQU   0x08


;RAM arEA

      CBLOCK      0x20
            Kount120us   ;Delay count (number of instr cycles for delay)
            Kount100us
            Kount1ms
            Kount10ms
            Kount200ms
            Kount1s
            Kount10s
            Kount1m
      ENDC


;Bootloader first execute the first 4 addresses
;=======================================================
      org         0x0000
      goto        START
;=======================================================
      org         0x05
START

      banksel     TRISD
      movlw       0x00
      movwf       TRISD       ;All ports are outputs
      banksel     PORTD
```

```
      clrf        PORTD       ;TWO LEDs OFF (STOP)

AGAIN

; action routine
FORWARD
      banksel     PORTD
      movlw       B'00000010' ;stop for 1 second
      movwf       PORTD
      call        delay1s
      banksel     PORTD
      movlw       B'11000010'
      movwf       PORTD

      call        delay1s
      call        delay1s
      call        delay1s           ;3 seconds of forward

REVERSE
      banksel     PORTD
      movlw       B'00000001' ;stop for 1 second
      movwf       PORTD
      call        delay1s
      banksel     PORTD
      movlw       B'10000001'
      movwf       PORTD
      call        delay1s
      call        delay1s
      call        delay1s           ;3 seconds of reverse

HOLDON
      banksel     PORTD
      movlw       B'00000000'
      movwf       PORTD
      call        delay1s           ;at least 1 second of STOP

      goto        AGAIN
```

Control by Transistors and H-Bridge Drivers

A small base current in a transistor brings up a larger collector current, in turn makes a circuit. A Darlington can generate even more current by cascading two transistor. So a transistor (regular or MOSFET (Metal-Oxide Semiconductor Field-Effect Transistor)) makes a good electronic switch. However, as we discussed above, a DC motor control requires more than just a switch: it needs a combination of switches to change the polarity of the motor terminals for direction control. The most famous transistor control scheme is to use so-called H-bridge. H-bridge is a combination of four transistors, and the name H-bridge came from the shape of the overall circuit with a DC motor at the center.

The diodes, called "fly back diodes" in the H-bridge, are very important in preventing voltage spikes of the motors from destroying the transistors. When a motor rotates and changes direction, the motor winding coils act as a generator and produce a current. This current is called the back electromotiveforce (EMF). This current travels back through the circuit in the form of

powerful voltage spikes to the transistor. Therefore, if this current is particularly large, when you reverse the direction of the motor, it may blow the transistors. The diode's job is then to allow this current to bypass the transistor and travel safely back to the battery.

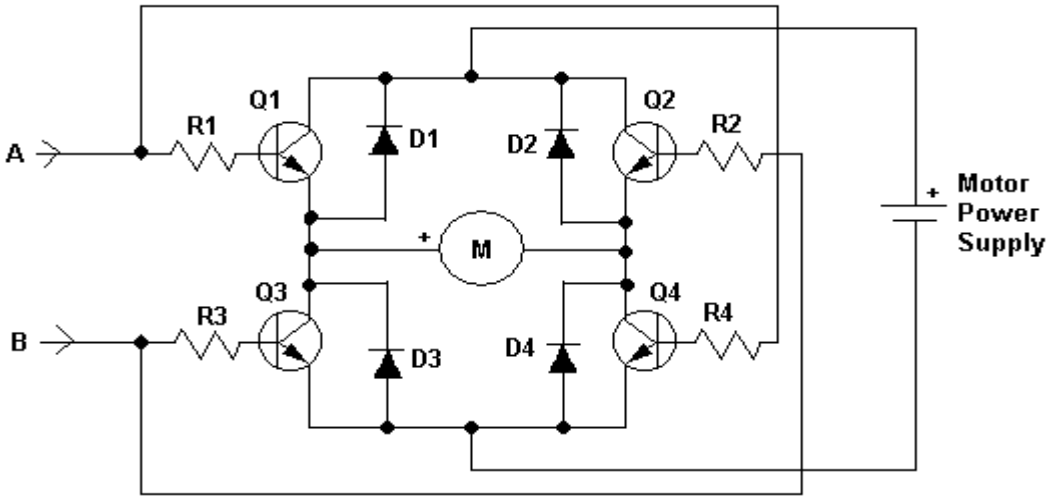


Fig. 41 Control by Transistors and H-Bridge Drivers

When A is High and B is Low, transistors Q1 and Q4 are turned on and Q2 and A3 are turned off. Therefore, the current flows from the motor power supply through Q1 to the positive terminal of the motor and through Q4 to the motor power supply. So with A High and B Low, the motor turns in forward direction.

On the other hand, when A is Low and B is High, transistors Q2 and Q3 are turned on and Q1 and Q4 are turned off. Therefore, current flow direction is, from the motor power supply, through Q2 and the negative terminal of the motor to Q3. In this case, the motor turns in reverse direction.

When both A and B are Low, the motor stops since there is no transistor turned on to flow the current through the motor.

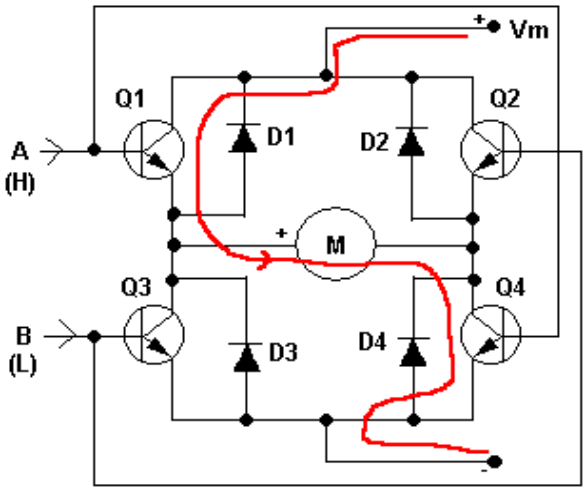


Fig. 42(a) Motor turning in forward direction

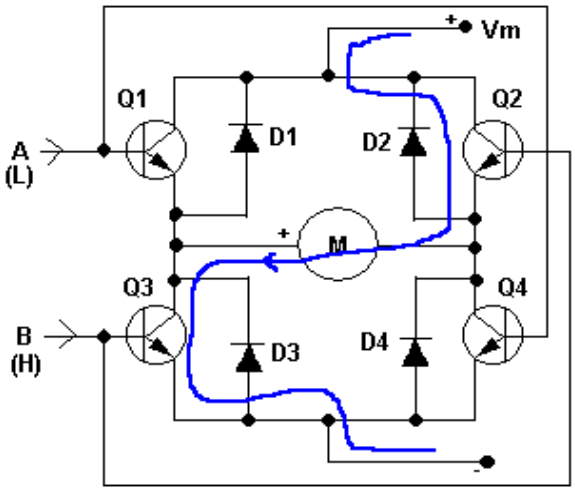


Fig. 42(b) Motor turning in reverse direction

There are numerous transistor types can be formed an H-bridge. The selection criterion must include how much current one stands and how much current a motor needs.

Nowadays, one or more H-bridges are packaged in a chip or motor driver chip which includes other functions such as speed control by PWM. A simple H-bridge motor driver chip contains at least one H-bridge. One of most famous high current (3 A) H-bridge driver is LMD18200 from National Semiconductor. LMD18200 supports motor supply voltages of 12 to 55V and current to 3A. As indicated below, LMD18200T (11 lead TO-220 package) two outputs for one DC motor. The DC motor must be of 12V or above. Motor with lower voltage may not work with LMD18200. Also is a direction input for forward and reverse motion. There is one speed control input, PWM input. As mentioned earlier, the discussion and application of PWM is set aside for a while. Instead, we will supply High (as 100% duty) as a fixed speed. The brake input is to stop the motor. The current sense output is to monitor the current flowing through the outputs, and this is to see if the current is too high for a given condition. Thermal flag output is to indicate the temperature condition of the operation, and is to prevent any excessive operating temperature conditions.

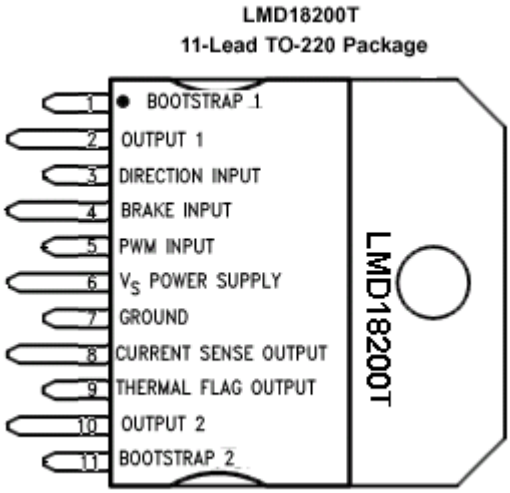


Fig. 43 LMD18200T (11 lead TO-220 package)

In order to drive a DMOS switch ON, its gate must be driven approximately 10V more positive than its source voltage. The lower switches of the H-Bridge have their source terminals connected to ground and their gate drive is derived from the VS motor supply voltage to the device. The two upper switches however have their source terminals connected to the output pins which are continually being switched between ground and VS . In order to generate the gate drive voltage for these switches, a charge pump circuit is used. For higher frequency operation, faster turn-on of the upper DMOS switches is necessary. This can be obtained through the use of external bootstrap capacitors. Since bootstrap capacitor CB is much larger than the input capacitance of the DMOS power transistors, these transistors can turn ON very rapidly, typically in about 100 ns, thus allowing operating the LMD18200 at switching frequencies up to 500 kHz. The recommended capacitance of two bootstrap capacitors is 10 nF.

One distinctive feature of LMD18200 is under-voltage lockout, which disables all of the switches when the DC power supply voltage falls below approximately 10V. The reason for

this feature is that reliable, well controlled operation of the switches cannot be assured without at least 10V applied. Again, therefore, we have to use a DC motor of 12V or above.

According to the datasheet of LMD18200, the following logic table can be formed.

PWM pin	DIR pin	BRAKE pin	MOTOR ACTION
H	H	L	Forward run
H	L	L	Reverse run
X	X	H	Stop

Let's connect one LMD18200 to our 16F877 and one 12V DC motor. The three control inputs are connected to the three pins of PORTD.

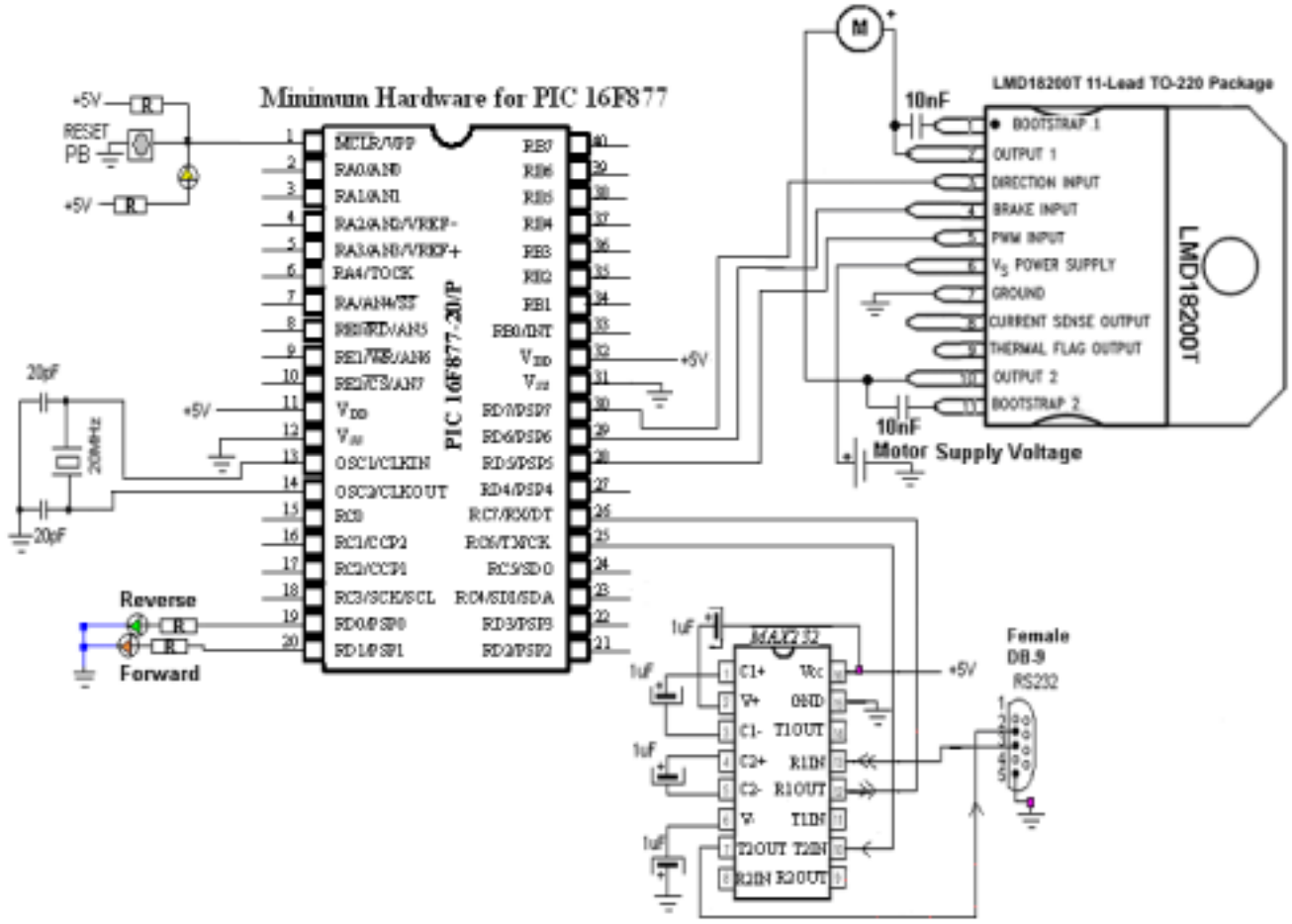


Fig. 44 PIC 16F877 connection to LMD 18200

The sample code we will examine is to, without external control inputs, turn the DC motor in forward motion for 10 seconds and reverse motion for 10 seconds followed by 3 seconds of stop. In both the forward and reverse motion, 1 second of stop is added at the end to minimize any reverse EMF effect to the circuit. Also, it's almost impossible to have an instant direction change because of the inertial force of the motor motion. Again here, two LEDs are connected to the lower two pins of PORTD to indicate the motor actions. As before, only the main part of the code is shown here.

```
;This program is to
; Control a DC (+12 V or abve) motor using LMD18200
; PWM from PIC is not utilized for this program
;
; PWM is connected to RD7  (Keep High for non-PWM used)
; BREAK is connected to RD6 (H means stop L means Go)
; DIRECTIOn control is connected to RD5  (H for FWD)
; LED1 is connected to RD1   (Forward motion indication)
; LED0 is connected to RD0   (Reverse motion indication)
;
;

;     ACTION            PORTD
;   ============    7 6 5 4 3 2 1 0 =====
;  STOP             0 1 0 0 0 0 0 0
;  ON/FORWARD       1 0 1 0 0 0 1 0
;  ON/REVERSE       1 0 0 0 0 0 0 1

        list P = 16F877

STATUS      EQU   0x03
TRISD       EQU   0x88
PORTD       EQU   0x08


;RAM area

      CBLOCK      0x20
            Kount120us   ;Delay count (number of instr cycles for delay)
            Kount100us
            Kount1ms
            Kount10ms
            Kount200ms
            Kount1s
            Kount10s
            Kount1m
      ENDC
;=====================================================
      org         0x0000
      goto        START
;=====================================================
      org         0x05
START
      banksel     TRISD
      movlw       H'00'
      movwf       TRISD       ;All ports are outputs
      banksel     PORTD
      movlw       B'01000000'
      movwf       PORTD       ;STOP Condition
AGAIN

; action routine
FORWARD
      banksel     PORTD
      movlw       B'10100010'
      movwf       PORTD
```

```
      call        delay10s
      banksel     PORTD
      movlw       B'01100000'
      movwf       PORTD       ;STOP
      call        delay1s


REVERSE
      banksel     PORTD
      movlw       B'10000001'
      movwf       PORTD
      call        delay10s
      banksel     PORTD
      movlw       B'01000001'
      movwf       PORTD
      call        delay1s


HOLDON
      banksel     PORTD
      movlw       B'01000000'
      movwf       PORTD
      call        delay1s           ;at least 3 seconds of Stop
      call        delay1s
      call        delay1s

      goto        AGAIN             ;repeat
```

DC motor control using another H-Bridge Driver

The H-Bridge motor driver is UDN2916 Dual Full-Bridge PWM Motor Driver from Allegro MicroSystems. The UDN2916 is designed to drive both windings of a bipolar stepper motor or bi-directionally control two DC motors, and is capable of sustaining 45 V and include internal pulse-width modulation (PWM) control of the output current to 750 mA. The driver includes both ground clamp and flyback diodes for protection against inductive transients. Internally generated delays prevent cross-over currents when switching current direction. Thermal protection circuitry disables the outputs if the chip temperature exceeds safe operating limits. UDN2916 comes with three package types. The UDN2916B is supplied in a 24-pin dual in-line plastic (DIP) package with a copper lead-frame and heat sinkable tabs for improved power dissipation capabilities. The UDN2916EB is supplied in a 44-lead power PLCC for surface mount applications. The UDN2916LB is supplied in a 24-lead surface-mountable SOIC.

Two logic level inputs (I0 and I1) allow digital selection of the motor winding current at 100%, 67%, 33%, or 0% of the maximum level. The 0% output current condition turns OFF all drivers in the bridge and can be used as an OUTPUT ENABLE function.

I0	I1	%of Max Current	
0	0	100	Can be used as "Enable" output
1	0	60%	
0	1	30%	
1	1	0%	Can be used as "Disable" output

Output current is sensed and controlled independently in each bridge by

an external sense resistor (R_S), internal comparator, and monostable multivibrator. When the bridge is turned ON, current increases in the motor winding and it is sensed by the external sense resistor until the sense voltage (V_{SENSE}) reaches the level set at the comparator's input. However, in most application of motor drive, this can be ignored without any problem. So in our example of a DC motor driving, we will simply ignore the current sensing part.

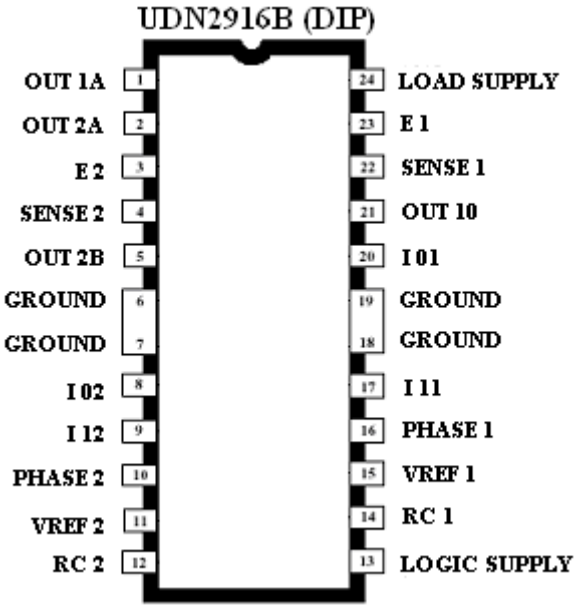


Fig. 45 UDN2916B

The PHASE input to each bridge determines the direction motor winding current flows. An internally generated dead-time (approximately 2 μs) prevents crossover currents that can occur when switching the PHASE input.

PHASE	OUT_A	OUT_B	
1	H	L	OUT_A terminal is positive
0	L	H	OUT_B terminal is positive

In this section we will use UDN2916B for a DC motor application. The driver works well with a wide range of motor supply voltages. The usual 1.5 V or 3 V small DC motors we use often in a robot project run very well with UDN2916B. Moreover, one UDN2916, since it contains dual H-Bridges, can control two DC motors. In the section of a stepper motor control, we use the same UDN1816B for a bipolar stepper motor control.

Here is our strategy of using UDN2916B for a DC motor control. If you want to use two DC motors, just make the same connection for the other motor as we do here for the first motor. However, use caution since pins of the same functions are not aligned side by side. Connection of two DC motors will follow after the discussion of one motor connection and control.

For PWM current control, we will use the I0 and I1 pins for Start (Enable) and Stop (Disable) inputs. So we tie them together and connect to an output pin of 16F877. PHASE pin is used for direction control. When we connect the OUT_A to the positive terminal of a motor (actually, there

is no making on a motor. So you decide which terminal is for forward or reverse turn) and OUT_B to the negative terminal of a motor, then High to PHASE input will turn the motor in a forward motion, and Low in a reverse motion. So the control of a DC motor with a UDN2916 is very simple.

As shown in the schematic, only the left side of H-Bridge is used for a DC motor control. On/Off control with two tied lines of I0 and I1 are connected to RD7, and the direction control PHASE is connected to RD6. The On/Off control is more like an active Low pin in that Low to the On/Off pin will start the motor, and High would stop the motor. As before, two LEDs are connected to indicate the motor action status in RD1 and RD0. As in the DC motor control by a relay, we can get the following control table.

Motor Action	PORTD								Remarks
	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	
Forward Motion	0	1	X	X	X	X	1	0	LED1 On /LED0 Off
Reverse Motion	0	0	X	X	X	X	0	1	LED0 On /LED1 Off
Stop	1	X	X	X	X	X	0	0	LED0 Off /LED1 Off

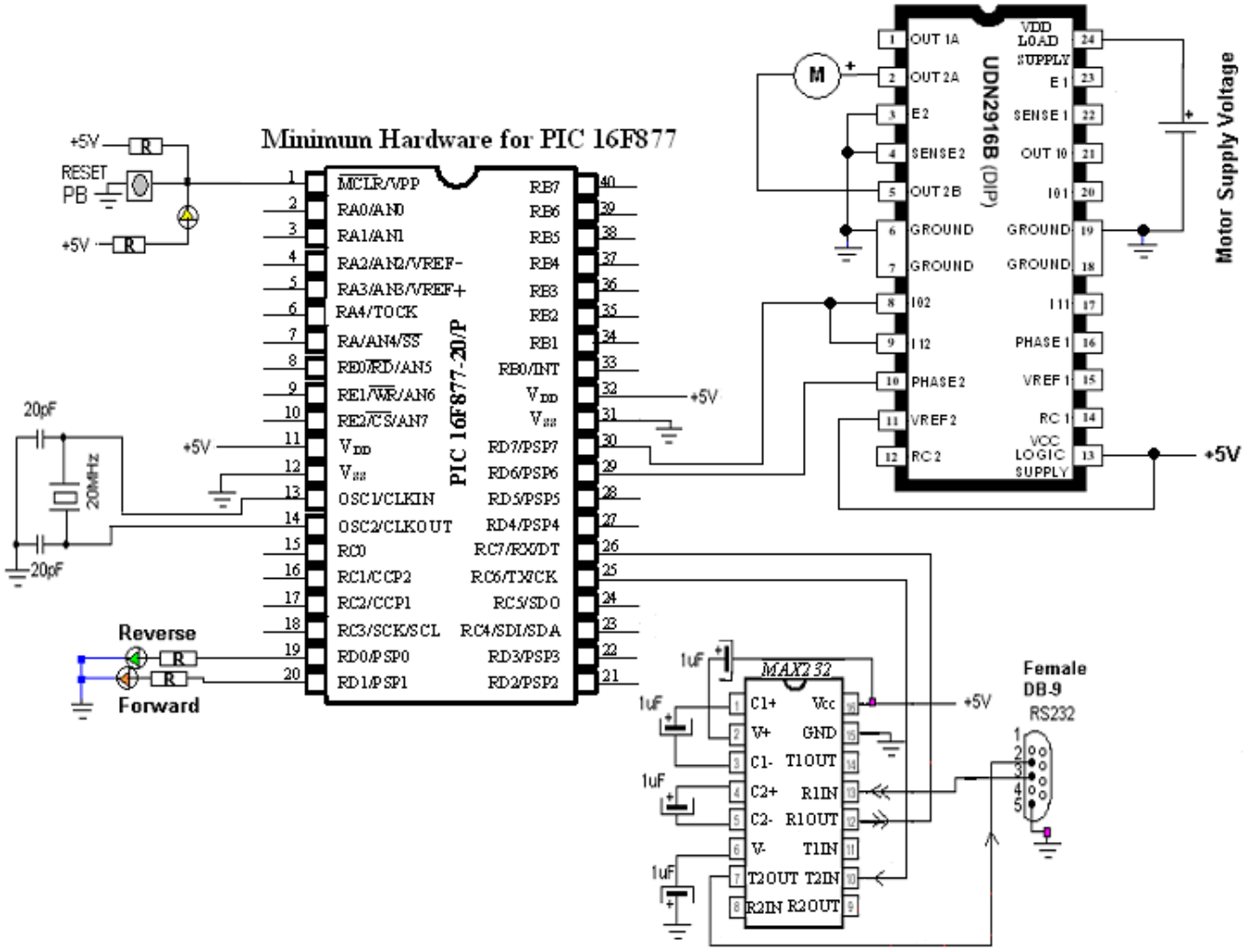


Fig. 46 PIC 16F877 connection to UDN2916B

So the coding is similar and simple. Forward run goes like this.

```
FORWARD
      movlw       B'01000010'
      banksel     PORTD
      movwf       PORTD
      call        delay1s
      call        delay1s     ;or as long as you want it to run
```

And the reverse motion is like this:

```
REVERSE
      movlw       B'00000001'
      banksel     PORTD
      movwf       PORTD
      call        delay1s
      call        delay1s           ;or as long as you want it to run
```

The stopping can be coded like this:

```
HOLDON
      movlw       B'10000000'
      banksel     PORTD
      movwf       PORTD
      call        delay1s           ;at least 1 second
                                    ;or as long as you want it to hold on
```

With two sets of a dual motor gearbox kit and two 56mm sport tires from Lynxmotion, we can easily build a platform for a four-wheel driving robot. And control of your robot (without any sensor, for the time being) is quite easy with two UDN2916B and two 3V battery packs (two 1.5 V battery in series).

Fig. 47(a) Dual motor gearbox kit

Fig. 47(b) 56mm sport tires

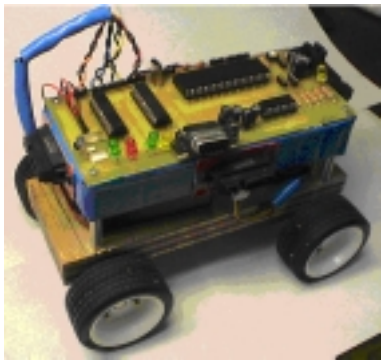


Fig. 4(c) Four-wheel driving robot

The first UDN2916B controls two front motors (Front Left (FL) and Front Right (FR)) and the second UDN2916B does for two rear motors (Rear Left (RL) and Rear Right (RR)). The On/Off control and Forward/Reverse (FW/RV) direction control lines from each motor, total 8, are connected to the PORTD pins. The example here runs all four wheels all the time, however, for a battery saving measure; you may want to run only two wheels all the time except climbing a slope or being with a heavy load.

Let's consider the maneuvering of the 4-wheel platform before we consider a coding example for

16F877. Stopping the platform is easy: turn off all four motors. Going forward is also simple with turning on all four motors and turning them all forward motions. Backing off is also simple with turning on all the motors and reversing them all. Turn left or right has many different operational options but the easiest one is: Run forward two right motors, front and rear, and stop two left motors, front and rear for "turn left" motion; and run forward two left motors, front and rear, and stop two right motors, front and rear for "turn right" motion. So here goes the table for the 4-wheel driving platform.

	Front Left Motor		Front Right Motor		Rear Left Motor		Rear Right Motor	
	ON/OFF	FW/RV	ON/OFF	FW/RV	ON/OFF	FW/RV	ON/OFF	FW/RV
	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0
STOP	1	X	1	X	1	X	1	X
GO FORWARD	0	1	0	1	0	1	0	1
GO BACKWARD	0	0	0	0	0	0	0	0
TURN LEFT	1	X	0	1	1	X	0	1
TURN RIGHT	0	1	1	X	0	1	1	X

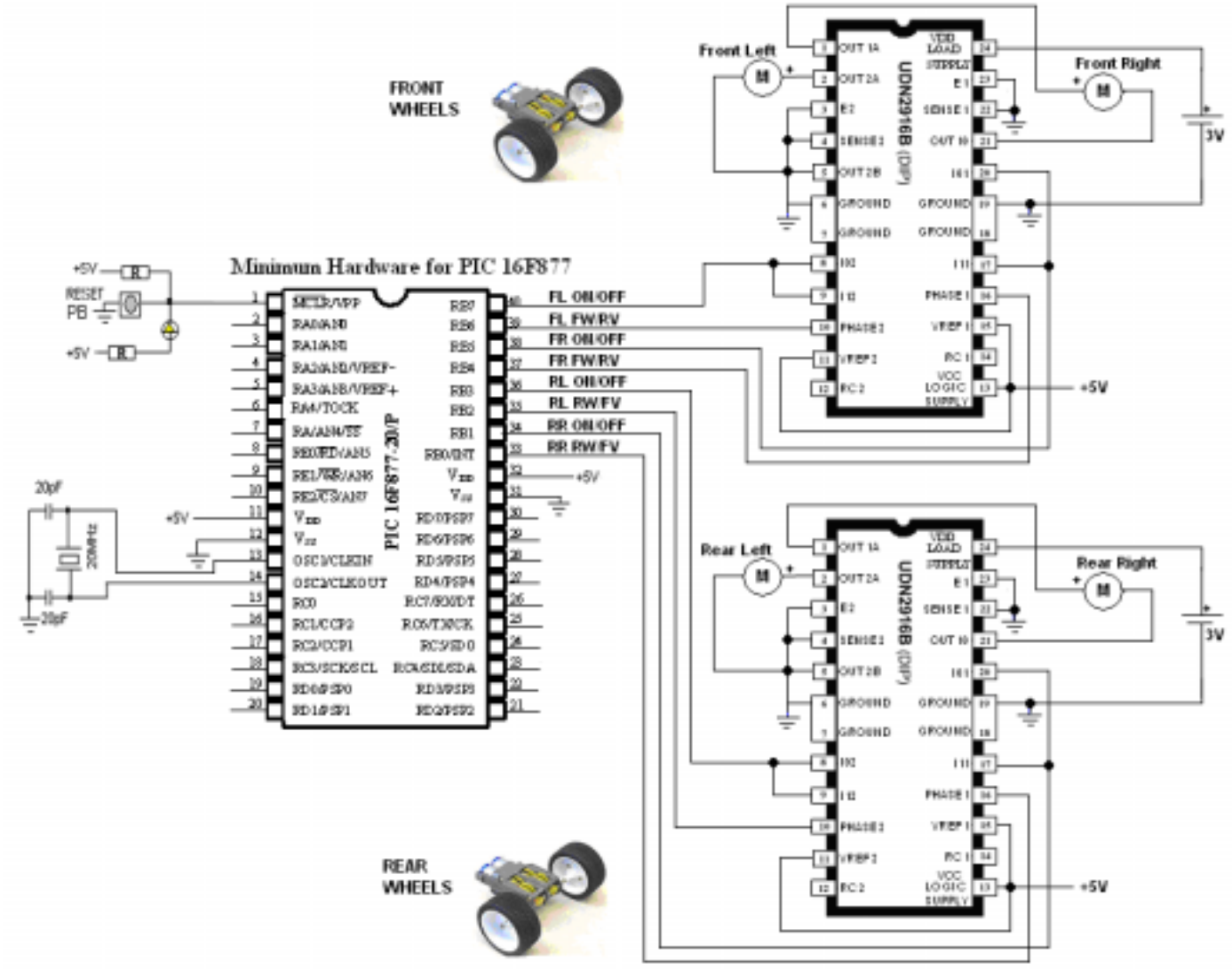


Fig. 48 Connection of PIC 16F877 to two UDN2916Bs

Then the initialization of PORTD and the motion of the platform go as follows. First we set the PORTD pins all as outputs.

```
      banksel     TRISD
      movlw       0x00
      movwf       TRISD       ;all pins of PORTD set as outputs
```

We do not want to see motor moving when we turn on the power, so we have to have a stop condition when power on reset mode.

```
      banksel     PORTD
      movlw       B'10101010'       ;all for motors are off
      movwf       PORTD
```

Then each motion could be easily coded.

```
go_fwd      movlw       B'01010101'
            movwf       PORTD
            return

go_bwd      movlw       B'00000000'
            movwf       PORTD
            return

r_turn      movlw       B'01100110'
            movwf       PORTD
            return

l_turn      movlw       B'10011001'
            movwf       PORTD
            return

stop        movlw       B'10101010'
            movwf       PORTD
            return
```

3. Stepper Motor Control

There are two distinctive stepper motors and their control methods and control drivers. Bipolar stepper motors are to be controlled by bipolar stepper drivers and unipolar steppers are to be controlled by unipolar drivers. Therefore, before you jump to get a stepper motor, you have to consider which stepper motor type you get depending upon control difficulties, external circuitry requirement in addition to a driver, toque, and price. The stepper motors presented here are strong enough at least to turn a monitor-top camera (for a possible tracking camera application). If you are considering a stepper motor for a vehicle steering application, you had better consider a servo instead. For servo, please wait until or jump to Chapter 13 PWM. However, some steering application for 360degree turn, a stepper motor could be handier than a servo.
As will be shown soon, some drivers are easier to control than others, while another driver is simpler to implement in terms of necessary external elements such as resistors and capacitors than others. So, based on your purpose, choose a stepper motor first then get a driver for the motor and control them.

Bipolar Stepper Motor Control

Biploar stepper motor's distinctive feature is that it has four lead lines. As you see pictures below (From left to right: Airpax 5V Bipolar Motor LB82773-M1, Symbol Technologies Bipolar Motor 21-02485, and a 12 V head motor inside a floppy disk driver) the four leads line are connected to two winding coils inside of the motor.

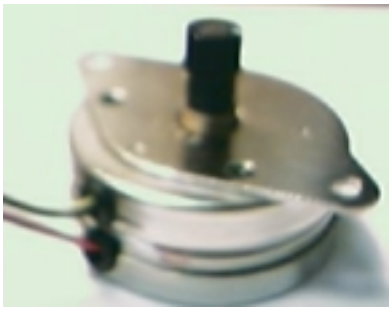


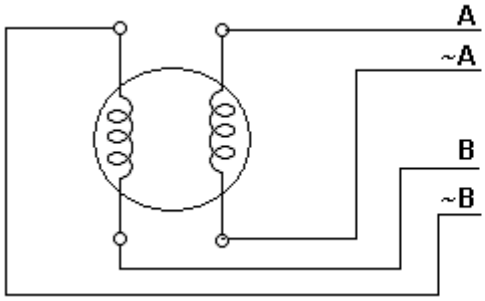
Fig. 49(a) Airpax 5V Bipolar Motor LB82773-M1

Fig. 49(b) Symbol Technologies Bipolar Motor 21-02485

Fig. 49(c) 12 V head motor inside a floppy disk driver

How much angle one step turn covers is provided by stepper motor manufactures. For example, the Airpax bipolar stepper motor has 7.5 degree/step characteristics. In other words, one step turns the rotor of the motor 7.5 degrees. Therefore, for a full one turn, it needs 48 steps. Again, for each step, we have to provide given pulses of polarity for four times following the sequence depending upon the direction of the turn. Any bipolar driver, therefore, has a way of changing polarities at its outputs.

The two coils must excited with a given sequence for a turn in a forward/reverse motion. As illustrated below, for forward step turns, the correct sequence of pulses must be provided to the coils in the order of 1 – 2 - 3 - 4 - 1. The each pulse will turn one step. Therefore, the one sequence of pulses turns four steps, or 7.5°x4 = 30°. For reverse step turns the order of excitation sequence should be reversed so that the excitation sequence of 1 - 4 - 3 – 2 - 1. Each pulse turns one step. Again, the sequence of 4 pulses turns the rotor 4 steps, or 7.5°x4 =30° for the Airpax bipolar stepper motor. To make one full turn, then, needs 12 sequences of four pulses.



Excitation Sequence for a Step Turn

	A	B	~A	~B
1	H	H	L	L
2	L	H	H	L
3	L	L	H	L
4	H	L	L	H

The UDN2916B we used in the DC motor control is designed to drive both windings of a bipolar stepper motor or bi-directionally control two DC motors. So for the Airpax bipolar motor, we apply the same driver. Since a bipolar motor has two windings we have to use both of the H-Bridges inside the UDN2916B.

Since the PHASE input to each bridge determines the direction motor winding current flows, we use PHASE1 and PHASE2 to control of the current flow directions for coil A & ~A pair and B & ~B pair, respectively. Then, we connect OUT_{1A} to A and OUT_{1B} to ~A of the motor, and OUT_{2A} to B and OUT_{2B} to ~B. Then, we can draw the following polarity table for two coils depending upon the PHASE values.

Coil Leads		**A**	~A	**B**	~B
PHASE1	PHASE2	**OUT_{1A}**	OUT_{1B}	**OUT_{2A}**	OUT_{2B}
0	0	**L**	H	**L**	H
0	1	**L**	H	**H**	L
1	0	**H**	L	**L**	H
1	1	**H**	L	**H**	L

Now, can you see the sequence for PHASE1 and PHASE2 logic values for a forward step turn of the motor? From the sequence table of the stepper motor for A and B (~A and ~B are just in complement relationship to the other end of the coil, respectively), you see that, for a forward turn, the sequence of (PHASE1, PHASE2)=(1, 1) should come first, which will give positive polarities for both A and B lead of the motor. The next excitation must be with L for A and H for B, (PHASE1, PHASE2)=(0,1) comes next.

Therefore a sequence map for PHASE1 and PHASE2 for forward and reverse one step turn can be drawn as follows:

	Forward one step turn		Reverse one step turn	
Sequence	PHASE1	PHASE2	PHASE1	PHASE2
1	1	1	1	1
2	0	1	1	0
3	0	0	0	0
4	1	0	0	1

Now, let's connect the Airpax bipolar motor to 16F877. Your moor supply voltage should come from a separate battery or power source rather than using the same +5V voltage source for your PIC board. As you see from the schematic, all for On/Off (or speed) control lines, I01, I02, I10, and I12, are tied together as a On/Off control input for the bipolar motor.

As seen in the schematic diagram, PHASE1 is connected to RD7, PHASE2 to RD6, and all I's together to RD5. Also the two LEDs are connected to RD1 and RD0, respectively, for rotational direction indication. Before examining a sample code, let's make a few subroutines based on the discussion we already had above on the step rotation direction. Based on the sequence table, let's make a subroutine for one forward sequence, `fsequence,` which generates 4 pulses in order.

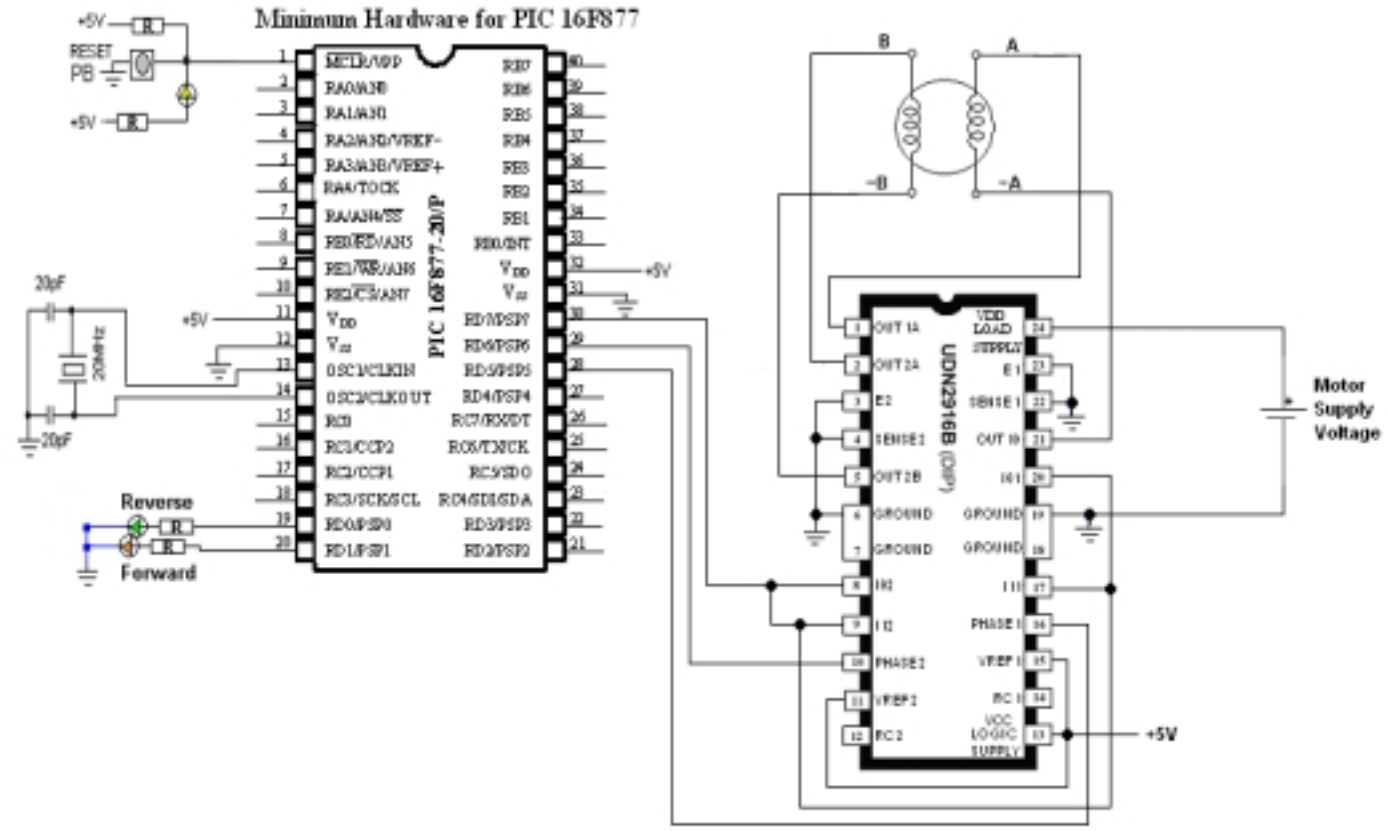


Fig. 50 Airpax bipolar motor connection to PIC 16F877

```
;  1 FORWARD sequence subroutine (will turn 4 steps)
;   A  B       7 6 5 4 3 2 1 0 (PORTD)
;   +  +   --->1 1 0 0 0 0 1 0
;   -  +   --->0 1 0 0 0 0 1 0
;   -  -   --->0 0 0 0 0 0 1 0
;   +  -   --->1 0 0 0 0 0 1 0
fsequence
      banksel     PORTD
      bcf         PORTD, 0x05       ;100% power (Start Condition)
      movlw       B'11000010'       ;1st excitation with LED1 on
      movwf       PORTD
      call        delay10ms
      call        delay10ms         ;this time delay determines
                                    ;how fast one step finishes
                                    ;longer delay slows the motor turn
      banksel     PORTD
      movlw       B'01000010'       ;2nd excitation with LED1 on
      movwf       PORTD
      call        delay10ms
      call        delay10ms
      banksel     PORTD
      movlw       B'00000010'       ;3rd excitation with LED1 on
      movwf       PORTD
      call        delay10ms
      call        delay10ms
      banksel     PORTD
      movlw       B'10000010'       ;4th excitation with LED1 on
```

```
      movwf       PORTD
      movlw       B'11000000'       ;back to the fist pulse with LEDs off
      movwf       PORTD
      call        delay10ms
      call        delay10ms
      return
```

The one reverse sequence subroutine, `rsequence`, can be similarly drawn for four pulses in the reverse order.

```
; 1 REVERSE sequence subroutine (will turn 4 steps)
;   A  B       7 6 5 4 3 2 1 0 (PORTD)
;   +  +   --->1 1 0 0 0 0 0 1
;   +  -   --->1 0 0 0 0 0 0 1
;   -  -   --->0 0 0 0 0 0 0 1
;   -  +   --->0 1 0 0 0 0 0 1
rsequence
      banksel     PORTD
      bcf         PORTD, 0x05       ;Start (100% power)
      movlw       B'11000001'       ;1st excitation with LED0 on
      movwf       PORTD
      call        delay10ms
      call        delay10ms
      banksel     PORTD
      movlw       B'10000001'       ;2nd excitation with LED0 on
      movwf       PORTD
      call        delay10ms
      call        delay10ms
      banksel     PORTD
      movlw       B'00000001'       ;3rd excitation with LED0 on
      movwf       PORTD
      call        delay10ms
      call        delay10ms
      banksel     PORTD
      movlw       B'01000001'       ;4th excitation with LED0 on
      movwf       PORTD
      call        delay100ms
      movlw       B'11000000'       ;back to the fist pulse with LEDs off
      movwf       PORTD
      call        delay10ms
      call        delay10ms
      return
```

The following code is to turn 8 steps to the right, then a few seconds of delay, followed by turn 5 steps to the left. As usual, subroutines are not listed in the example code.

```
;BPmotor.asm
;
Motor Control Chip is ALLEGRO UDN2916B (for Bipolar Stepper & DC control)
;
;
; Phase 1 control (for A and A~ coil) is connected to RD7
; Phase 2 control (for B and B~ coil) is connected to RD6
; PWM control lines (I01, I02, I11, and I12) are tied to RD5
; LED1 is connected to RD1   (RVS motion indication)
; LED0 is connected to RD0   (FWD motion indication)
;
;

        list P = 16F877
```

```
STATUS      EQU   0x03
TRISD       EQU   0x88
PORTD       EQU   0x08

;RAM area

      CBLOCK      0x20
            TIMEBLOCK
            Kount120us   ;Delay count (number of instr cycles for delay)
            Kount100us
            Kount1ms
            Kount10ms
            Kount200ms
            Kount1s
            Kount10s
            Kount1m
      ENDC


;=======================================================
      org         0x0000            ;line 1
      goto        START       ;line 2 ($0000)
;=======================================================
      org         0x05
START
      banksel     TRISD
      movlw       H'00'
      movwf T     RISD        ;All ports are outputs


AGAIN
      banksel     PORTD
      bcf         PORTD, 0x05       ;Start Condition (100% power)


      call        fSequence               ;forward 5 sequences
      call        fSequence
      call        fSequence
      call        fsequence         ;total 4x4=16 steps (120 degrees)
      bsf         PORTD, 0x05       ;Stop (0% power)
      call        delay1s

      call        rSequence               ;backward 8 sequences
      call        rSequence
      call        rSequence
      call        rSequence
      call        rSequence
      call        rSequence
      call        rSequence
      call        rSequence               ;total 8x4=32 steps (240 degrees)
      bsf         PORTD, 0x05       ;Stop condition (0% power)

      call        delay1s
      call        delay1s
      goto        AGAIN                   ;Continue
```

For other bipolar stepper motors, same schematic and code can apply without any change.

Unipolar Stepper Motor Control:
The coil current direction in the coils of a unipolar stepper motor is always in on direction or no current flow. Therefore each coil is center-tapped to a terminal for positive supply voltage. The From the center-tapped terminal, current flow to either side of the coils (called a phase). Therefore, unipolar stepper motor usually has 6 lead lines: 3 for a coil and the 3 for the other. However, some unpolar stepper comes with only 5 lead lines since the two center-tapped terminals can be conjoined for the positive supply voltage.

The motor power's positive polarity must be connected to the terminals 1 and 2 of the illustration shown below. The control of a unipolar stepper motor is, thus, to turn on and off the current flow through the four coils in a sequence for the step direction and speed. As we see a unipolar stepper motors has at least 5 external wires: 5 when terminals 1 and 2 are jointed together into one terminal and 6 when two are left as separate terminals. Both unipolar stepper motors shown below (Left: KH42HM2-901 from Japan Servo. Right: 5015-824 from Applied Motion Products) have 6 lead lines.

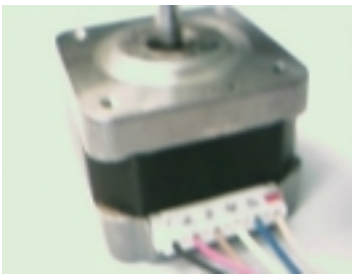


Fig. 51(a) KH42HM2-901 from Japan Servo

Fig. 51(b) 5015-824 from Applied Motion Products

When motor comes without detailed coil and terminal diagram, we can easily check the coil connections by measuring resistance of any two terminals. You would have same reading between a center-tapped terminal and one end of a coil. If your reading between two terminals is infinite, then they are the two center-tapped terminals of the coils.
Stepper motor is characterized by torque, voltage, and resistance/inductance per phase (or half coil). In torque there are two types are involved. The holding or static torque is the basic characteristic of a step motor. In other words, holding torque is a maximum torque that can be applied to the shaft of a motor with one or more phases energized without causing rotation. This characteristic is the restoring torque developed when the motor is forced away (displaced by a load torque) from its true detent position (energized but unloaded rest position). Detent torque is defined as a maximum torque that can be applied to the shaft of an unexcited motor without causing rotation.

The KH42HM2-901 unipolar stepper motor has the following characteristics.

Motor Voltage	3 V
Step Angle	1.8°/Step
Motor Current	0.9 A/Phase
Winding Resistance	3.4Ω/Phase
Inductance	2.4 mH/Phase
Holding Torque	20 oz.in (140 mN.m)
Detent Torque	1.7 oz.in (11.8 mN.m)
Shaft Diameter	0.2 in

Couplers are very useful elements to connect the motor via the shaft to something to be turned. In Jameco.com you can find a few different sets of couplers for different shaft diameters. Using a set of couplers and rubber spider, we can connect a motor shaft to any non-heavy stuff, say, a monitor-top camera for a possible auto-tracking security application. If you add a PIR motion detector sensor, you can make your stepper motor an important element for a smart tracking camera. An additional ranger (distance measuring sensor: this will be discussed in the chapter of A/D conversion) will greatly help to auto-focus and track an object in a security zone.

Fig. 52 Few different sets of couplers for different shaft diameters

The phase coils must be excited in a sequence for a right/left step turn as shown below. The positive terminal of the motor supply voltage (MV) must be applied to both center-tapped terminals (Red and Blue leads). On the other hand, the terminal lead of a phase must have negative polarity when that phase to be excited, with current flowing from a center-tapped terminal to that phase terminal. The sequence of step turn is shown in the excitation sequence. Since the two center-tapped terminals are connected to the voltage source, only four terminals are to be polarized. If a phase is polarized with High, the no current flows through the phase; a Low polarization would enable excitation current to flow through the phase coil. Each sequence is translated into a step movement.

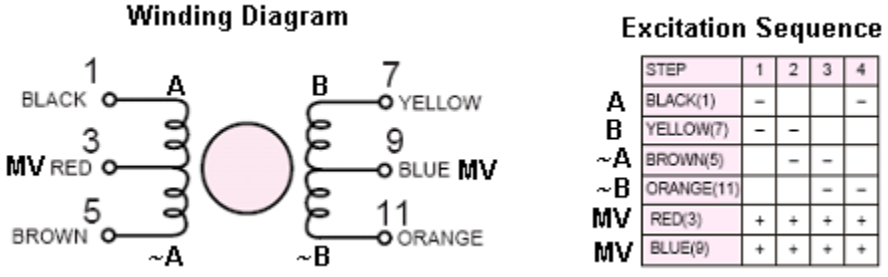


Fig. 53 Winding diagram

For forward step turns, the excitation sequence of 1 – 2 – 3 – 4 – 1 should follow, and the reverse step turns require the sequence of 1 – 4 – 3 – 2 – 1.

The first unipolar driver chip we are going to is Allegro MicroSystems UCN5804. Combining low-power CMOS logic with high-current and high-voltage outputs, the UCN5804 provides complete control and drive for a unipolar stepper-motor with continuous output current ratings to 1.25 A per phase (1.5 A start-up) and 35 V.
CMOS logic section provides the sequencing logic, DIRECTION OUTPUT ENABLE control, and a power-on reset function. The inputs are compatible with standard PMOS, and NMOS circuits. TTL or LSTTL may require the use of appropriate pull-up resistors to ensure a proper input-logic high.
UCN5804 comes with two types of packages. DIP and SOIC. The UCN5804B is supplied in a 16-pin dual in-line plastic (DIP) package with a copper lead frame and heat-sinkable tabs for improved power dissipation capabilities. The UCN5804LB is supplied in a 16-lead plastic SOIC package with a copper lead frame and heat-sinkable tabs. So the natural choice for our example is the 16-pin DIP type UCN5804B.

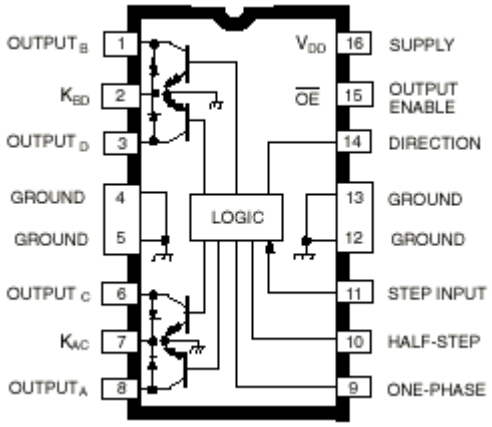
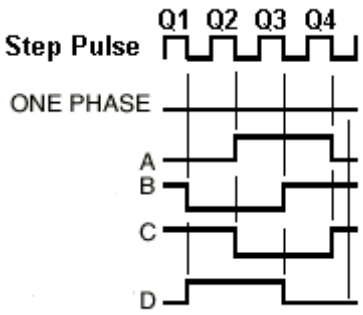


Fig. 54 16-pin DIP type UCN5804B

As seen from the pin diagram, we see four outputs ($OUTPUT_A$ or A, $OUTPUT_B$ or B, $OUTPUT_C$ or C, $OUTPUT_D$ or D) and two inputs of direction and step input. The wave-drive format consists of energizing one motor phase at a time in an A-B-C-D (or D-C-B-A) sequence. This excitation mode consumes the least power and assures positional accuracy regardless of any winding unbalance in the motor. Two-phase drive energizes two adjacent phases in each detent position (AB-BC-CD-DA). This sequence mode offers an improved torque-speed product, greater detent torque, and is less susceptible to motor resonance. Half-step excitation alternates between the one-phase and two-phase modes (A-AB-B-BC-C-CD-D-DA), providing an eight-step sequence. As seen from the timing diagram, the falling edge of the step input pulse changes the output values. Each pulse of step turns one step movement.

	Q1	Q2	Q3	Q4
A	**L**	**H**	**H**	**L**
B	**L**	**L**	**H**	**H**
C	**H**	**L**	**L**	**H**
D	**H**	**H**	**L**	**L**



In our example we follow the two phase excitation at a time. Since the lettering system of the chip is different from usual stepper motor winding, it may be slightly confusing. But if you compare the wave-format of UCN5804 and the required excitation sequence of the stepper motor, we can easily relate the terminals of the motor to the output pins of the UCN5804. However, if you see the excitation and the output sequences of the stepper motor and UCN5804, we can easily see the matching leads of the motor and outputs of UCN5804. In other words, the A lead of the motor is exactly in the same sequence with the A output of UCN5804B, as B lead to B output, and ~A lead to C, and ~B to D output.

Stepper Motor Excitation Sequence

	STEP	1	2	3	4
A	BLACK(1)	–			–
B	YELLOW(7)	–	–		
~A	BROWN(5)		–	–	
~B	ORANGE(11)			–	–
MV	RED(3)	+	+	+	+
MV	BLUE(9)	+	+	+	+

	Q1	Q2	Q3	Q4
A	**L**	**H**	**H**	**L**
B	**L**	**L**	**H**	**H**
C	**H**	**L**	**L**	**H**
D	**H**	**H**	**L**	**L**

By grounding pins 9 and 10, we configure the chip as two-phase control mode. We tie the Output Enable pin to the ground to enable the output. To prevent any current (like back EMF) from flowing toward the output pins, we added four diodes at each of the output pins. Also, the motor supply voltage is directly connected to K pins. Then, the two control inputs of Direction and Step Input, we connect to two pins of PORTD of 16F877, respectively.

Before writing an example code, remember that the UCN5804B chip provides the sequence outputs when there is a High-to-Low transition. Also, according to the datasheet of the driver, the width of the input step pulse must be longer than 6 ns but giving them more time like 1ms would be safe.
The control strategy is rather simple. Forward 1 step will be accomplished by setting RD and sending 1 pulse. Another pulse will move one more step at the same direction. Reverse 1 step would be accomplished by clearing RD6 and sending 1 pulse. The successive pulse would turn the rotor in the reverse direction.

One caution we have to use is that, when we change the direction, RD7 must be in Low state.

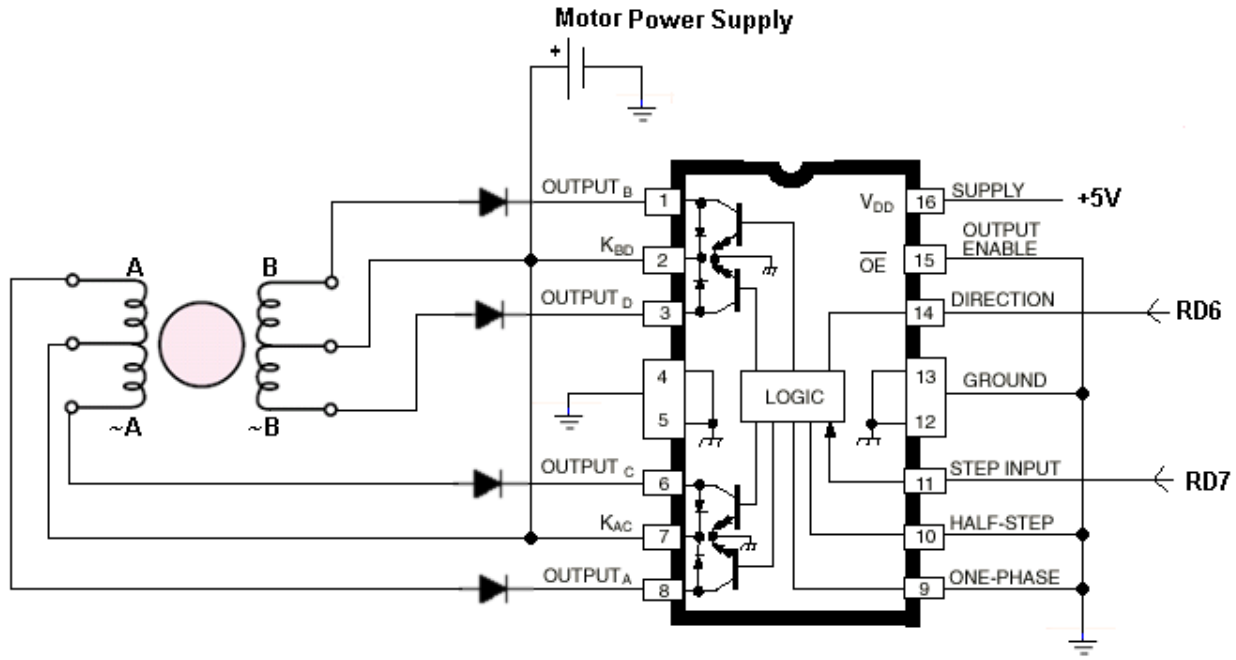


Fig. 55 UCN5804B connection to PIC 16F877

Therefore the first subroutine we need is to have a pulse generation routine.

```
; subroutine PULSE
PULSE
      banksel     PORTD
      bsf         PORTD, STEP       ;STEP = 0x07 (RD7)
      call        delay200ms        ;reduce the time delay for a smooth turn
      banksel     PORTD
      bcf         PORTD, STEP
      call        delay200ms
      return
```

Then the forward and reverse step subroutines are:

```
;subroutine fstep
fstep
      banksel     PORTD
      movlw       B'01000010'       ;RD6=1, start with RD7=0
      movwf       PORTD
      call        PULSE             ;call a pulse for a step
      return

;subroutine rstep
rstep
      banksel     PORTD
      movlw       B'00000001'       ;RD6=0, Start with RD7=0
      movwf       PORTD
      call        PULSE             ;call a pulse for a step
      return
```

Therefore, an example code to turn the motor 3 steps to the right and 3 steps to the left would be like below. Subroutines are not included as usual since we already discussed.

```
;up5804.asm
;
;This program is to:
;1. Control a Unipolar Stepper Motor
;2. Turn Forward 3 steps then Reverse 3 steps
;3. Motor Control Chip is ALLEGRO UCN5804B (for Unipolar Stepper control)
;
;
; HIGH-To-LOW PULSE output connected to RD7
; DIRECTION output is connected to RD6
;
; LED1 is connected to RD1   (RVS motion indication)
; LED0 is connected to RD0   (FWD motion indication)
;
;
;  ACTION         PORTD
;
;  1 STEP FORWARD: set RD6 H   then   (H --> L) RD7
;  1 STEP REVERSE: set RD6 L   then  H --> L of RD7
;   To change the direction, RD7 must be in L state
;
;                 PORTD 76543210
;1 STEP     FORWARD     01000010
;                       T         ;transitional pulse
;
;1 STEP REVERSE         00000001
;                       T         ;Transitional pulse

        list P = 16F877

STATUS      EQU   0x03
TRISD       EQU   0x88
PORTD       EQU   0x08
STEP        EQU   0x07



;RAM arEA

      CBLOCK      0x20
            TIMEBLOCK
            Kount120us   ;Delay count (number of instr cycles for delay)
            Kount100us
            Kount1ms
            Kount10ms
            Kount200ms
            Kount1s
            Kount10s
            Kount1m
      ENDC


;================================================
      org         0x0000            ;line 1
      GOTO        START       ;line 2 ($0000)
;================================================
      org         0x05
START

      banksel     TRISD
      movlw       H'00'
      movwf       TRISD       ;All ports are outputs

AGAIN
```

```
banksel     PORTD
movlw       B'00000000'
movwf       PORTD       ;STOP Condition

call        fstep
call        fstep
call        fstep       ;3 forward steps
call        delay1s
call        rstep
call        rstep
call        rstep       ;3 backward steps
call        delay1s
goto        AGAIN       ;continue
```

The next unipolar stepper driver is another Allegro MicSystems chip SLA7024M, high-current PWM,unipolar stepper motor controller/driver. The SLA7024M is designed for high-efficiency and high-performance operation of 2-phase, unipolar stepper motors with rated for an absolute maximum limit voltage of 46 V. Minimum voltage is around 9V. The peak current is 3.0A and the continuois output current is 1.5A. So this lever is much higher than the previous driver, UCN5804 which has 0.75 A as continuous current output. SLA7024M has the package type of 18-lead power-tab SIP (single-in-line) package.

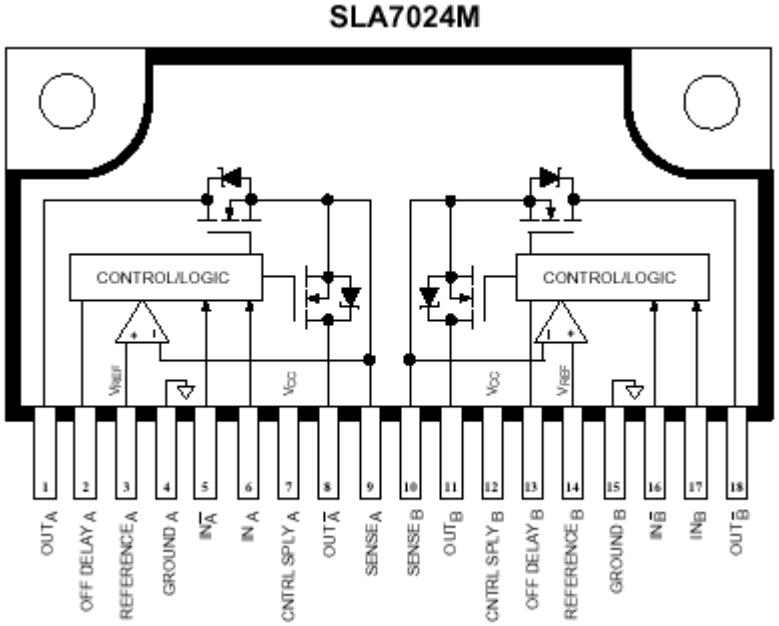


Fig.56 Allegro MicSystems chip SLA7024M

As you see the pin out diagram, for controls, there are four inputs IN_A, $IN_{\sim A}$, IN_B, and $IN_{\sim B}$, and four outputs OUT_A, $OUT_{\sim A}$, OUT_B, and $OUT_{\sim B}$. The subscript labels match the four lead lines of a unipolar stepper motor. As the table below shows for a double phase operation, the input sequence decides the output sequence. For example, the first pulse of the sequence for inputs are (H, L, H, L) and then the OUT_A and OUT_B will be turn on, and the current flows from the motor power supply to these two output pins. On the other hand, the input pulse of (L, H, H, L) will make the motor current flows from the supply voltage to the output pins of $OUT_{\sim A}$ and OUT_B. The only different between UCN5804 an SLA7024, therefore, is that the former needs

just a H-to-L transition pulse for a pulse for step rotation since the internal logic provides the necessary combination of four pulses, while the latter needs actual four inputs for a step rotation.

Sequence	1	2	3	4	1
IN_A	H	L	L	H	H
$IN_{\sim A}$	L	H	H	L	L
IN_B	H	H	L	L	H
$IN_{\sim B}$	L	L	H	H	L
Outputs turned on	A & B	~A & B	~A & ~B	A & ~B	A & B

A full application of SLA7024M requires several external elements of resistors and capacitors. The circuit diagram and schematic introduced here follows the recommended typical motor application circuit. As usual the LEDs are connected to RD1 and RD0, respectively, just to indicate the rotational direction of the step movement.

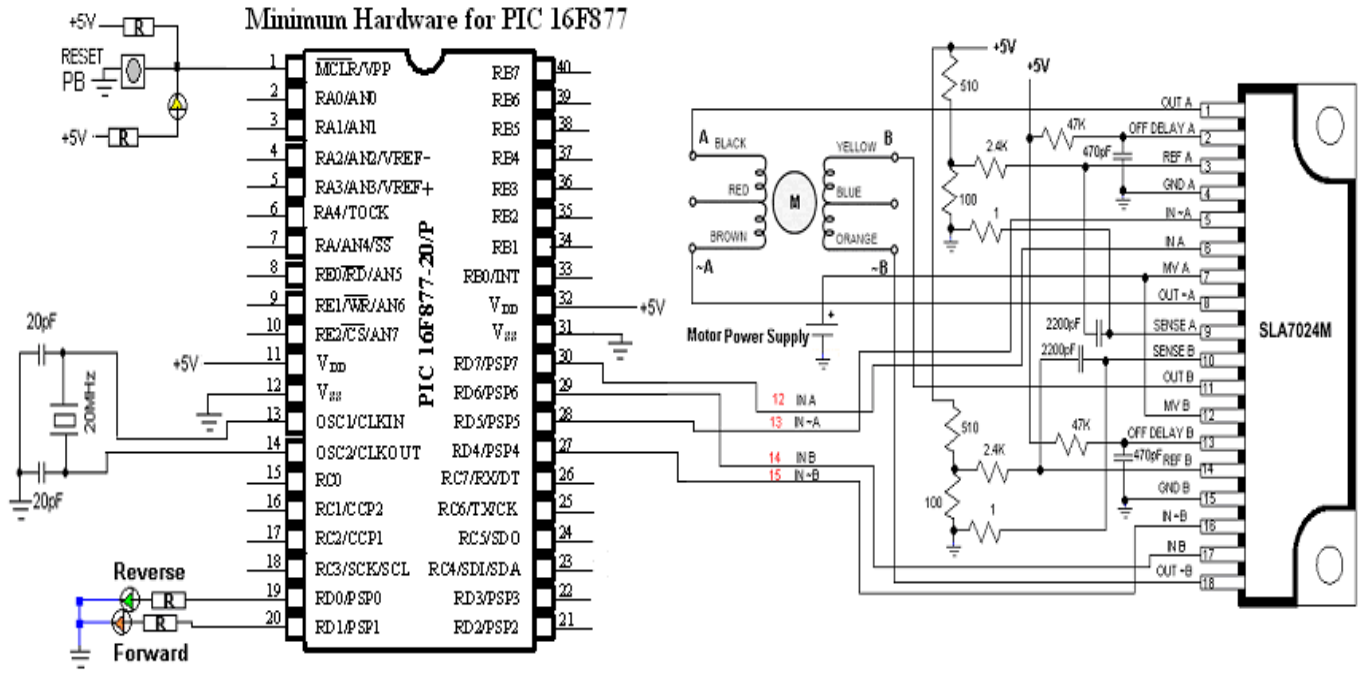


Fig. 57 SLA7024M connection to PIC 16F887

Since the excitation sequence is the same for the stepper motor, whether we use SLA7024M or UCN5804, we apply similar excitation sequence for four step movement of forward and reverse rotation.

```
;====subroutine fSequence===
; 1 FORWARD Sequence of 4 pulses (4 step movement) and LED1 on
;   A  B  ~A  ~B      7 6 5 4 3 2 1 0 (PORTD)
;   +  +   -   -  --->1 1 0 0 0 0 1 0
;   -  +   +   -  --->0 1 1 0 0 0 1 0
;   -  +   -   +  --->0 1 0 1 0 0 1 0
;   +  -   -   +  --->1 0 0 1 0 0 1 0
;To generate forward sequence of 4 pulses
;to move the rotor forward 4 steps
fSequence
```

```
      banksel     PORTD
      movlw       B'110000010'
      movwf       PORTD
      call        delay100ms
      banksel     PORTD
      movlw       B'01100010'
      movwf       PORTD
      call        delay100ms
      banksel     PORTD
      movlw       B'00110010'
      movwf       PORTD
      call        delay100ms
      banksel     PORTD
      movlw       B'10010010'
      movwf       PORTD
      call        delay100ms
      movlw       B'11000000'
      movwf       PORTD
      return

;====subroutine rSequence===
 ;1 REVERSE sequence of 4 pulses (4 step turn) with LED0 ON
;   A  B  ~A  ~B      7 6 5 4 3 2 1 1 (PORTD)
;   +  +   -   -  --->1 1 0 0 0 0 0 1
;   +  -   -   +  --->1 0 0 1 0 0 0 1
;   -  +   -   +  --->0 1 0 1 0 0 0 1
;   -  +   +   -  --->0 1 1 0 0 0 0 1
;To generate forward sequence of 4 pulses
;to move the rotor backward 4 steps (i.e., 1.8x4 = 7.2 degrees)
rSequence
      banksel     PORTD
      movlw       B'11000001'
      movwf       PORTD
      call        delay100ms
      banksel     PORTD
      movlw       B'10010001'
      movwf       PORTD
      call        delay100ms
      banksel     PORTD
      movlw       B'00110001'
      movwf       PORTD
      call        delay100ms
      banksel     PORTD
      movlw       B'01100001'
      movwf       PORTD
      call        delay100ms
      movlw       B'11000000'
      movwf       PORTD
      return
```

The above sequences turn 1.8x4 = 7.2 degrees. The KH42 stepper motor used in the example has step degree of 1.8° per step. The following program is, as an example, is to turn 72 degrees to the right and, after a second of delay, to turn 14 degrees to the left. As usual subroutines are not included in the code.

```
;upsla7024.asm
;
;This program is to:
;
;Motor Control Chip is ALLEGRO sla7024M (for unipolar control)
;The motor used is a KH42 with 1.8 degree/step
;
```

```
; INA is connected to RD7
; INB is connected to RD6
; IN_A is tied to RD5
; IN_B is tied to RD4
; LED1 is connected to RD1   (FW motion indication)
; LED0 is connected to RD0   (RV motion indication)
;
;
        list P = 16F877

STATUS      EQU   0x03
TRISD       EQU   0x88
PORTD       EQU   0x08



;RAM arEA

      CBLOCK      0x20
            TIMEBLOCK
            Kount120us   ;Delay count (number of instr cycles for delay)
            Kount100us
            Kount1ms
            Kount10ms
            Kount100ms
            Kount1s
            Kount10s
            Kount1m
      ENDC


;
;Bootloader first execute the first 4 addresses
;then jump to the address what the execution directs
;=======================================================
      org         0x0000
      GOTO        START
      org         0x05
;=======================================================
;start of the program from $0004
START


      banksel     TRISD
      movlw       H'00'
      movwf       TRISD       ;All ports are outputs


AGAIN
;
;forward
      call        fSequence         ; 7.2 degrees each
      call        fSequence
      call        fSequence
      call        fSequence
      call        fSequence
      call        fSequence
      call        fSequence
      call        fSequence
      call        fSequence
      call        fSequence         ;total 72 degrees

      call        delay1s

      call        delay1s
```

```
;reverse
      call        rSequence         ;7.2 degrees each
      call        rSequence         ;roughly 14 degrees

      call        delay1s
      call        delay1s           ;2 seconds of delay

      goto        AGAIN             ;repeat
```