High Power DC Motor Driver
(10A Continuous - 15 Peak)

This DC motor driver can drive high current DC motor up to 10A continuous and 15A peak current for 10 seconds (Motor start).

It support both locked-antiphase and sign-magnitude PWM signal as well as using full solid state components which result in faster response time and eliminate the wear and tear of the mechanical relay.

Features

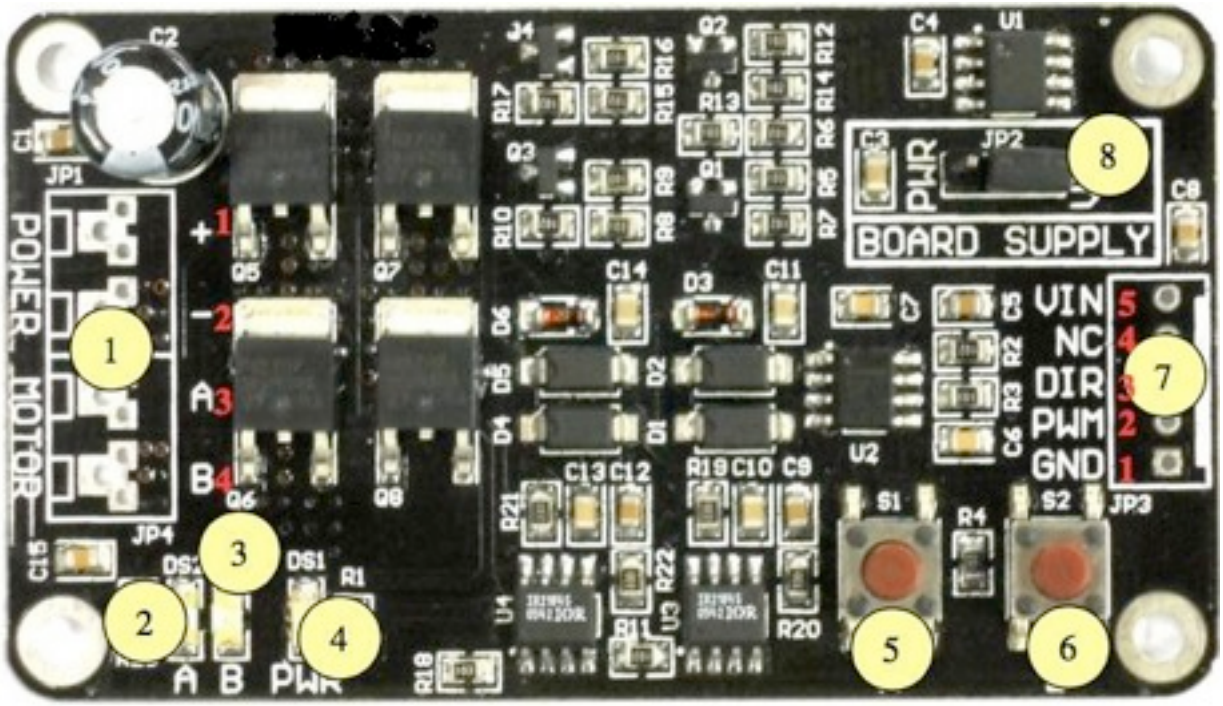
- Bi-directional control for 1 DC motor.
- Support motor voltage ranges from 3V to 25V.
- Maximum current up to 10A continuous and 15A peak (10 second).
- 3.3V and 5V logic level input.
- Solid state components provide faster response time and eliminate the wear and tear of mechanical relay.
- Fully NMOS H-Bridge for better efficiency and no heat sink is required.
- Speed control PWM frequency up to 10KHz.
- Support both locked-antiphase and sign-magnitude PWM operation (check this technical paper for difference between 2 techniques).

PRODUCT SPECIFICATION AND LIMITATIONS

No	Parameters	Min	Typical	Max	Unit
1	Power Input Voltage (Motor Supply Voltage)	3	-	25	V
2	I_{MAX} (Maximum Continuous Motor Current)	-	-	10	A
3	I_{PEAK} – (Peak Motor Current) *	-	-	15	A
4	V_{IN} (Board Supply Voltage)	11	12	14	V
5	V_{IOH} (Logic Input – High Level)	3	-	5.5	V
6	V_{IOL} (Logic Input – Low Level)	0	0	0.5	V
7	Maximum PWM Frequency	-	-	10	KHz

Must not exceed 10 seconds.

BOARD LAYOUT



1. Terminal Block – Connect to motor and power source.

Pin No.	Pin Name	Description
1	POWER +	Positive supply.
2	POWER -	Negative supply.
3	Motor Output A	Connect to motor terminal A.
4	Motor Output B	Connect to motor terminal B.

2. Red LED A – Turns on when the output A is high and output B is low. Indicates the current flows from output A to B.

3. Red LED B – Turns on when the output A is low and output B is high. Indicates the current flows from output B to A.

4. Green LED – Power LED. Should be on when the board is powered on.

5. Test Button A – When this button is pressed, current flows from output A to B and motor will turn CW (or CCW depending on the connection).

6. Test Button B – When this button is pressed, current flows from output B to A and motor will turn CCW (or CW depending on the connection).

7. Input

Pin No.	Pin Name	Description
1	GND	Logic ground.
2	PWM	PWM input for speed control.
3	DIR	Direction control.
4	NC	Not connected. This pin is not used.
5	VIN*	Board power supply.

** This can be left unconnected if the board is powered by motor power input.*

The truth table for the control logic is as follow:

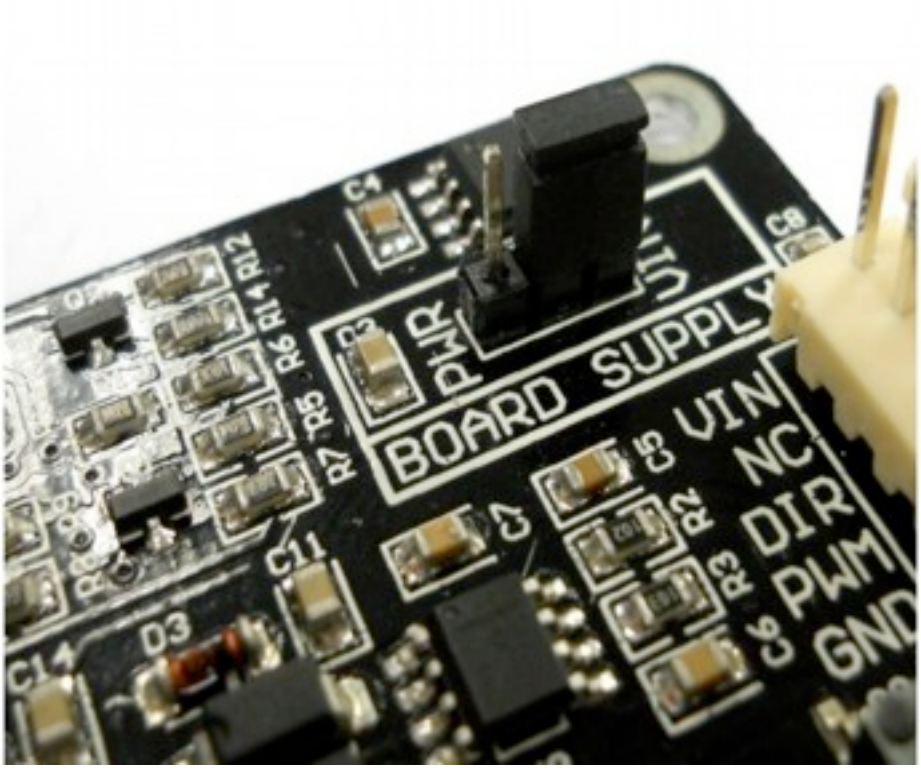
Pin 2 (PWM)	Pin 3 (DIR)	Output A	Output B
Low	X (Don't Care)	Low	Low
High	Low	High	Low
High	High	Low	High

8. Jumper – Board Supply Selector

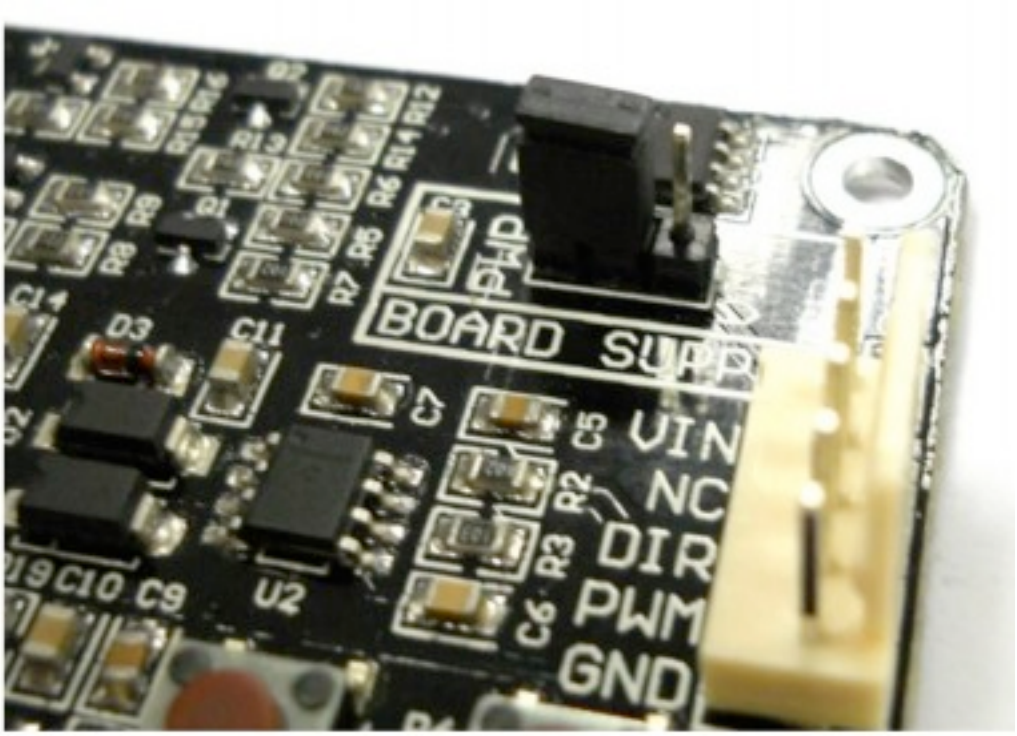
PWR: Board is powered by motor power input. Only select this when the motor
power input is > 14V.
VIN: Board is powered by VIN.12V must be supplied to the VIN pin and motor
power input can be 3V – 25V.

INSTALLATION AND GETTING STARTED

Before using the motor driver, user needs to define the power source for the board by using the onboard jumper. By default, VIN is selected as the board supply and this can be used in all cases. In this mode, 12V must be supplied to the VIN pin at the input port. See photo below



If the Motor driver is used to drive a DC motor from 14V – 25V, the board can be optionally powered by the motor power input. To do so, just select PWR by using the onboard jumper. In this mode, the VIN pin at the input port can be left unconnected. See photo below.



Pulse Width Modulation (PWM):

The DC motor driver is compatible with 2 types of PWM operation, which are:

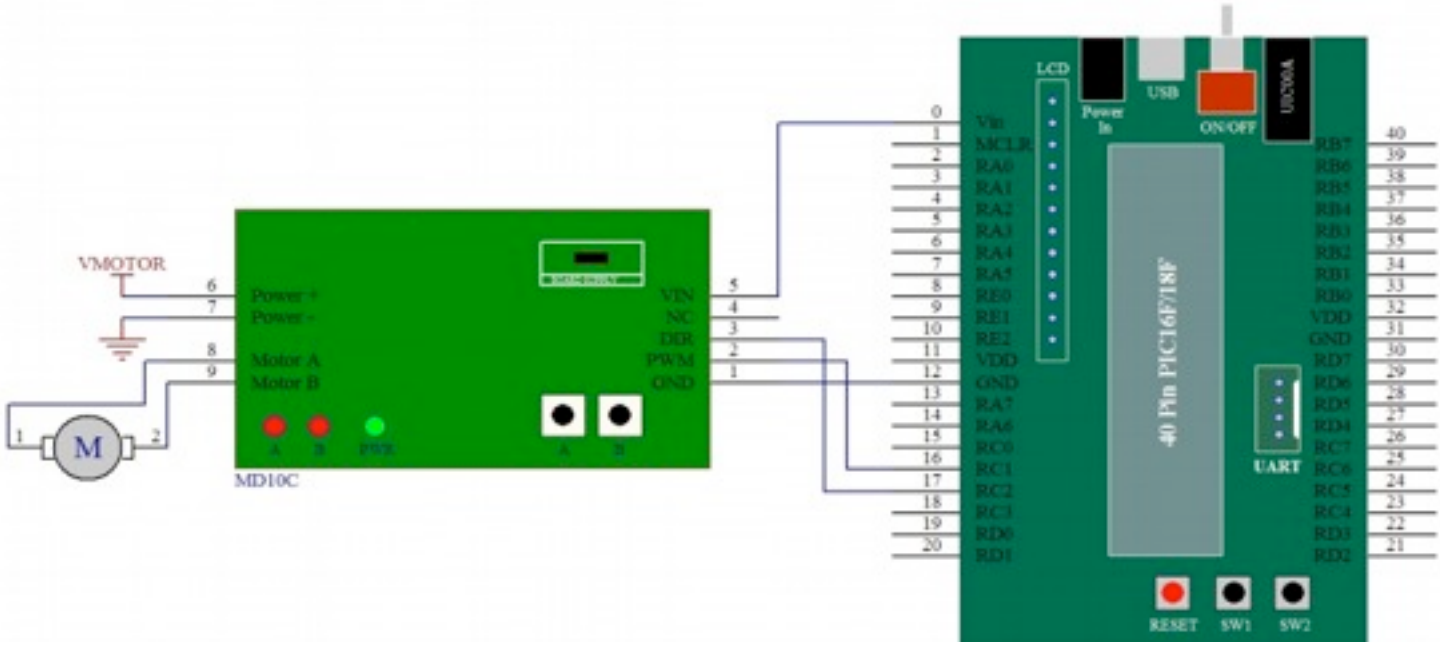
1. **Sign-Magnitude PWM**
For sign-magnitude PWM operation, 2 control signals are used to control the speed and direction of the motor. PWM is feed to the PWM pin to control the speed while DIR pin is used to control the direction of the motor.

2. **Locked-Antiphase PWM**

For locked-antiphase PWM operation, only 1 control signal is needed to control the speed and direction of the motor. PWM pin is connected to logic high while the DIR pin is being feed with the PWM signal. When the PWM signal has 50% duty cycle, the motor stops running. If the PWM has less than 50% duty cycle, the

motor will turn CW (or CCW depending on the connection). If the PWM signal has morethan 50% duty cycle, motor will turn CCW (or CW depending on the connection).

Sample source code for using PIC16F877A to control the motor with motor driver is provided below:



```
//
================================================================
//     Author                    : CRYTON
//     Project                   : 10 AMotor driver Test Program
//     Project description       : Test the functionality of the driver by using
//
//
//
================================================================

//     include
//
================================================================
#include <pic.h>

//configuration
//
================================================================
__CONFIG ( 0x3F32 );          //configuration for the  microcontroller

//     define
//
================================================================
#define  rs             RB4          //RS pin of the LCD display
#define  e              RB5          //E pin of the LCD display

#define  lcd_data PORTD       //LCD 8-bit data PORT

#define  SW1                RB0
#define  SW2                RB1

#define  LED1           RB6
#define  LED2           RB7

#define DIR             RC2
```

```
#define PWM                 RC1

//    function prototype       (every function must have a function prototype)
//
================================================================
================
void delay(unsigned long data);
void send_config(unsigned char data);
void send_char(unsigned char data);
void lcd_goto(unsigned char data);
void lcd_clr(void);
void send_string(const char *s);
void uart_send(unsigned char data);



//    global variable
//
================================================================
================

//    main function                    (main fucntion of the program)
//
================================================================
================
void main()
{
      unsigned long delay_time=5000;

      //set I/O input output
      TRISB = 0b00000011;              //configure PORTB I/O direction
      TRISD = 0b00000000;              //configure PORTD I/O direction
      TRISA = 0b00000111;              //configure PORTA I/O direction

      // Set RC1 and RC2 as output pin.
      TRISC1 = 0;
      TRISC2 = 0;

      LED1=0;                                  //OFF LED1
      LED2=0;                                  //OFF LED2
```

```
	PWM = 0;                                        // Stop motor.
	DIR = 0;                                    // Direction = 0;

	//setup ADC
	ADCON1 = 0b00000110;                        //set ADx pin digital I/O

	// Setting PWM frequency = 4.88khz.
	PR2 = 0xff;
	T2CKPS1 = 0;
	T2CKPS0 = 1;      // Timer 2 prescale = 4.
	TMR2ON = 1;             // Turn on Timer 2.



	//configure lcd
	send_config(0b00000001);            //clear display at lcd
	send_config(0b00000010);            //lcd return to home
	send_config(0b00000110);            //entry mode-cursor increase 1
	send_config(0b00001100);            //display on, cursor off and cursor
blink off
	send_config(0b00111000);            //function set



	while(1)
	{
		PWM = 0;                                        // Stop motor.
		DIR = 0;                                    // Direction = 0;

		// CCP1 as normal IO.
		CCP1M3 = 0;
		CCP1M2 = 0;

		// CCP2 as normal IO.
		CCP2M3 = 0;
		CCP2M2 = 0;
```

```
// Display the message.
lcd_clr();
send_string("MD10C TESTER");
lcd_goto(0x10);
send_string("Press SW1 to run");

// Waiting for user to press SW1.
while (SW1 == 1);



// Sign-Magnitude Mode.
// PWM pin determines the speed.
// DIR pin determines the direction.

// CCP1 as normal IO.
CCP1M3 = 0;
CCP1M2 = 0;

// Setup CCP2 as PWM.
CCP2M3 = 1;
CCP2M2 = 1;
CCPR2L = 0x0;    // Stop the motor.

// Display the message.
lcd_clr();
send_string("Sign-Magnitude");
lcd_goto(0x10);
send_string("DIR: 0");

// Run the motor.
DIR = 0;
LED1 = 1;
LED2 = 0;

while (++CCPR2L != 0xff)
{
      delay(2000);
}
```

```
while (--CCPR2L != 0x0)
{
      delay(2000);
}

LED1 = 0;
LED2 = 0;



// Display the message.
lcd_goto(0x10);
send_string("DIR: 1");

// Run the motor.
DIR = 1;
LED1 = 0;
LED2 = 1;

while (++CCPR2L != 0xff)
{
      delay(2000);
}

while (--CCPR2L != 0x0)
{
      delay(2000);
}

LED1 = 0;
LED2 = 0;



// Locked Antiphase Mode.
// PWM pin is always high.
// DIR pin determines the speed and direction.
// 50% duty cycle on DIR = Stop.
// Less than 50% duty cycle on DIR = Run with Direction 0.
// More than 50% duty cycle on DIR = Run with Direction 1.
```

```
// CCP1 as PWM.
CCP1M3 = 1;
CCP1M2 = 1;
CCPR1L = 0x80;  // Stop the motor.

// CCP2 as normal IO.
CCP2M3 = 0;
CCP2M2 = 0;

// PWM pin should always high.
PWM = 1;

// Display the message.
lcd_clr();
send_string("Locked Antiphase");
lcd_goto(0x10);
send_string("DIR: 0");

// Run the motor.
LED1 = 1;
LED2 = 0;

while (--CCPR1L != 0x0)
{
      delay(4000);
}

while (++CCPR1L != 0x80)
{
      delay(4000);
}

LED1 = 0;
LED2 = 0;



// Display the message.
lcd_goto(0x10);
```

```
            send_string("DIR: 1");

            // Run the motor.
            LED1 = 0;
            LED2 = 1;

            while (++CCPR1L != 0xff)
            {
                  delay(4000);
            }

            while (--CCPR1L != 0x80)
            {
                  delay(4000);
            }

            LED1 = 0;
            LED2 = 0;
      }
}

//    functions
//
================================================================
================
void delay(unsigned long data)          //delay function, the delay time
{                                             //depend on the given value
      for( ;data>0;data--);
}

void send_config(unsigned char data)    //send lcd configuration
{
      rs=0;                                   //set lcd to configuration mode
      lcd_data=data;                          //lcd data port = data
      e=1;                                    //pulse e to confirm the data
      delay(50);
      e=0;
      delay(50);
}
```

```
void send_char(unsigned char data)      //send lcd character
{
      rs=1;                                   //set lcd to display mode
      lcd_data=data;                          //lcd data port = data
      e=1;                              //pulse e to confirm the data
      delay(10);
      e=0;
      delay(10);
}

void lcd_goto(unsigned char data)       //set the location of the lcd cursor
{                                             //if the given value is (0-15)
the
      if(data<16)                             //cursor will be at the upper
line
      {                                       //if the given value is (20-35)
the
            send_config(0x80+data);           //cursor will be at the lower
line
      }                                       //location of the lcd cursor
(2X16):
      else                              //
-------------------------------------------------
      {                                       // | |00|01|02|03|04|05|06|07|
08|09|10|11|12|13|14|15| |
            data=data-20;                     // | |20|21|22|23|24|25|26|27|
28|29|30|31|32|33|34|35| |
            send_config(0xc0+data);           //
-------------------------------------------------
      }
}

void lcd_clr(void)                      //clear the lcd
{
      send_config(0x01);
      delay(600);
}

void send_string(const char *s)         //send a string to display in the lcd
{
```

```
    while (s && *s)send_char (*s++);
}

void uart_send(unsigned char data)
{
    while(TXIF==0);                         //only send the new data after
    TXREG=data;                                 //the previous data finish
sent
}
```