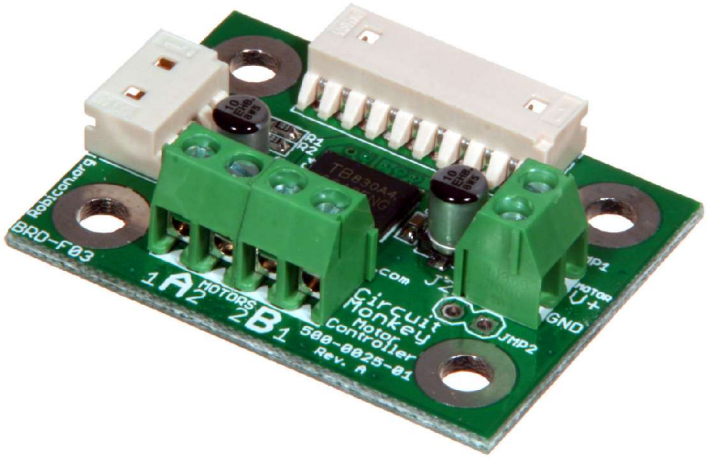
Circuit Monkey

500-0025-01

Motor Controller

TB6612FNG

User Guide



Version: 1.0

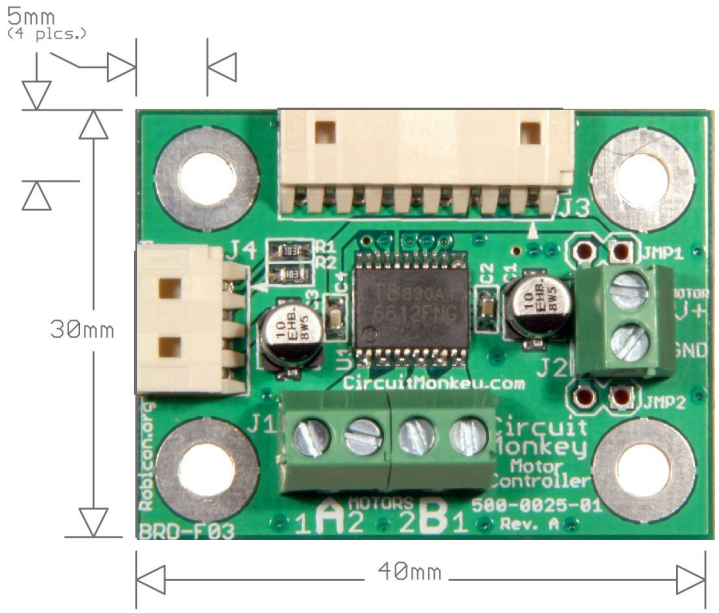
2009/08/24

Illustrations and photographs by Mark J Koch
This document was created with OpenOffice 3.0.

Table of Contents

Overview

Circuit Monkey [http://circuitmonkey.com] presents part number ***500-0025-01***, a versatile ***dual-channel motor controller*** based on the *Toshiba TB6612FNG* chip. This controller is designed to allow easy and reliable hookup of small hobby-type motors for experimenters, hobbyists and engineers. The control signals are connected using *Molex 53015 (MicroBlade)* series for reliable and vibration resistant connection. The motor signals are connected using industry standard terminal blocks and accept a wide range of wire gauges (24-18AWG). The board is 30mm x 40mm and features four mounting holes placed on a 20mm x 30mm mounting grid. The holes are plated and connected to logic ground and accept screws of size M3 or less *(M3, 6-32, M2.5, 4-40, etc.)*.

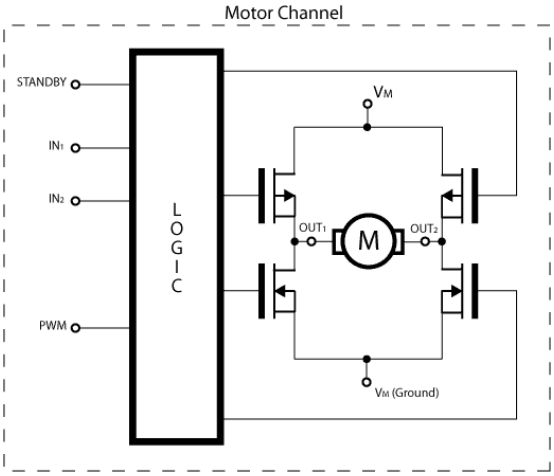


Features

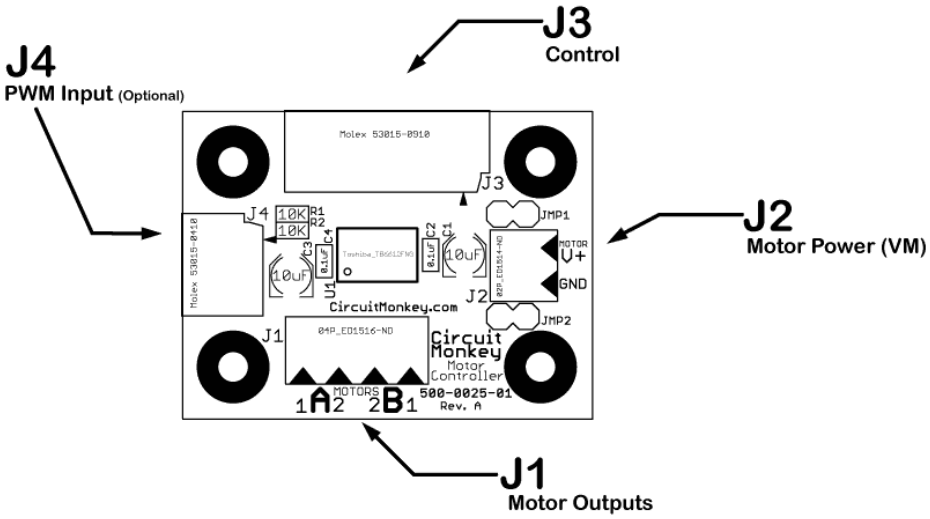
- Controller Power supply voltage; V_{CC} = 5V
- Motor Power supply voltage; V_M = $2.5V_{(min)}$ - $15V_{(max)}$
- Output current; I_{OUT}=1.0A(avg) / 3.0 A (peak)
- Output low ON resistance; 0.5Ω (upper+lower Typ. @$V_M \geq$ 5 V)
- Standby (Power save) mode
- CW / CCW / short brake / stop function modes
- Built-in thermal shutdown circuit and low voltage detecting circuit
- Easy to attach terminal blocks for motor power (V_M) and motor wires
- RobiCon compliant connector for control and logic power.
- RobiCon compliant connector for optional PWM control (internal pull-ups when not used).
- RobiCon compliant board outline and mounting.
- 40mm x 30mm x 12.1mm (WxDxH)
- 1.6"W x 1.2"D x 0.48"H
- 7.94g (0.28oz)

Block Diagram

A block diagram representing a single channel in the *TB6612FNG* chip is shown below. IN_1 and IN_2 control the four output states of the motor controller. The PWM pin is optional and allows for varying the speed of the motor by sending a pulsed signal to this pin. The Standby pin must be asserted high or low depending on the desired operating state. Asserting the pin low (0V) will turn off power to the chip, thereby saving power when not in use. This pin must be asserted to logic level high (+5V) to allow the chip to operate the motors. The Motor attaches to OUT_1 and OUT_2. Motor Power is supplied via the V_M and V_M(GROUND) pins. Note that the ground (return path) for the motor voltage is isolated from the logic ground.



Connectors

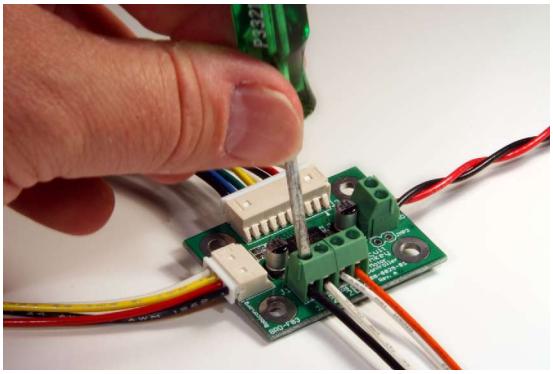


J1 – Motor Outputs

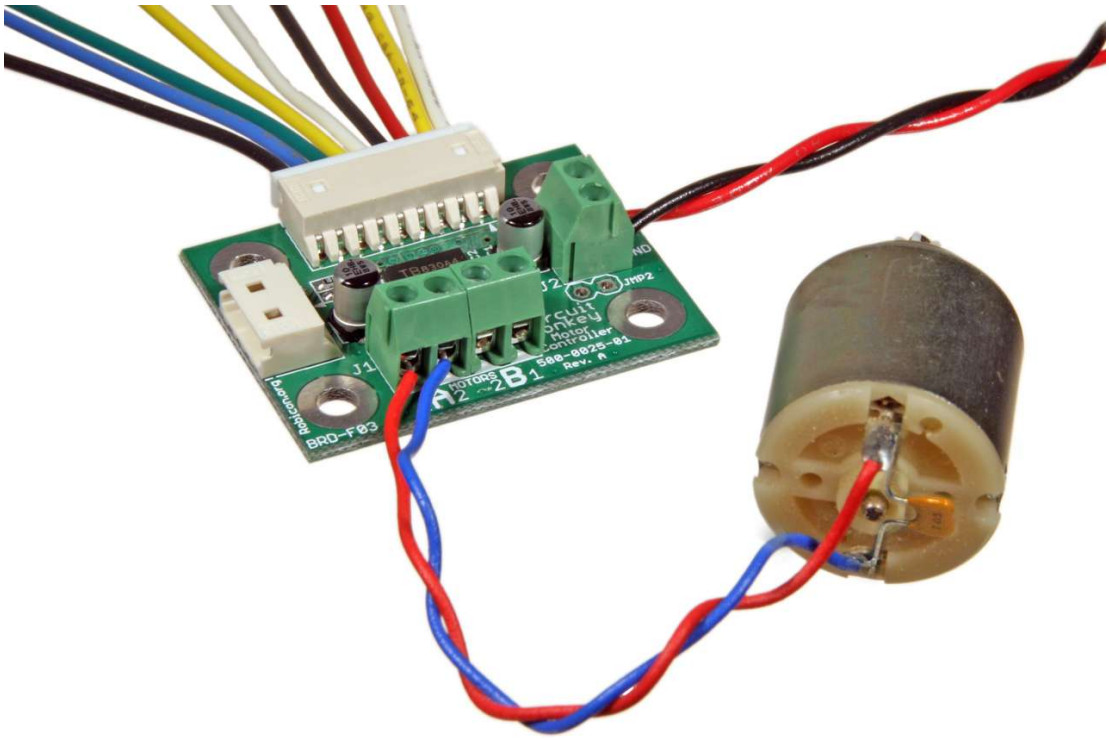
This terminal block features four contacts, two for each motor channel, A and B. For most applications, hook pin 1 to the positive lead of the motor and pin 2 to the negative lead of the motor. If you notice the motor running in reverse then swap the wires.

Pin	Description
A1	Motor A Positive
A2	Motor A Negative
B2	Motor B Negative
B1	Motor B Positive

Using a small screwdriver, turn the terminal block screw counter-clockwise to loosen that contact's wire clamp mechanism. Prepare the wire by stripping approximately 1/4-inch (6mm) of insulation from the end. Tinning the end of the wire with solder is recommended and will help with handling and fraying, but is not required. Insert wire into the terminal block contact hole. Secure the wire by turning clockwise on that terminal's clamp screw until the wire is solidly captured in the clamp system. Test the connection by tugging on the wire. A good connection should not let the wire pull or slide out.



A typical motor hook-up is shown here:



Tip:

Add a **0.1uF capacitor** across the input leads of the motor (as close as possible to the motor) to help prevent motor induced electrical noise in the system. Failure to do so *could* cause erratic behavior in the controller or your microcontroller system.

J2 – Motor Power

This terminal block features two contacts, *motor power* and *motor ground*. The motor power input voltage should match the voltage rating for your motor. For example, a 3V motor should have a 3V input source. Using excessive motor power input voltages for motors that are not rated for that voltage range may damage your motor and motor controller.

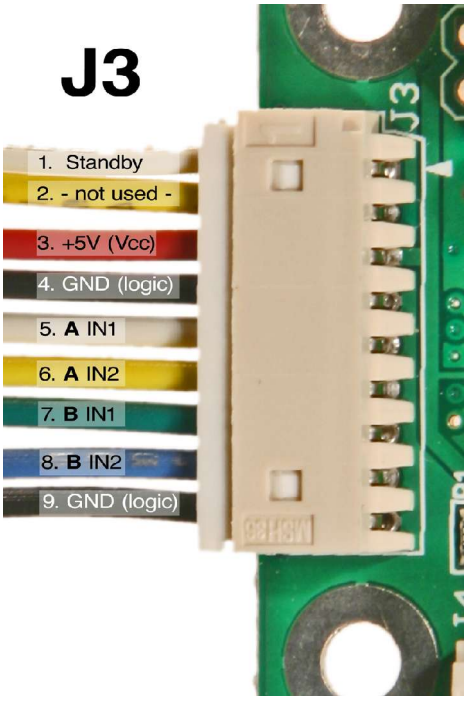
Pin	Description
VM	Motor Power
VM GND	Motor Power Ground

The motor input voltage (and ground) is *separate* from the motor controller's *logic* voltage supply. Do not use the terminal block ground (GND) connection as a reference for any logic signal measurements. It is not recommended that users tie these two sources together (i.e. powering the motors with 5V Vcc from the microcontroller) ***unless*** they understand the implications of doing so. **Damage to the motors and/or motor controller may result from improper configuration of these voltages.**

J3 – Control

This connector is a *Molex 53015-0910* and is compatible with the the *RobiCon.org* system *(a proposed open standard for robotics interconnect)*. The connector supplies the ***logic power*** and ***logic ground*** as well as the ***Standby, A1, A2, B1,*** and ***B2*** signals to control the motor chip. All signal levels are 5V TTL. Read the *Toshiba TB6612FNG* chip specification for how to properly use these signals.

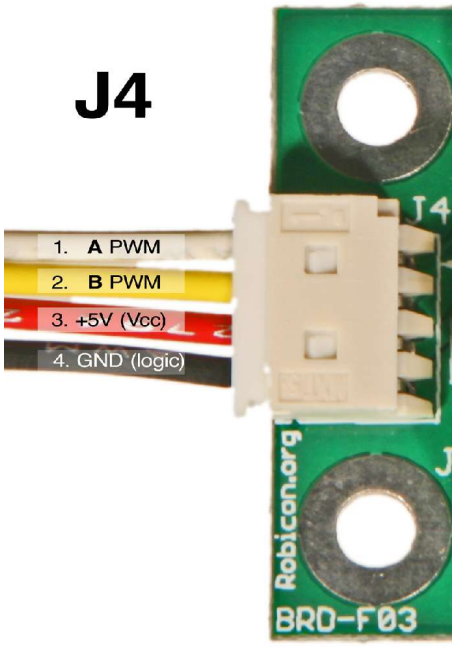
Pin	Signal	Description
1	Standby	Power Save: Low Active. Keep high (+5V) for normal use.
2	*- not used -*	*Unused. Not connected.*
3	**Vcc**	+5V DC logic supply
4	**Ground**	Logic Ground
5	A1	Channel A – Input 1
6	A2	Channel A – Input 2
7	B1	Channel B – Input 1
8	B2	Channel B – Input 2
9	**Ground**	Logic Ground



J4 – PWM Input (optional)

This connector is a *Molex 53015-0410* and allows for advanced control of motor speed by using a *Pulse Width Modulated (PWM)* signal from a microcontroller or other source. The maximum switching frequency for the PWM pin is 100KHz. Pulse widths (high or low) should generally be greater than 20uS. Read the *Toshiba TB6612FNG* chip specification for proper use of these signals. These signals are internally pulled to logic level high (+5V) when not connected. This equates to a 100% on duty cycle state.

Pin	Signal	Description
1	PWM A	PWM Input for Channel A
2	PWM B	PWM Input for Channel B
3	**Vcc**	+5V DC logic supply (not always needed but is attached to board's +5V Vcc logic supply).
4	**Ground**	Logic Ground



Applications

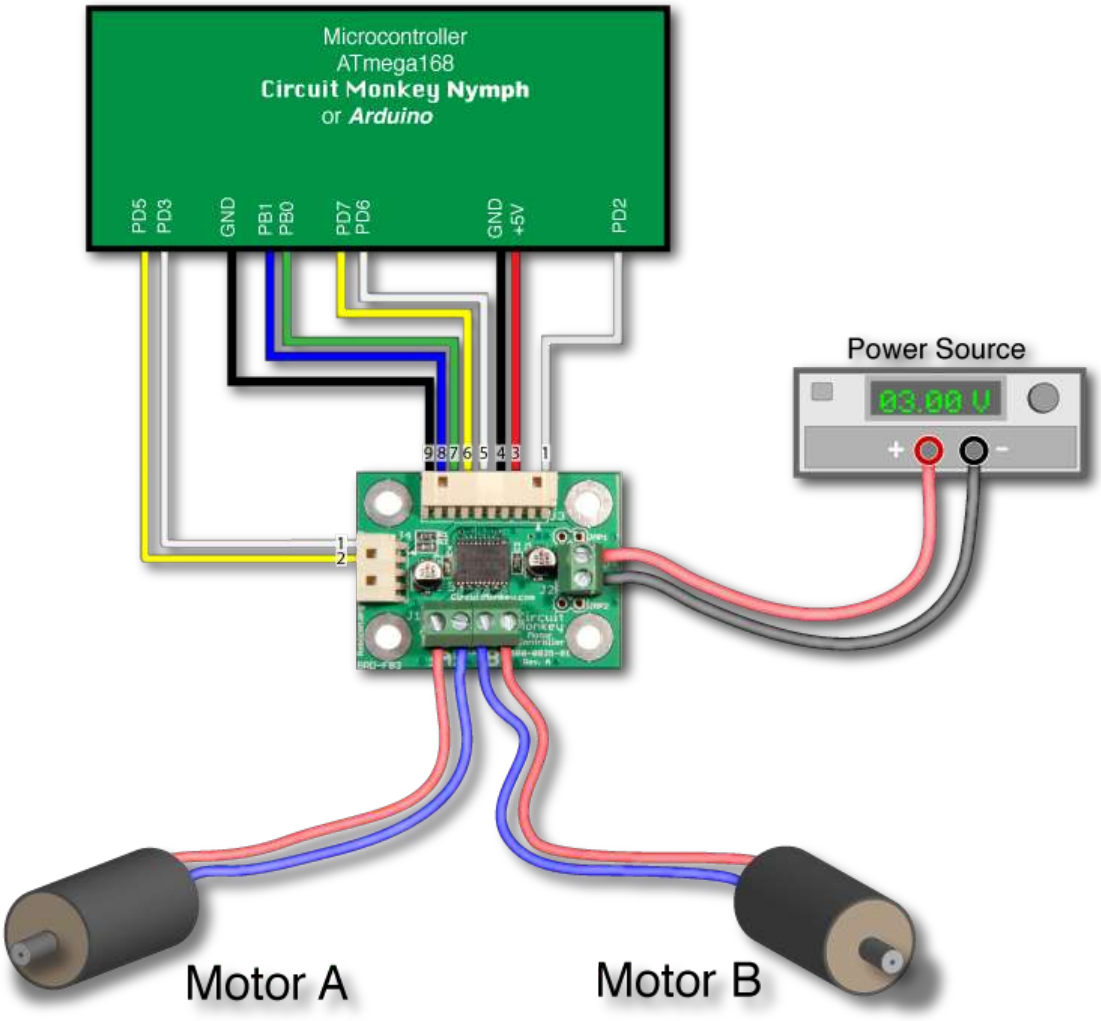
Prevent Chip Damage! – Use Dead-Time

Refer to page 4 of the *Toshiba TB6612FNG* chip specification. When switching between active motor states, it is highly recommended that the user briefly switch the motor controller to OFF mode. The dead time allows the H-bridge transistors to turn off completely before others turn on. Some transient situations would encounter a state where both transistors in the H-bridge are conducting, creating a V_M to V_M(Ground) short. Such a state could cause permanent damage to the chip.

Setting the motor inputs or Standby input low for a minimum of 250nS (or 0.00025mS, an extremely short period of time in the microcontroller world). In a programming situation, simply setting the OFF state and moving to the next state should be more than adequate. We attempt to demonstrate this in our test code.

Test Configuration

A simple test configuration diagram is shown below. Actual test code follows.



Test Code (for Atmega168 – Nymph and Arduino Compatible)

Using the preceding hookup diagram, one can test the functionality of the motor controller with the following code. It was designed to run on a *ATmega168/328* board. We tested with our own *Nymph* product but this code should also run on an *Arduino/Freeduino* or similar board, as they are also *Atmel AVR* based.

```
/**
 **
  Circuit Monkey
  500-0025-01  -- Motor Controller Test/Demo

  Designed for Nymph (ATmega328) Microcontroller

  Author:  Mark J Koch

================================================================================
License:   BSD

Copyright (c) 2009, Circuit Monkey
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

    * Redistributions of source code must retain the above copyright notice,
      this list of conditions and the following disclaimer.
    * Redistributions  in binary form   must reproduce the  above copyright
      notice,  this list of conditions and  the following disclaimer in the
      documentation and/or other materials provided with the distribution.
    * Neither the name of Circuit Monkey  nor the names of its contributors
      may be used to endorse or promote products derived from this software
      without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING,  BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
LIABLE FOR ANY   DIRECT,  INDIRECT,  INCIDENTAL,  SPECIAL,  EXEMPLARY,  OR
CONSEQUENTIAL  DAMAGES  (INCLUDING,  BUT  NOT  LIMITED TO,  PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES;  LOSS OF USE, DATA, OR PROFITS;  OR BUSINESS
INTERRUPTION)  HOWEVER CAUSED  AND ON ANY THEORY OF LIABILITY,  WHETHER IN
CONTRACT,  STRICT  LIABILITY, OR TORT  (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
================================================================================

  Misc Useful Links:
   Excellent PWM Tutorial:
```

```
        http://www.8051projects.net/pulse-width-modulation/avr-pwm-example.php

 **
 **
 **/
#include <avr/io.h>
#define F_CPU 16000000UL  // 16MHz
#include <util/delay.h>

// Misc Defines
#define A       0
#define B       1
#define OFF     0
#define ON      1
#define BKWD    0
#define FWD     1

/*
 * Port and Pin definitions
 * Based on Circuit Monkey Nymph (PN: 500-0011-03)  microcontroller.
 * These settings should be compatible with Arduino board pinouts.
 */
// Define LED Pin on Nymph (D13 = PB5)
#define LEDOUT          PORTB5
#define LEDPORT         PORTB
#define LEDDDR          DDRB
#define LEDDDRPIN       DDB5

// Define Motor Standby Pin on Nymph (D2 = PD2)
#define MOT_STBY_PORT   PORTD
#define MOT_STBY_OUT    PORTD2
#define MOT_STBY_DDR    DDRD
#define MOT_STBY_DDRPIN DDD2

// Define Motor A Pins on Nymph (D6 = PD6, D7 = PD7)
#define MOT_A_PORT      PORTD
#define MOT_A_OUT_1     PORTD6
#define MOT_A_OUT_2     PORTD7
#define MOT_A_DDR       DDRD
#define MOT_A_DDRPIN_1  DDD6
#define MOT_A_DDRPIN_2  DDD7

// Define Motor B Pins on Nymph (D8 = PB0, D9 = PB1)
#define MOT_B_PORT      PORTB
#define MOT_B_OUT_1     PORTB0
#define MOT_B_OUT_2     PORTB1
#define MOT_B_DDR       DDRB
#define MOT_B_DDRPIN_1  DDB0
#define MOT_B_DDRPIN_2  DDB1
```

```
// Define PWM Pins on Nymph (D3 = PD3, D5 = PD5)
// Note these pins are PWM (Output Compare Timer) Pins
// We use the internal counter/timer feature of the microcontroller.
#define MOT_PWM_PORT        PORTD
#define MOT_A_PWM_OUT       PORTD3
#define MOT_B_PWM_OUT       PORTD5
#define MOT_PWM_DDR         DDRD
#define MOT_A_PWM_DDRPIN    DDD3
#define MOT_B_PWM_DDRPIN    DDD5

#define MOT_A_PWM_OCR       OCR2B
#define MOT_B_PWM_OCR       OCR0B

#define MOT_A_PWM_TCCRA     TCCR2A
#define MOT_A_PWM_TCCRB     TCCR2B
#define MOT_B_PWM_TCCRA     TCCR0A
#define MOT_B_PWM_TCCRB     TCCR0B
#define MOT_A_PWM_TCCRA_MASK 0x31 // Mask/Value will change depending on what
#define MOT_A_PWM_TCCRA_MODE 0x21 // PWM (Output Compare Pin) you use.
#define MOT_B_PWM_TCCRA_MASK 0x31
#define MOT_B_PWM_TCCRA_MODE 0x21

// PWM duty cycles for 8-bit timer register
#define PW_0    0x00
#define PW_25   0x40
#define PW_50   0x80
#define PW_75   0xC0
#define PW_100  0xFF


// delay milliseconds
// Facade for _delay_ms
//  Helps reduce binary size since _delay_ms needs a float value
//  and we want to use an int value.  Calling the float version
//  all over your code would bloat your hex binary.
void delayMs(unsigned int ms)
{
    while(ms){
        _delay_ms(0.96);
        ms--;
    }
}

void initLED( void ) {
    /* enable pin as output */
    LEDDDR|= (1<<LEDDDRPIN);
}
```

```
void initMotor(int mot) {
    switch( mot ) {
        case A:
            /* enable pin as output */
            MOT_A_DDR|= (1<<MOT_A_DDRPIN_1);
            MOT_A_DDR|= (1<<MOT_A_DDRPIN_2);
            break;
        case B:
            /* enable pin as output */
            MOT_B_DDR|= (1<<MOT_B_DDRPIN_1);
            MOT_B_DDR|= (1<<MOT_B_DDRPIN_2);
            break;
    }
}

void initPWM(int mot) {
    switch( mot ) {
        case A:
            MOT_A_PWM_OCR = PW_50; // Set initial Pulse Width
            MOT_PWM_DDR  |= (1<<MOT_A_PWM_DDRPIN); // enable pin as output
            MOT_PWM_PORT |= (1<<MOT_A_PWM_OUT);    // Set high.
            // 8-bit non-inverted
            MOT_A_PWM_TCCRA &= ~MOT_A_PWM_TCCRA_MASK; // Clear the TCCR bits.
            MOT_A_PWM_TCCRA |= MOT_A_PWM_TCCRA_MODE; //Clear-On-Match mode.
            break;
        case B:
            // Set initial Pulse Width
            MOT_B_PWM_OCR = PW_50;
            /* enable pin as output */
            MOT_PWM_DDR  |= (1<<MOT_B_PWM_DDRPIN);
            MOT_PWM_PORT |= (1<<MOT_B_PWM_OUT); // Set high.
            // 8-bit non-inverted
            MOT_B_PWM_TCCRA &= ~MOT_B_PWM_TCCRA_MASK; // Clear the TCCR bits.
            MOT_B_PWM_TCCRA |= MOT_B_PWM_TCCRA_MODE; //Clear-On-Match mode.
            break;
    }
}

void setPWM(int mot, unsigned int pw ) {
    if ( pw > PW_100 ) pw = PW_100; // Clamp high (TOP) value
    switch(mot) {
        case A:
            MOT_A_PWM_OCR = pw;
            break;
        case B:
            MOT_B_PWM_OCR = pw;
            break;
    }
}
```

```
void enablePWM(int mot) {
    switch(mot) {
        case A:
            MOT_A_PWM_TCCRB = 1; // Start the PWM
            break;
        case B:
            MOT_B_PWM_TCCRB = 1; // Start the PWM
            break;
    }
}

void disablePWM(int mot) {
    switch(mot) {
        case A:
            MOT_A_PWM_TCCRB = 0; // Stop the PWM
            break;
        case B:
            MOT_B_PWM_TCCRB = 0; // Stop the PWM
            break;
    }
}

void setStandby(int mode) {
    switch (mode) {
        case OFF:
            MOT_STBY_PORT |= (1<<MOT_STBY_OUT);   //Pin High
            break;
        case ON:
            MOT_STBY_PORT &= ~(1<<MOT_STBY_OUT);  //Pin Low
            break;
    }
}

void initStandby(void) {
    /* enable pin as output */
    MOT_STBY_DDR|= (1<<MOT_STBY_DDRPIN);
    setStandby( OFF ); // OFF mode turns chip on.
}

void motorForward(int mot) {
    switch( mot ) {
        case A:
            MOT_A_PORT |=  (1<<MOT_A_OUT_1);
            MOT_A_PORT &= ~(1<<MOT_A_OUT_2);
            break;
        case B:
            MOT_B_PORT |=  (1<<MOT_B_OUT_1);
            MOT_B_PORT &= ~(1<<MOT_B_OUT_2);
```

```
            break;
    }
}

void motorBackward(int mot) {
    switch( mot ) {
        case A:
            MOT_A_PORT &= ~(1<<MOT_A_OUT_1);
            MOT_A_PORT |=  (1<<MOT_A_OUT_2);
            break;
        case B:
            MOT_B_PORT &= ~(1<<MOT_B_OUT_1);
            MOT_B_PORT |=  (1<<MOT_B_OUT_2);
            break;
    }
}

void motorStop(int mot) {
    switch( mot ) {
        case A:
            MOT_A_PORT &= ~(1<<MOT_A_OUT_1);
            MOT_A_PORT &= ~(1<<MOT_A_OUT_2);
            break;
        case B:
            MOT_B_PORT &= ~(1<<MOT_B_OUT_1);
            MOT_B_PORT &= ~(1<<MOT_B_OUT_2);
            break;
    }
}

void motorBrake (int mot) {
    switch( mot ) {
        case A:
            MOT_A_PORT |=  (1<<MOT_A_OUT_1);
            MOT_A_PORT |=  (1<<MOT_A_OUT_2);
            break;
        case B:
            MOT_B_PORT |=  (1<<MOT_B_OUT_1);
            MOT_B_PORT |=  (1<<MOT_B_OUT_2);
            break;
    }
}

void setLED( int mode ) {
    switch(mode) {
        case ON:
            LEDPORT|= (1<<LEDOUT);    // Set output to 5V, LED on
            break;
        case OFF:
```

```
            LEDPORT &= ~(1<<LEDOUT); // set output to 0V, LED off
            break;
    }
}

void testMotor(int mot, int dir, unsigned int pw) {
    setLED(ON);
    setPWM(mot, pw); // Set up PW for motor
    enablePWM(mot); // Enable PWM mode for motor
    switch (dir) {
        case BKWD:   // Run backward
            motorBackward(mot);
            break;
        case FWD: // Run forward
            motorForward(mot); // Run the motor
            break;
    }
    delayMs(2000);  // Run for 2 seconds

    setLED(OFF);
    disablePWM(mot); // Clear the PWM mode
    motorStop(mot); // Stop motor
    motorBrake(mot);
    delayMs(500);
    motorStop(mot);
}

int main(void)
{
    initLED();
    initStandby();
    initMotor(A);
    initMotor(B);
    initPWM(A);
    initPWM(B);

    while (1) {
        testMotor(A, BKWD, PW_25);
        testMotor(A, BKWD, PW_50);
        testMotor(A, BKWD, PW_75);
        testMotor(A, BKWD, PW_100);

        testMotor(A, FWD, PW_25);
        testMotor(A, FWD, PW_50);
        testMotor(A, FWD, PW_75);
        testMotor(A, FWD, PW_100);

        testMotor(B, BKWD, PW_25);
        testMotor(B, BKWD, PW_50);
```

```
          testMotor(B, BKWD, PW_75);
          testMotor(B, BKWD, PW_100);

          testMotor(B, FWD, PW_25);
          testMotor(B, FWD, PW_50);
          testMotor(B, FWD, PW_75);
          testMotor(B, FWD, PW_100);
          }
      // Never gets here.
          return(0);
  }
```

References and Links

Circuit Monkey

http://www.circuitmonkey.com

Toshiba TB6612FNG Datasheet

http://www.semicon.toshiba.co.jp/openb2b/websearch/productDetails.jsp?partKey=TB6612FNG

DigiKey.com TB6612FNG Component Detail and Pricing

http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=TB6612FNGCT-ND

RobiCon.org

A proposed open standard for Robotics Interconnect. Currently lead by the owner of Circuit Monkey.

http://www.robicon.org

Atmel

Creators of the *AVR/ATmega* microcontrollers.

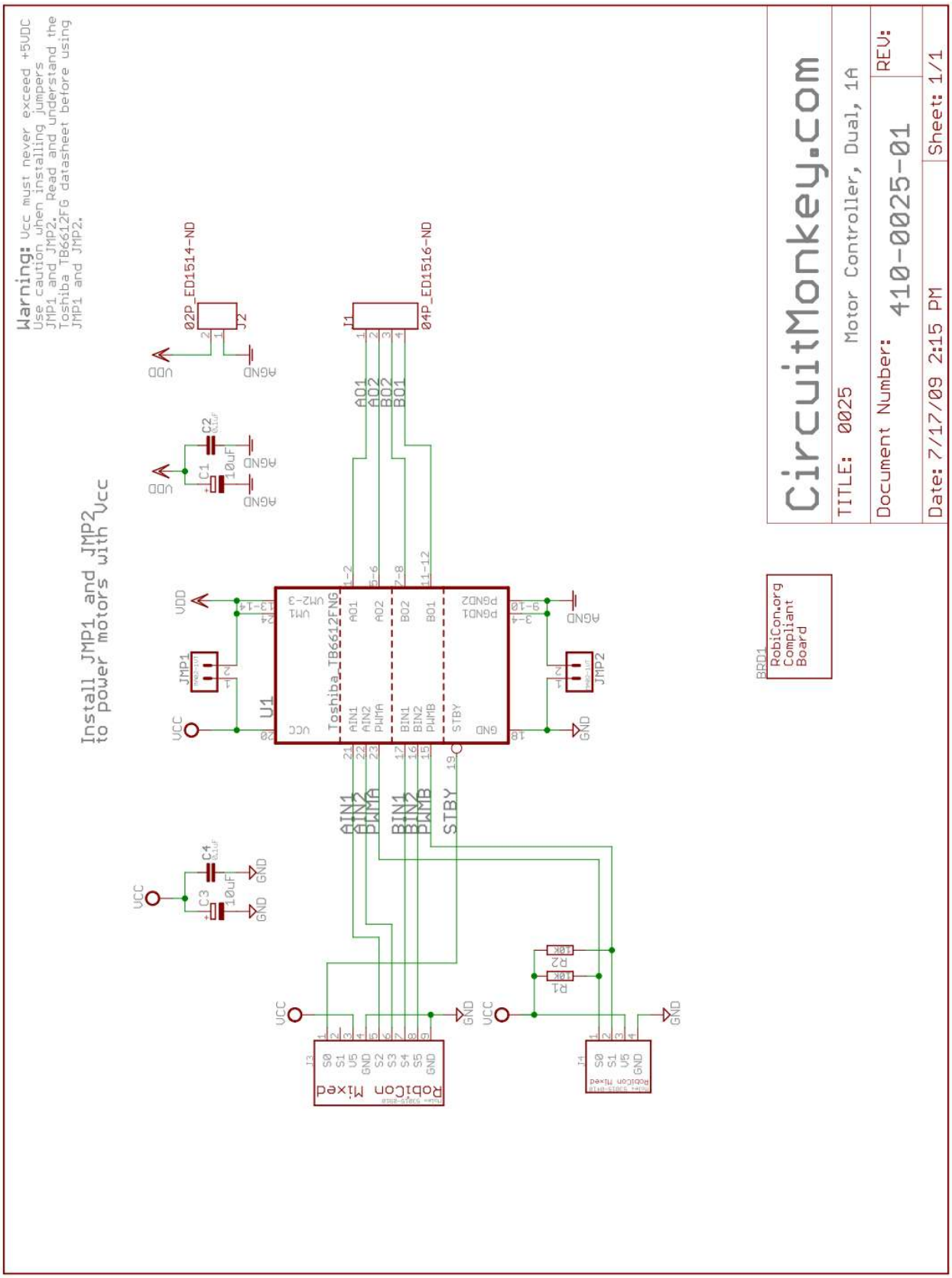
http://atmel.com/products/AVR/

Ricky's World of Microcontrollers and Microprocessors

Great 8051/AVR PWM tutorial and more!

http://www.8051projects.net/pulse-width-modulation/avr-pwm-example.php

Schematic


Warning: Vcc must never exceed +5VDC
Use caution when installing jumpers
JMP1 and JMP2. Read and understand the
Toshiba TB6612FG datasheet before using
JMP1 and JMP2.
Install JMP1 and JMP2
to power motors with Vcc
02P_ED1514-ND
04P_ED1516-ND
Toshiba_TB6612FNG
AIN1
AIN2
PWMA
BIN1
BIN2
PWMB
STBY
Robicon Mixed
RobiCon.org
Compliant
Board
CircuitMonkey.com
TITLE: 0025 Motor Controller, Dual, 1A
Document Number: 410-0025-01
REV:
Date: 7/17/09 2:15 PM
Sheet: 1/1