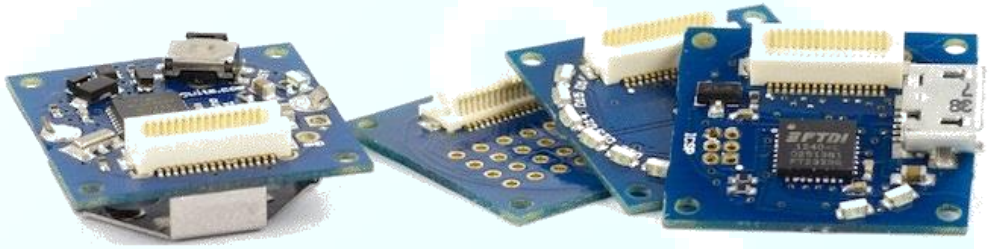
TINYCIRCUITS



DUAL MOTOR TINYSHIELD REFERENCE MANUAL

B_FARMER
AKROSENSE LLC. | CANAL PLACE, SUITE #457 - MAIN ST. - AKRON, OH

Introduction to the Dual Motor TinyShield (ASD2302-R)

DESCRIPTION

The Dual Motor TinyShield allows you to drive two independently controlled DC brushed motors from your TinyDuino. It includes two motor drivers and a microcontroller to control them. This microcontroller allows for I^2C control of its 16 bit PWM (Pulse Width Modulation) outputs. Up to 4 of these TinyShields may be stacked together, allowing control of up to 8 DC motors.

In the default configuration, the Dual Motor TinyShield must have a battery connected to either the processor board or the motor driver board, (which are both connected to the VBATT line on the TinyShield expansion connector). To use a higher voltage source for the motors, resistor R3 must be removed from the TinyShield and the TinyDuino must be powered separately.

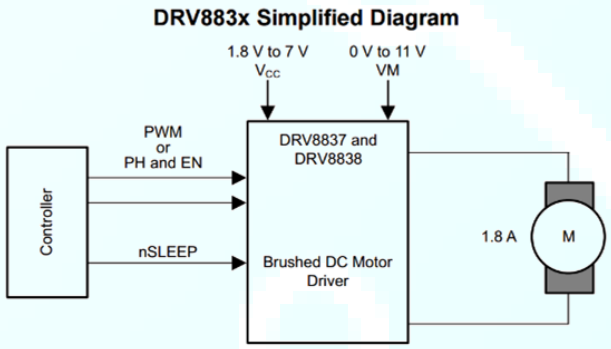


Image Courtesy of: TI DRV883x Datasheet

TECHNICAL DETAILS

TI DRV8837 Motor Driver

- Low MOSFET On-Resistance: HS + LS 280mOhm
- Short Circuit, Overcurrent, Overtemperature protection built in
- 1.8A Max Drive Current (Recommend 500mA max without heatsinking)
- 1.8V to 11V Motor Operating Supply Voltage Range

ATtiny841

- 8 bit AVR with 8KB flash, 512 bytes RAM
- Preloaded with firmware that allows motor control from TinyDuino via I2C

TinyDuino Power Requirements

- Voltage: 3.0V - 5.5V
- Current: 5mA (Logic only)

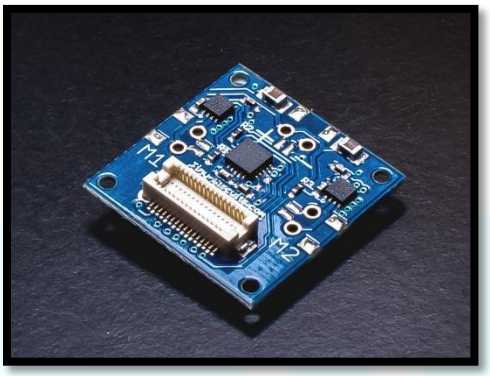
Pins Used

- A5/SCL - I2C Serial Clock line
- A4/SDA - I2C Serial Data line

MotorDriver.h Library Documentation and Example Code

The ASD2302 Dual Motor TinyShield was developed to drive 2 Brushed-DC Motors

Library Last Modified: 30 July 2015 | Documentation Last Modified: 7 February 2017



Initialization and Control:

MotorDriver()

Description

Declares the motor object and the address of the TinyShield on the I^2C bus.

Syntax

MotorDriver *motor*(address);

Parameters

motor: an object of the type MotorDriver

address: the I2C address on the TinyShield set by hardware resistors R1 and R2, default of 0. Also accepts 1, 2 or 3.

Example

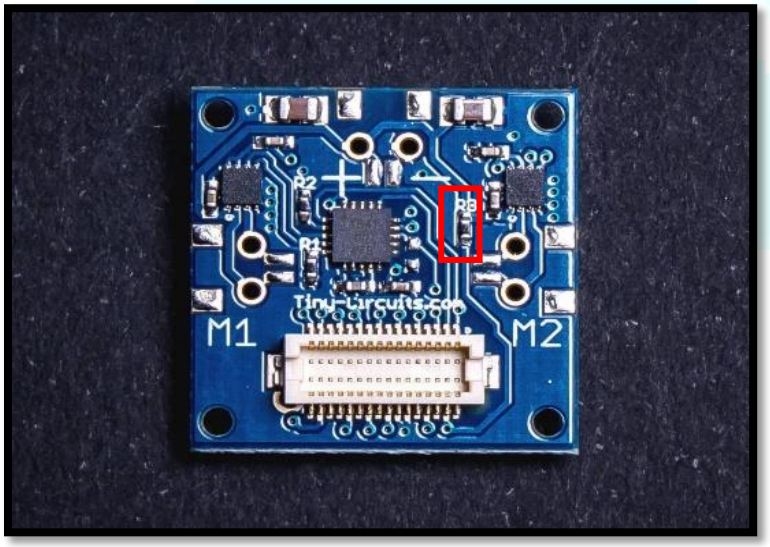
```
MotorDriver motor(0);  // Default address of the TinyShield, configured by hardware resistors R1 + R2
```

Note

The default address of this TinyShield is 0. The I2C address of the Dual Motor TinyShield can be changed by reconfiguring the resistors as follows:

R1 + R2	address: 0
R1	address: 1
R2	address: 2
Neither	address: 3

When using an higher voltage source for the motors, resistor R3 can be removed by desoldering and removing, or by gently cutting through the black conductive film on the top of the resistor, as shown below. (The TinyDuino processor will then need its own separate power source).



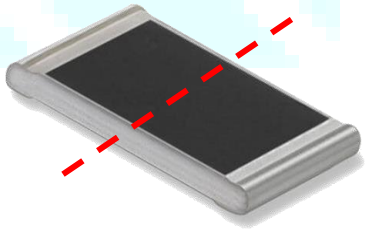


Image Courtesy of: Megatron.de

begin()

Description

Initializes the library for the MotorDriver by beginning communication with the ATtiny841.

Syntax

motor.begin(maxPWM);

Parameters

motor: an object of type MotorDriver

maxPWM: a 16 bit integer variable which sets the largest PWM value. (up to 65,535) . This value also determines the output frequency – by default, 8MHz divided by the maxPWM value.

Example

```
If(motor.begin(maxPWM)) {      //Checks to see if the TinyShield is attached
Serial.println("Motor driver not detected!");
while(1);          //Halts code execution if TinyShield was not detected, remains in infinite while loop
}
```

Note

The PWM (Pulse Width Modulation) output from the ATtiny841 is determined by the maxPWM value (counter value), which can be visualized below. A more in depth tutorial can be found here: http://www.evilmadscientist.com/2007/quick-and-dirty-d-to-a-on-the-avr-a-timer-tutorial

If the Dual Motor TinyShield is not detected, the begin() function will halt the code execution, performed by while(1).

For More information about how PWM works, see this tutorial: https://www.arduino.cc/en/Tutorial/PWM

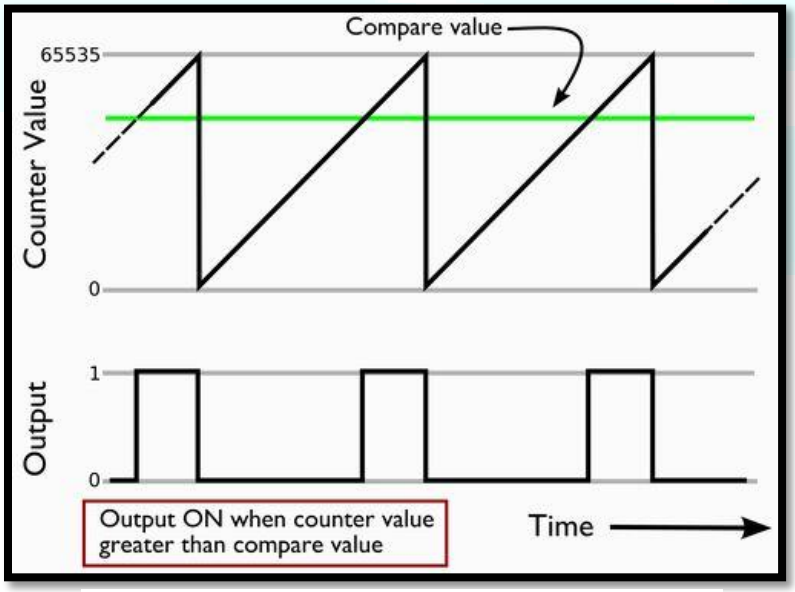


Image Courtesy of: Evil Mad Scientist Laboratories

writeCommand()

Description

Sends a command and one, two, or four 16-bit values to the ATtiny841 firmware via I^2C.

Syntax

motor.writeCommand(command, val1);

motor.writeCommand(command, val1, val2);

motor.writeCommand(command, val1, val2, val3, val4);

Parameters

motor: an object of type MotorDriver

command: any of the defined commands used to talk to the ATtiny841

val1: 16 bit value passed to the ATtiny841 firmware.

val2: Optional 16 bit value passed to the ATtiny841 firmware.

val3: Optional 16 bit value passed to the ATtiny841 firmware.

val4: Optional 16 bit value passed to the ATtiny841 firmware.

Example

```
motor.writeCommand(COMMAND_MOTOR_1, 100);
```

setFailsafe()

Description

The failsafe turns off the motors if a command is not sent in a certain amount of time. It is used to prevent damage to the motors or stop unintended movement if the processor fails or gets hung up. Set it to 0 or comment out to disable.

Syntax

motor.setFailsafe(value);

Parameters

motor: an object of type MotorDriver

value: the time (in milliseconds) for which the motors will turn off

Example

```
motor.setFailsafe(1000);  // Sets a 1 second timeout
```

setMotor()

Description

Drives the specified motor at the specified speed and direction.

Syntax

motor.setMotor(output, value);

Parameters

motor: an object of type MotorDriver

output: which motor to address, motor 1 or motor 2

value: -maxPWM to +maxPWM where (–) makes the motor spin in the opposite direction.

Example

```
// Spins both motors up from rest to full speed, simultaneously.
for(int i=0; i<=maxPWM, i+=stepSize) {   //Increments the speed value
        delay(10);                       //10ms delay
        motor.setMotor(1, i);            //Spin motor 1 at speed i
        motor.setMotor(2, i);            //Spin motor 2 at speed i
}
```

read()

Description

This function is used by the library internally to read register values inside the ATtiny841.

Syntax

motor.read(register);

Parameters

motor: an object of type MotorDriver

register: the address of the ATtiny841 memory register to read from

Returns

The data read from the ATtiny841 register

---

writeByte()

Description

This function is used by the library internally to send byte commands or values to the ATtiny841 firmware.

Syntax

motor.writeByte(b1);

motor.writeByte(b1, b2);

motor.writeByte(b1, b2, b3);

Parameters

motor: an object of type MotorDriver

b1: byte 1 of information to be sent to the ATtiny841

b2: byte 2 of information to be sent to the ATtiny841

b3: byte 3 of information to be sent to the ATtiny841

Included Definitions and Constants

*These definitions and constants can be found in the header file for this library and are used internally

```
#define T841_ADDRESS 0x62
#define COMMAND_SET_MODE 0x00
#define COMMAND_SERVO_1 0x01
#define COMMAND_SERVO_2 0x02
#define COMMAND_SERVO_3 0x03
#define COMMAND_SERVO_4 0x04
#define COMMAND_MOTOR_1 0x05
#define COMMAND_MOTOR_2 0x06
#define COMMAND_ALL_PWM 0x07
#define COMMAND_TIMER_1 0x08
#define COMMAND_TIMER_2 0x09
#define COMMAND_PRESCALER_1 0x0A
#define COMMAND_PRESCALER_2 0x0B
#define COMMAND_CLOCK_PRESCALER 0x0C
#define COMMAND_SET_SLEEP_MODE 0x0D
#define COMMAND_SLEEP 0x0E
#define COMMAND_SET_FAILSAFE_VALUES 0x0F
#define COMMAND_SET_FAILSAFE_PRESCALER 0x10
#define COMMAND_SET_FAILSAFE_TIMEOUT 0x11
#define COMMAND_ALL_PWM_8 0x12

#define T841_CLOCK_PRESCALER_1 0x00
#define T841_CLOCK_PRESCALER_2 0x01
#define T841_CLOCK_PRESCALER_4 0x02
#define T841_CLOCK_PRESCALER_8 0x03
#define T841_CLOCK_PRESCALER_16 0x04
#define T841_CLOCK_PRESCALER_32 0x05
#define T841_CLOCK_PRESCALER_64 0x06
#define T841_CLOCK_PRESCALER_128 0x07
#define T841_CLOCK_PRESCALER_256 0x08

#define T841_TIMER_PRESCALER_0 0x00
#define T841_TIMER_PRESCALER_1 0x01
#define T841_TIMER_PRESCALER_8 0x02
#define T841_TIMER_PRESCALER_64 0x03
#define T841_TIMER_PRESCALER_256 0x04
#define T841_TIMER_PRESCALER_1024 0x05

#define T841_SLEEP_MODE_IDLE 0
#define T841_SLEEP_MODE_ADC _BV(T841_SM0)
#define T841_SLEEP_MODE_PWR_DOWN _BV(T841_SM1)

#define MODE_REGISTER_INC 0xAA
#define MODE_REGISTER_DEC 0xAB
#define MODE_COMMAND 0xAC

#define FIRMWARE_REVISION_REG 0x19
#define EXPECTED_FIRMWARE 0x1A
```

Links to External Resources

- http://www.ti.com/lit/ds/symlink/drv8837.pdf
- https://tinycircuits.com/collections/all/products/dual-motor-tinyshield
- https://github.com/search?utf8=%E2%9C%93&q=tinycircuits+motor&type=Repositories&ref=searchresults

Reference Home

Revision History

- **Revision 0: New Document Released 02/07/2017**