ICN16

H-Bridge Motor Controller

Revision:

27 March 2018 - preliminary

Introduction

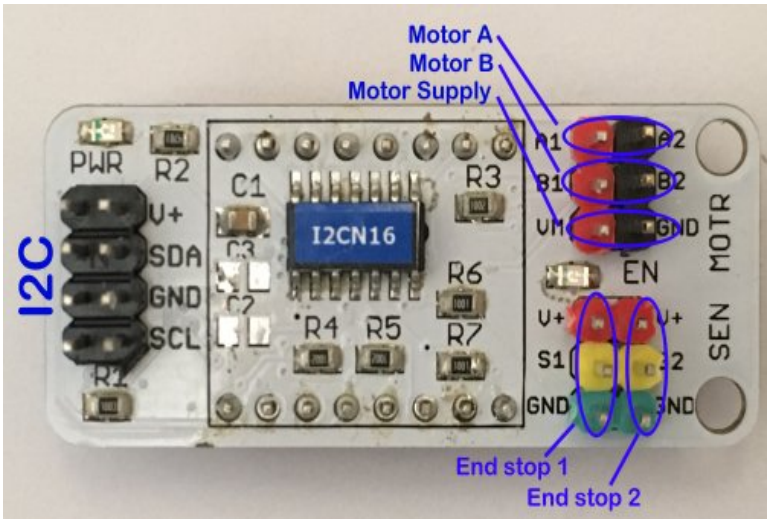
Motor controller based on the TB6612 Dual H-Bridge capable of 1.2A with 3.2A peek with a maximum of 15V.

This is an I2C device to control two DC motors or any high power requirement. In addition it has and interface and commands for 2 IR slot switches to count pulses or to provide end stops.

Features

- Configurable I2C of any address
- 4.5V to 15V Motor driver, 3.6A Peak, 1.2A per channel
- 3.3 to 5V I2C supply
- Over current protection and thermal shutdown
- Output for two motors forward and reverse
- Input for tow slot or end stop switches
- Pulse counter for servo operation

Physical Description



Pin	Pin	Description
V+	V+	Logic voltage can be from 2.7 to 5V to suit host
SDA	SDA	I2C data line
GND	DND	Ground
SCL	SCL	I2C Clock line

I2C connector

The i2C connector is a dual row of duplicate pins, this enables easy daisy chaining of the bus if required.

Pin	Pin	Description
A1	A2	Motor A output, connect a DC motor across these pins
B1	B2	Motor B output, connect a DC motor across these pins
VM	GND	This is for the motor power supply which can be up to 15V

Motor Connector

The Motor connector provides output to motors A and B and also the power supply input for both motors.

Pin	Pin	Description
V+	V+	This is intended to drive an IR slot device, it is the logic supply and be whatever that voltage is.
S1	S2	This is the input to the device and output from each slot switch. S1 is used with motor A and S2 for Motor B.
GND	GND	Ground

Slot / End stop Detection

This connector can take two standard end stop slot switches

V = V+, S = S1 or S2 and G = ground

The built in LED illuminates when the slot is empty, these can also be used for counting 'steps' when used with an optical encoder, for example in the hobby smart car.

The S1 or S2 input can also be used with a micro switch if connected from Sn to ground. When not used with the IR slot there is a pull up resistor holding the pin high, when the switch closes this will take the pin low.

EEPROM Locations

Adr	Default	Notes
0	0x55	System use
1	0x54	I2C address
4	0	Motor A Stop Value (direction)
5	0	Motor B Stop Value (direction)
14	0x54	I2C address copy
240	0x54	I2C address copy

EEPROM usage

Direction

This can have a value from 0 to 3 and applies output to A/B1 and A/B2 as follows:

Direction	**A(B)1**	**A(B)2**
0	0	0
1	1	0
2	0	1

3	1	1

Direction table

The motor will of course stop if direction 0 or 3 is used as there will be no potential difference between the pins. The EEPROM Stop Value setting can therefore be 0 or 3 and this will be applied when a command stops the motor.

Power

Power to the motor wither A or B is applied to a PWM signal the range is 0 to 255.

I2C Commands and Operation

For this device all I2C instructions are command based, that is a command followed by optional parameters.

Some commands will require a read after the command is sent. If is is a 16bit value then the high byte is sent first.

Driving the motors

The power to the H-Bridge and thus connected motor is controlled by PWM and direction. The PWM determines the power and the direction determines the direction and if the motor is on or off, see the table above.

Commands & I2C

The 8 bit address is 84 (0x54) and the 7 bit address (used on Arduino and Rpi) is 42 (0x2A).

In the examples the a shorthand is used as follows:

i2c(<adr>,[bytes to send],<bytes to receive)

So for example to send bytes 3,4,5 and receive 2 bytes back from a device with the address 42 would be:

i2c(42,[3,4,5],2)

Command	Range	Notes
Applies to all Motors		

1	0 or 1	**Enable**
		This enables the TB6612 motor driver, power can be saved by disabling (0)
		Example
		i2c(42,[1,0],0) // disable device
		i2c(42,[1,1],0) // enable device

2	n/a	**Status** Returns a status byte, see notes at the end of this table

3	n/a	**Reset** Resets the device as at first switch on. Depending on the I2C master this will likely cause a time out as there will be no reply from the device and so this may cause an I2C error from the master Example i2c(42,[3],0)
Motor / Slot Commands Motor A is 10+ Motor B is 20+		
10 or 20	0 to 255	**Motor Power** Sets the motor power Example i2c(42,[10,128],0) // set ½ power
11 or 21	0 to 3	**Motor Direction** See the direction table in the above text Example i2c(42,[11,1],0)
12 or 22	0 to 255 0 to 3	**Motor BOTH Power and Direction** Example I2c(42,[22,75,2],0)
13 or 23	n/a	**Gets Current power setting** Example i2c(42,[13],1)
14 or 24	n/a	**Gets value on S** Gets the value on S1 or S2 directly Example i2c(42,[24],1) // gets value on S2
15 or 25	n/a	**Resets slot count** Each time either S1 or S2 goes from 0 to 1 a counter is incremented, this will reset the counter to 0 Example i2c(42,[25],0) // reset counter for S2
16 or 26	n/a	**Gets Slot Count** Each time either S1 or S2 goes from 0 to 1 a counter is incremented, this will return the value of that counter. NOTE this is a 16 bit counter and so will return back to 0 when a count of 65535 is reached. Example i2c(42,[16],2) // 2 bytes returned high byte first
17 or 27	1 or 2 0 or 1	**Drive to End Stop with direction** This will set the motor going (assuming power has been set) in a given direction until the end stop is activated at which point the motor will stop. Direction is 1 or 2 as given in the direction table (above text). The end stop value is either 0 or 1 which will depend on if the motor is

		required to stop when S is a 0 or when S is a 1 Example i2c(42,[17,1,0],0) // This will start motor A in direction 1 and when S becomes 0 the motor will stop
18 or 28	1 to 65535 1 or 2	**Drive to Count with direction** This is for when the slot switch is set up as a rotary encoder, each transition from 0 to 1 is counted as 1 and stored in an internal counter. When a limit is reached set by this command the motor stops. The number of steps is a 16 bit value, high byte first. Example i2c(42,[18,1,0xf4,2) // start motor A in direction 2 and stop when 500 counts reached
System		
		Reset EEPROM This is a special general call (address 0) command that will reset the contents of the EEPROM. It does not need an I2C address and so can reset the device even if the I2C address is corrupt. Example I2C(0,[0x55],0) // 0x55 must be sent after general call
60	Adr 6-255	**Change I2C address** The address must be an even value, this is the 8 bit or full address where an even address is write and an odd address is read. It will not take effect until reset. The same effect can be achieved by writing to the EEPROM Example i2c.(42,[60, 88],0) // change 7 bit address to 44
61	Adr 0-255	**Read EEPROM** This will read a single value from an address in EEPROM Example i2c(42,[61,5],1) // read byte from EEPROM address 5
62	Adr 0-255 data 0-255	**Write EEPROM** Writes a single byte to the given address Example i2c(42,[62,33,7],0) // write 7 to EEPROM address 33
63	n/a	**Device ID** Returns device ID Example i2c(42,[63],2) // returns a 16 bit value
64	n/a	**Firmware Version** Returns 3 bytes Vh,Vm,Vl Example i2c(42,[64],3)
66	n/a	**Reset**

		Resets device similar to first switch on Example i2c(42,[66],0)

Status

Bits	Description
7:6	Motor B Mode
5:4	Motor A mode
3:2	Motor B Direction
1:0	Motor A direction

Status Table

Direction is 0-3 as set by the direction command

Mode is either 1 for end stop mode or 2 for slot count mode