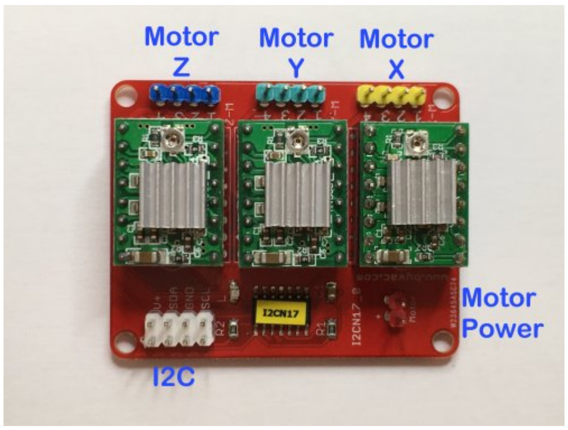
ICN17

I2C 3 Port stepper Motor Driver



Revision

- 17 March 2017 First

Introduction

This is an I2C stepper motor controller that will support up to 3 stepper motors with multiple stepping options.

Features

- Configurable I2C of any address
- Operates from 3V3 to 5V
- 3 independent stepper motor drivers
- I2C can be daisy changed
- I2C address can be changed by the user

Motor Driver Specification

Each stepper driver (A4988) can be independently controlled for current limiting and step ratio.

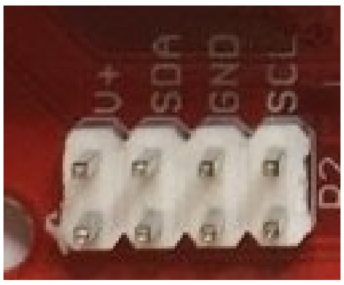
- Operates from 8V to 35V
- Up to 2A with sufficient cooling
- Five optional step mode: full, 1/2, 1/4, 1/8 and 1/16.
- Adjustable current control lets you set the maximum current output with a potentiometer, which lets you use voltages above your stepper motor’s rated voltage to achieve higher step rates

- Over-temperature thermal shutdown, under-voltage lockout, and crossover-current protection
- Short-to-ground and shorted-load protection

Physical Description

The board is a carrier and I2C interface for 3 intendant stepper modules.

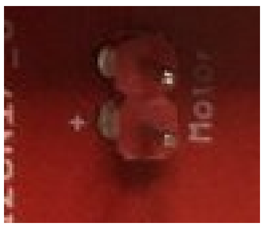
I2C Interface



SCL	I2C Clock line
GND	Ground
SDA	I2C Data Line
V+	Logic voltage can be from 2.7 to 5.5V

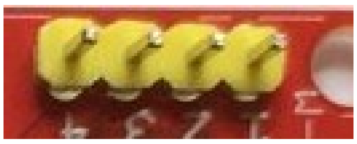
** The pins are duplicated to make daisy chaining easier

Motor Power



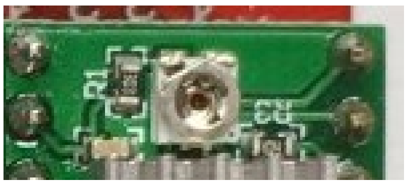
GND	Motor ground pin
+	Motor + 8V to 35V

Stepper



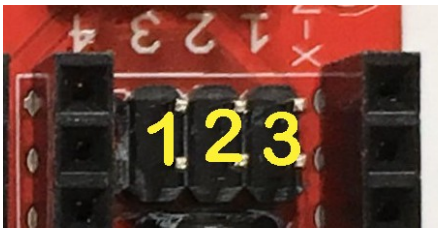
There are three of these one each for X, Y and Z

Current Control



The small trimmer on top of each module can control the current to the motor.

Stepping Rate



There are three jumpers per motor designated as above, this controls the step rate for each motor

1	**2**	**3**	**Rate**
			Full step
x			Half step
	x		Quarter step
x	x		Eight step
x	x	x	Sixteenth step

X is closed

EEPROM Locations

Adr	Default	Notes
0	0x55	System use
1	0x52	I2C address
2	19	reserved
14	0x52	I2C Address
16	5	X Ramp up # steps
17	5	X Ramp down # steps
18	5	Y Ramp up # steps
19	5	Y Ramp down # steps
20	5	Z Ramp up # steps
21	5	Z Ramp down # steps
240	0x52	I2C address copy

EEPROM locations and defaults

The EEPROM contains some system data that will normally be changed by the appropriate I2C command.

Any unused locations can be changed by the user, there will normally not be any need to change the system locations using the I2C read and write commands.

I2C Address

For security this is stored in three places and at switch on is checked that all three agree on the address. This is so that in the unlikely event that one location becomes corrupt it will be changed back on a voting system.

There is an i2C command to change the address that will change all three values at once.

Ramp Values

The values here are the default values at switch on, they can only be changed by the I2C Write to EEPROM command.

Motor Operation

Individual

The I2C controls the three A4988 drivers, each has a separate command. Once the number of steps are sent the motor begins stepping at the specified steps per second.

At any time the I2C can read the number of steps there are left to go for a particular channel so that the host knows when the operation has finished

Synchronous

There is a very small amount of time that the I2C command takes to send a command and it may be that two or more motors need to start exactly at the same time. For this there is a command that will send the number of steps to each motor all at once.

Note

This device is not intended to drive CNC machines as the stepping and synchronizations would unlikely not be precise enough for that job. To track a circle or ellipse for example would need good synchronisation between 2 motors and the I2C interface is not really capable of that.

I2C Commands and Operation

Default device address is 0x52 (82) 8 bit address this translates to the following:

Eight bit addressing where the read write flag is part of the address:

Write address 82

Read address 83

Seven bit addressing where there are read and write i2c commands

(Arduino, MicroPython)

I2c Address 41

The I2C operates independently from the stepping action and so will always respond to the host.

Commands

The general format of the I2C is to send a command followed by the parameters to that command.

The example uses python simple i2c which is one command for both read and write:

i2c(<address>,[bytes to send],<number of bytes to receive 0 for none)

For example, an EEPROM at address 0xD0 requires a word (2 bytes) for the address to be sent then it will return 1 byte from that location the exampe would be for reading at address 200:

i2c(0xd0,[0,300],1)

Another example would be to write 25 to address 1000:

i2c(0xd0,[3,0xe8, 25],0)

General Notes

The maximum number of steps is 65534 and is always supplied as two bytes, high byte first so 1000 would be 3,0xe8

Command	Range	Notes
1	0-1	**Enable** Globally enables (1) or disables (0) the stepper motors. This can be used to stop the motors overheating. Example: i2c(41,[1,1],0) // enable motors
2		**Status** Returns status byte: Bit 0: Motor X 1 busy 0 idle Bit 1: Motor Y 1 busy 0 idle Bit 2: Motor Z 1 busy 0 idle Bit 3: Enable state 1 is enabled Examples i2c(41,[2],1)
3		**Reset** Resets motor controller, this is just the motor part, for a full reset see the system reset below Example i2c(41,[3],0)
4	0-65534	**Step All** This command will step all of the motors at once but ignore any motor with a 0 value. All three values must be sent at once. i2c(adr,[X,Y,Z],0)

		Example, Step motors X and Z 300 steps to start together i2c(41,[4,1,44,0,0,1,44],0) 4 is the command 1,44 is 300, supplied high byte low byte 0,0 is Y which will be ignored 1,44 is Z same as X
Individual Motor 10 refers to X, 20 refers to Y and 30 refers to Z		
10,20,30	0-255	**Ramp Up** Sets the ramp up in number of steps used. The Steps per second (sps) value is initially divided by this value and incremented until it reached the desired sps. The Ramp up value is applied at the beginning of stepping. Example i2c(41,[10,15],0) // Motor X ramp up value
11,21,31	0-255	**Ramp Down** This is as ramp Up except that it is applied to the end of stepping Example i2c(41,[31,10],0) // sets ramp down of Z to 10
12,22,32	0 – 20000	**Speed** Sets the speed in steps per second (sps) Example i2c(41,[22,1,0xf4],0) // set Y to 500 sps
13,23,33	0 – 65534	**Steps** Sets the total number of steps for the individual motor to step. This will also initiate the stepping providing the motor driver is enabled. This value is the total value regardless of any ramp up or ramp down. Example i2c(41,[13,0,48],0) // step X 48 times
14,24,34	0 – 1	**Direction** Sets the direction of step for that motor

15,25,35	0 – 65534
	0 - 1
		Step and Direction
		This is a combination of the commands above and sets the number of steps and direction in one command.
		Example
		i2c(41,[35,0,96,1],0) // Z 96 steps direction 1
		NOTE: Direction depends on motor wiring but 0 is opposite to 1

16,26,36		**Steps to go** Returns the number of steps to go for the individual motor. Example i2c(41,[26],2) // Number of steps for Y motor

		NOTE: 2 bytes are returned high byte first
	System	
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(41,[60,<new adr>],0)
61	Adr 0-255	**Read EEPROM** This will read a single value from an address in EEPROM Example i2c(41,[61,adr,],1)
62	Adr 0-255 data 0-255	**Write EEPROM** Writes a single byte to the given address Example i2c(41,[62,adr,data],0)
63		**Gets device ID as 2 bytes** The first byte is the high byte of a 16 bit number and the second byte is the low byte Example i2c(41,[63],2)
64		**Gets firmware version as 2 bytes** Example i2c(41,[64],2)
65	N/a	**Sleep** Places device in sleep mode Example i2c(41,[65],0)
66	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(41,[66],0)