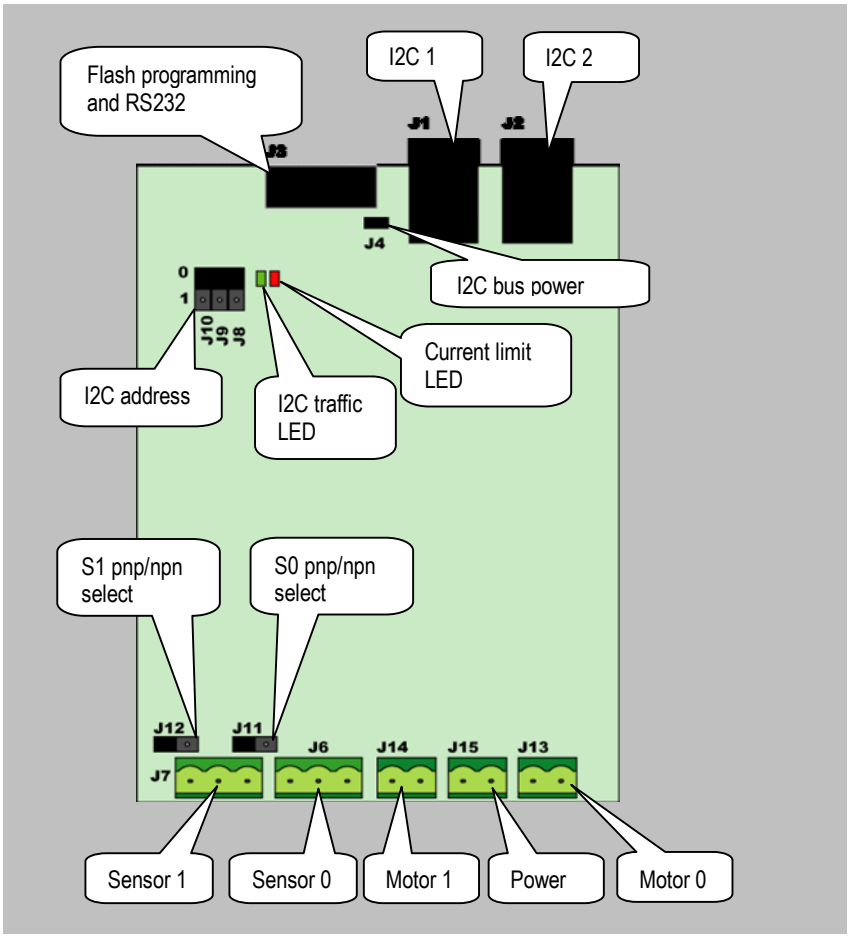
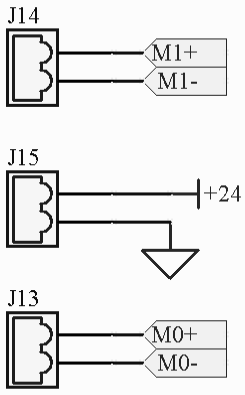
CONNECTORS AND JUMPERS



POWER AND MOTOR OUTPUTS

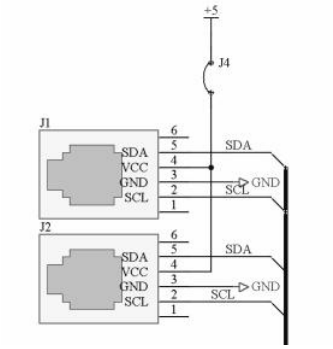


24V power is supplied at J15.

Each motor is driven with independent speed, direction and current limit setting. The PWM frequency is 22 kHz. Current limit is implemented by limiting the pulse duration cycle-by-cycle when necessary.

If either motor reaches its current limit, the red LED is lit.

I2C BUS



There are two RJ-45 I2C connectors wired in parallel. Having two connectors allows to daisy-chain multiple boards.

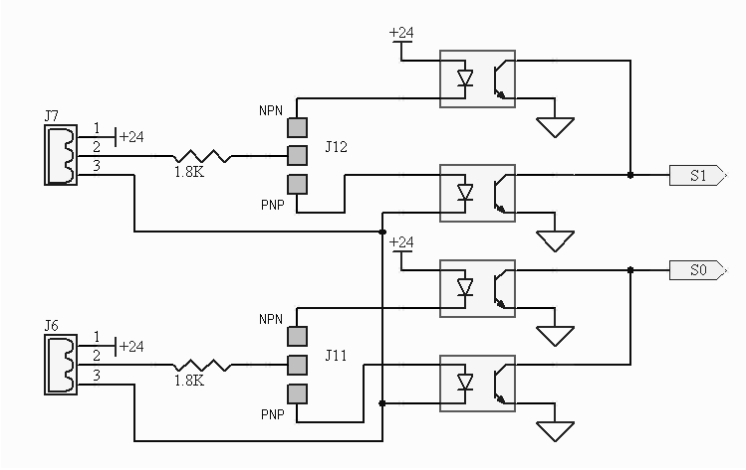
The jumper J4 can optionally provide +5V up to 100mA to the bus from the on-board supply. Typically, this is used in single-board configurations. If bus power is provided by the i2c master, **J4 must be removed**.

Each board is assigned an address between 0x000 and 0x111 using J10 J9 J8 to set the corresponding bit (J10 is MSB), allowing up to 8 boards on the bus.

The green LED indicates both power-on and I2C traffic. It blinks at 2s interval whenever the board has power and blinks each time the board is being addressed by the I2C master. This is useful for diagnosing I2C traffic problems in master-slave setups.

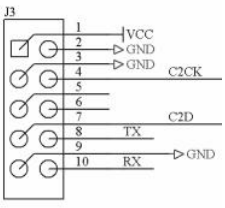
Typically, a master will poll the board often (for example at 50ms) to read the current limit status and the inputs. This will cause the green LED to blink fast. Slow blinking at 2s indicates that the master doesn't talk to the board. The reason could be simply wrong board address (J10 J9 J8).

INPUTS



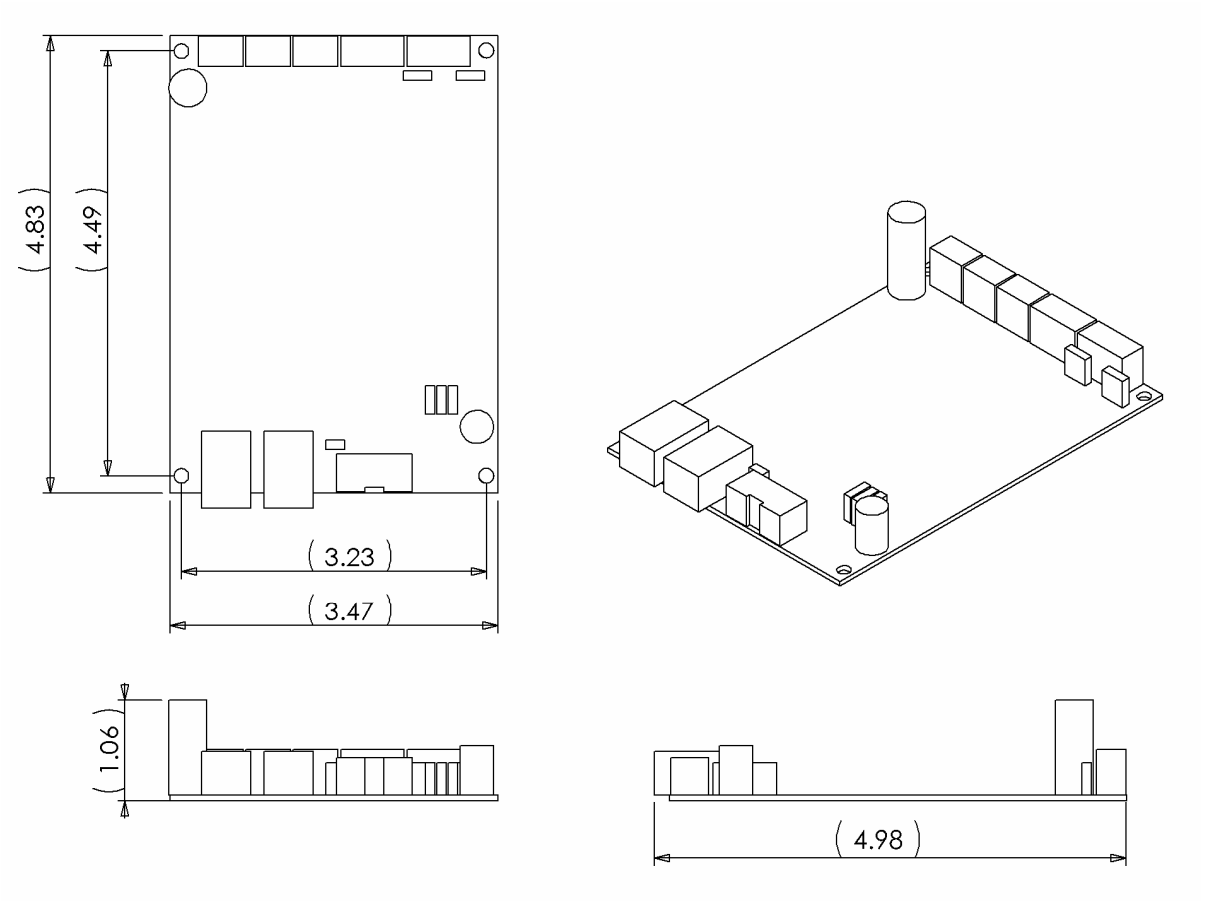
The inputs accept optical or magnetic sensors with either PNP or NPN outputs. The sensor type is configured by J11 and J12.

FLASH PROGRAMMING AND RS232



The board firmware can be reprogrammed via J3 using the standard USB programmer available from Silicon Laboratories. On pins 8, 9 and 10, which are not used for programming, a 3.3V level RS232 interface is available.

DIMENSIONS



LOW LEVEL I/O

At low level, the master controller needs to implement only two basic i2c functions:

- Writing two bytes to the board
- Reading two bytes from the board

The I2c protocol requires an address byte to be sent after the start condition. The address byte has the following structure:

	Device id				Board address			Read/write
bit	B7	B6	B5	B4	B3	B2	B1	B0
value	1	0	0	0	A2	A1	A0	R/W

Here A2A1A0 is the board address set by the jumpers J10 J9 J8. The last bit indicates Read (R/W=1) or Write (R/W=0) operation.

The read and write sequences follow the i2c standard. In pseudo code they would look like this:

Writing

```
I2C_START
SEND_ADDR_BYTE_WRITE
SEND_FIRST_BYTE
SEND_SECOND_BYTE
I2C_STOP
```

Reading

```
I2C_START
SEND_ADDR_BYTE_READ
READ_FIRST_BYTE
I2C_ACK
READ_SECOND_BYTE
I2C_NAK
I2C_STOP
```

CONTROLLING THE MOTORS

All commands to the board have the following format:

- First byte is a command

- Second byte is a parameter value

Byte1 value	Command	Byte2 meaning	Byte2 value	Note
1	Run M0 forward	Speed	0, 10-245	Speed=0 stops the motor
2	Run M0 reverse	Speed	0, 10-245	
3	Run M1 forward	Speed	0, 10-245	
4	Run M1 reverse	Speed	0, 10-245	
5	Set current sense blanking	Current Sense Blank Period	5-245	Command normally not used. See discussion ahead
6	Set M0 current limit	Current limit	0-255	Current [mA] = 27.3 * Byte2
7	Set M0 current limit	Current limit	0-255	

The RUN commands have a one byte speed parameter. There is no STOP command; to stop a motor send a RUN command with SPEED=0. Any other value will set a proportional duty cycle to the PWM with 255 corresponding to the maximum speed.

READING THE BOARD

The read command results in two bytes whose bits have the following meaning.

Byte1

	Current limit flags and sensor status							
bit	B7	B6	B5	B4	B3	B2	B1	B0
value	0	0	0	0	CL1	CL0	S1	S0

Byte2

	Pulse counter for M1 (PC1)				Pulse counter for M0 (PC0)			
bit	B7	B6	B5	B4	B3	B2	B1	B0
value	PC13	PC12	PC11	PC10	PC03	PC02	PC01	PC00

The first returned byte contains 4 status bits. S0 and S1 reflect the sensor status. CL0 and CL1 are flags which, when raised, indicate that the current of the corresponding motor has reached the preprogramed limit and each PWM pulse is being shortened so as not to exceed this limit.

Byte2 contains two counters which are used if one keeps track of the number of turns performed by each motor. Some motors contain a built-in hall sensor that produces a pulse on each turn. Such a sensor can be connected to the corresponding sensor input.

If the board is polled at a particular interval, say 50ms, the polling interval is too big to actually count the number of pulses coming from the hall sensor. This is when the pulse counters are used. A pulse counter will contain the number of pulses that came since the last read operation. Each read operation clears the pulse counter. So, to keep track of the total number of turns the

master needs simply to accumulate the pulse counter value. Of course, polling must be fast enough so that the 4-bit pulse counters don't overflow between readings.

TIPS ON USAGE

The range of possible applications for VD203 is very broad, but no controller is suitable for every task.

VD203 is good for:

- Open loop speed control
- Systems with up to 16 motors and optical/magnetic sensors using master controller
- Simple systems with up to 2 motors and 2 sensors, which can be implemented using a single VD203 with custom firmware.
- Systems with high friction and low inertia. VD203 is not designed for active braking of a speeding motor with high-inertia load. Motors with worm or spur gearboxes are best suited for VD203.

VD203 is not suitable for

- Closed loop speed control.
- Systems with high inertia and low friction that require active motor braking (i.e. dissipating load's kinetic energy). Motors with no gearbox where the load is directly coupled with the shaft can be problematic.

Avoiding end-position switches with the current limit feature.
Often a mechanism needs to move between two fixed positions. Some examples are:

- opening/closing doors,
- lifting/lowering mechanisms
- extend/retract mechanisms

A typical implementation uses end position switches to turn on/off the actuators. However, mechanical switches are among the first components to fail. Optical and magnetic sensors can be expensive and need alignment.
With VD203, switches can still be used, but there is an alternative approach: detect an end position by the raise in motor current. The current limit ensures that no excessive force is applied.

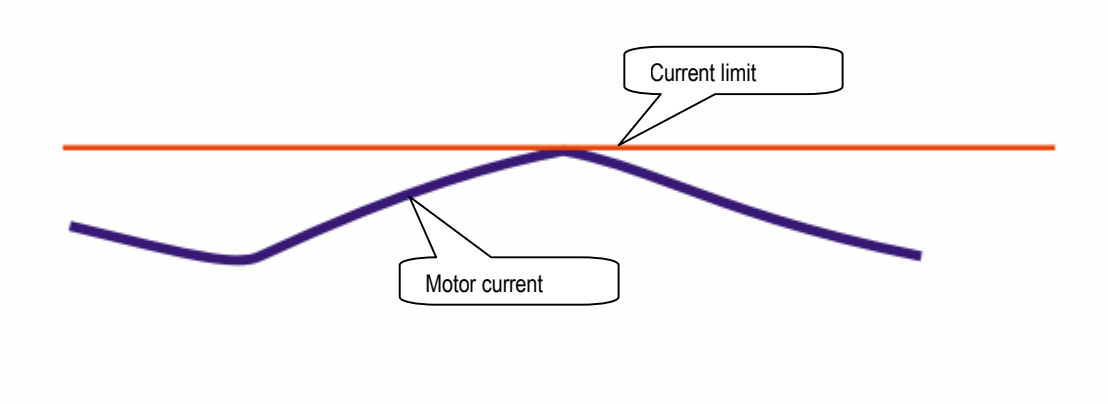
Here is a sample work sequence:

- Set current limit to a reasonable value. If the current limit is too low, the motor might not be able to move the load. If it's too high, a small motor might not be able to reach it.
- Start the motor by setting speed greater than zero. For the first 200-300 ms ignore the CL flag. Often in the beginning of the motion the current limit will kick in due to the load inertia.
- Monitor the CL flag. When it becomes 1, turn the motor off by setting its speed to zero.

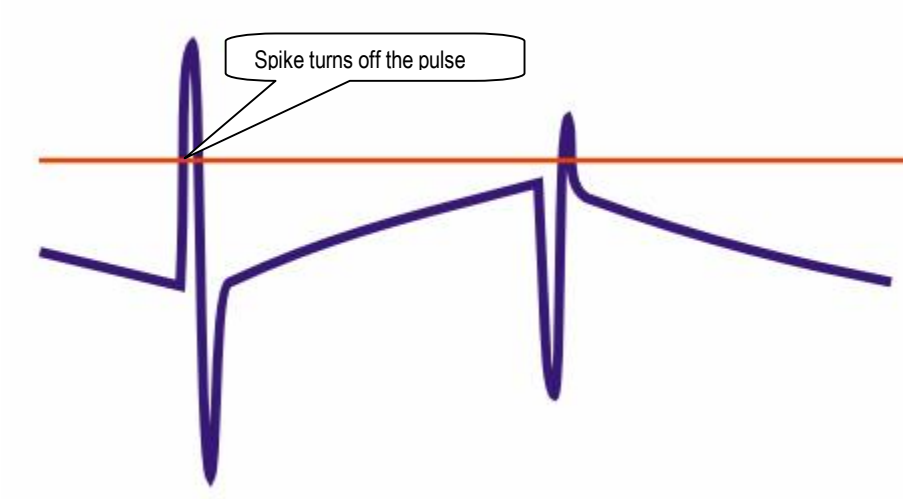
MOTORS WITH INTERNAL CAPACITORS

Some brush motors have internal capacitors meant to filter the brush switching noise. However in applications that use PWM with cycle-by cycle current limit, these capacitors might become a problem, especially if their value is larger.

A motor without capacitors is an inductive load. When driven with PWM, its current looks like this:



A motor with filter capacitors shows current spikes at each pulse, due to the capacitor current:



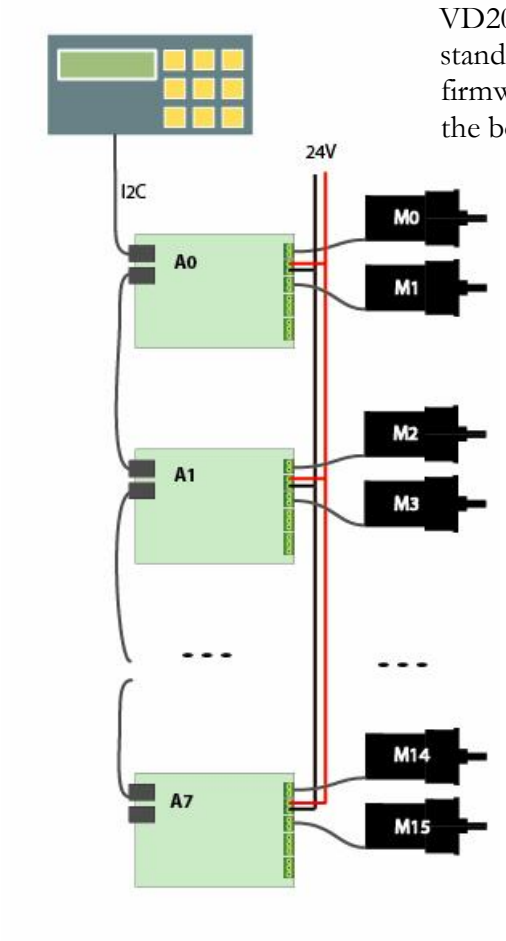
These spikes fool the current limit circuitry into thinking that the limit has been reached and turn off the PWM pulse prematurely. The result is that the PWM stays at minimum duty cycle and the motor barely moves or doesn't move at all.

In VD203 this issue is addressed by disabling the current sense circuitry right after the switching, until the capacitor current spike has settled. The time during which current sense is disabled is controlled by the **Current Sense Blank Period** parameter. The default value loaded at power on is 16, and has been found to be adequate for most motors. Changing this parameter is not recommended, unless a particular motor has very large filter capacitors whose spikes take longer to settle.

CONTROL TIMEOUT SAFETY FEATURE

VD203 in slave mode must be constantly supervised by the master. If the master fails to communicate (disconected line or a hung program) , the motors attached to VD203 could be left running until they reach the end point and enter current limit mode. If the master is not alive to turn them off, the motors and the board can overheat.
To prevent this from happening, VD203 requires that the master talks to it at intervals not longer than 2 seconds. If no command or status polling comes for more than 2 seconds, VD203 turns off both motors.
The timeout is also helpful during master debugging on a running machine. Setting breakpoints in master code could be dangerous, because slave boards could be left running, possibly damaging the mechanics. The timeout feature ensures that upon hitting a breakpoint, all motors will stop within 2 seconds.

POSSIBLE CONFIGURATIONS



VD203 is used in two basic configurations : master/slave and standalone. Off the shelf, the board is shipped with slave firmware. When assembling master/slave systems, grounding the boards is a critical issue. The problem arises from the motor ground currents affecting the i2c bus. For best results, all boards should be mounted on a metal plate with conductive standoffs, providing low impedance paths for the ground currents.

For small cost sensitive machines, a single VD203 with custom software can often do the job. With 16K flash and 1K RAM onboard, VD203 can tackle quite complex tasks by itself. Such a system may include nothing but the board, the motors and up to two sensors. Velox Devices offers a 2x16 character display with 4 buttons that can be driven by VD203 in standalone mode.