VideoRay Motor Controller Firmware

Version 0.4.0

Contents

[Summary: 3](#_Toc397580821)

[LED Blink Patterns 4](#_Toc397580822)

[Faults 5](#_Toc397580823)

[Communication Timeout 5](#_Toc397580824)

[CSR Memory Map 6](#_Toc397580825)

[CSR Field definitions 8](#_Toc397580826)

[Custom Commands and Responses 9](#_Toc397580827)

[PROPULSION\_COMMAND: 0xaa 9](#_Toc397580828)

[RESPONSE\_THRUSTER\_STANDARD: 0x02 9](#_Toc397580829)

[Diagnostic mode 10](#_Toc397580830)

[Motor Actuation Mode 10](#_Toc397580831)

[Configuration Mode 10](#_Toc397580832)

[Configuration fields 12](#_Toc397580833)

[1: Motor control flags: 12](#_Toc397580834)

[2: Motor poles: 12](#_Toc397580835)

[3: PWM deadband: 12](#_Toc397580836)

[4: Commutation threshold: 12](#_Toc397580837)

[5: Commutation loss timeout: 12](#_Toc397580838)

[6: Startup duty cycle: 12](#_Toc397580839)

[7: Startup initial rpm: 12](#_Toc397580840)

[8: Startup end rpm: 12](#_Toc397580841)

[9: Ramp startup time: 12](#_Toc397580842)

[10: Negative deadband: 12](#_Toc397580843)

[11: Positive deadband: 13](#_Toc397580844)

[12: Negative power limit: 13](#_Toc397580845)

[13: Positive power limit: 13](#_Toc397580846)

[14: Ramp-up rate: 13](#_Toc397580847)

[15: Ramp-down rate: 13](#_Toc397580848)

[16: Fault control: 13](#_Toc397580849)

[17: Minimum voltage: 13](#_Toc397580850)

[18: Max bus voltage: 13](#_Toc397580851)

[19: Max current: 13](#_Toc397580852)

[20: Max temperature: 14](#_Toc397580853)

[21: Max stall count: 14](#_Toc397580854)

[22: Node Id: 14](#_Toc397580855)

[23: Group Id: 14](#_Toc397580856)

[24: Motor Id: 14](#_Toc397580857)

[25: System control flags: 14](#_Toc397580858)

[26: Motor fault interlock: 0 14](#_Toc397580859)

# Summary:

The motor controller speaks RS485 at 115200 baud.

On power up a serial bootloader is run.

The vr\_refresh.exe application can be used to flash new firmware onto the controller. (See upgrading firmware section below)

The bootloader will jump to the application after 1 second.

If “+++++” (5 plus signs) are entered within 5 seconds the device will enter diagnostic test mode where a simple ascii user interface is implemented and terminal can be used to control and configure the device. (See Diagnostic mode below)

In normal operation the device implements a CSR type memory mapped data space. The VideoRay CSR comms protocol is used for communication. See <https://github.com/videoray/VRCommsProtocol_doc/raw/master/VR_CSR_Communication_Protocol.doc> for more information on the base binary protocol.

# LED Blink Patterns

The POWER LED is a constant on RED when power is applied.

The other indicator is a 3 color LED package, which is made up of the STATUS led (blue), the MOTOR led (green) , and the FAULT led (RED)

In diagnostic mode the STATUS and FAULT LEDs are turned ON.

The STATUS LED is blue and has several blink patters:

Bootloader:

3 blinks on startup

Rapid blink if no valid application or Commanded to stay in bootloader

Normal Operation:

3 blinks on application startup

1 blink per second

Fast triple blink when a CSR communication packet is accepted

THE MOTOR LED is green and blink at a rate proportional to the RPM of the motor.

The FAULT LED is red and is active when a fault condition has been detected.

# Faults

If a fault occurs the thruster will de-energize and the fault must be cleared and the thruster re-initialized before it will actuate the motor.

## Communication Timeout

The motor controller has a communication lost timeout of 1 second. If the time since last vr\_csr packet is >= 1 second the motor will set the power to 0. Note that this does not occur in diagnostic mode. Also not that that motor is NOT de-energized but remains active with the motor power set to 0.

# CSR Memory Map

|  |  |  |
| --- | --- | --- |
| 0x0 | float rpm\_target | // rpm (RW) NOT USED YET |
| 0x4 | float pwr\_target | // -1 to 1 (RW) |
| 0x8 | float rpm | // rpm (R) |
| 0xc | float bus\_v | // volts (R) |
| 0x10 | float bus\_i | // amps (R) |
| 0x14 | uint32\_t fault | // fault flags (R) |
| 0x18 | float temp | // deg C (R) |
| 0x1c | float pwr\_actual | // -1 to 1 (R) |
| 0x20 | float rpm\_P | // 0 to 1 (R) NOT USED YET |
| 0x24 | float rpm\_I | // 0 to 1 (R) NOT USED YET |
| 0x28 | float rpm\_D | // 0 to 1 (R) NOT USED YET |
| 0x2c | RESERVED |  |
| 0x4c | uint16\_t thruster\_ID | // ordinal (RW) |
| 0x4e | RESERVED |  |
| 0x50 | uint8\_t operation\_flags |  |
| 0x51 | RESERVED |  |
| 0x54 | uint32\_t motor\_fault\_interlock | // password (RW) |
| 0x58 | RESERVED |  |
| 0x60 | uint8\_t motor\_control\_flags | // Bitfield (RW) |
| 0x61 | uint8\_t poles | // count (RW) |
| 0x62 | uint8\_t pwm\_deadband | // ticks (RW) |
| 0x63 | RESERVED |  |
| 0x64 | float commutation\_threshold | // 0 to 1 (RW) |
| 0x68 | uint32\_t commutation\_loss\_timeout | // ms (RW) |
| 0x6c | float startup\_dutycycle | // 0 to 1 (RW) |
| 0x70 | uint16\_t startup\_initial\_rpm | // rpm (RW) |
| 0x72 | uint16\_t startup\_final\_rpm | // rpm (RW) |
| 0x74 | float startup\_duration | // mS (RW) |
| 0x78 | float deadband\_neg | // -1 to 0 (RW) |
| 0x7c | float deadband\_pos | // 0 to 1 (RW) |
| 0x80 | float limit\_neg | // -1 to 0 (RW) |
| 0x84 | float limit\_pos | // 0 to 1 (RW) |
| 0x88 | float slew\_rate\_up | // delta/mS (RW) |
| 0x8c | float slew\_rate\_down | // delta/mS (RW) |
| 0x90 | float rpm\_kP | // -1 to 1 (RW) NOT USED YET |
| 0x94 | float rpm\_kI | // -1 to 1 (RW) NOT USED YET |
| 0x98 | float rpm\_kD | // -1 to 1 (RW) NOT USED YET |
| 0x9c | RESERVED |  |
| 0xa4 | uint8\_t fault\_control | // flag (RW) |
| 0xa5 | uint8\_t undervoltage\_trigger | // volts (RW) |
| 0xa6 | uint8\_t overvoltage\_trigger | // volts (RW) |
| 0xa7 | uint8\_t overcurrent\_trigger | // amps (RW) |
| 0xa8 | uint8\_t temp\_trigger | // deg C (RW) |
| 0xa9 | uint8\_t stall\_count\_max | // count (RW) |
| 0xaa | RESERVED |  |
| 0xac | uint32\_t undervoltage\_err\_cnt | // count (R) |
| 0xb0 | uint32\_t overvoltage\_err\_cnt | // count (R) |
| 0xb4 | uint32\_t overcurrent\_err\_cnt | // count (R) |
| 0xb8 | uint32\_t temp\_err\_cnt | // count (R) |
| 0xbc | uint32\_t stall\_err\_cnt | // count (R) |
| 0xc0 | RESERVED |  |
| 0xd8 | uint32\_t comms\_sync1\_err\_cnt | // count (R) |
| 0xdc | uint32\_t comms\_sync2\_err\_cnt | // count (R) |
| 0xe0 | uint32\_t comms\_headerxsum\_err\_cnt | // count (R) |
| 0xe4 | uint32\_t comms\_overrun\_err\_cnt | // count (R) |
| 0xe8 | uint32\_t comms\_payloadxsum\_err\_cnt | // count (R) |
| 0xec | uint16\_t comms\_err\_flag |  |
| 0xee | uint16\_t save\_settings | // code (W) |
| 0xf0 | uint32\_t custom\_command | // (W) Special Register |
| 0xf4 | uint32\_t FACTORY\_SERVICE\_DATA | // (R) Device specific service data |
| 0xf8 | uint16\_t CONFIG\_DATA\_SIZE | // (R) |
| 0xfa | uint8\_t CONFIG\_DATA | // (R) Special Register |
| 0xfb | uint8\_t FIRMWARE\_VERSION | // (R) Special Register |
| 0xfc | uint8\_t NODE\_ID | // (RW) Special Register |
| 0xfd | uint8\_t GROUP\_ID | // (RW) Special Register |
| 0xfe | uint16\_t UTILITY | // (W) Special Register |
|  |  |  |

# CSR Field definitions

TO BE ADDED

# Custom Commands and Responses

The motor controller firmware also support a custom command which allow for the instantaneous setting of multiple controllers on a party line comms bus.

## PROPULSION\_COMMAND: 0xaa

The propulsion command is an application custom command which is sent as a write request to CSR address 0xF0 (the custom command register. It has the following data payload format:

‘0xAA R\_ID THRUST\_0 THRUST\_1 THRUST\_2 ... THRUST\_N’

Where:

0xAA is the command byte

R\_ID is the NODE ID of the thruster to respond with data

THRUST\_X is the thruster power value (-1 to 1) for the thruster with motor id X

Typically this is sent as a group multicast to address 0x81 which is reserved for thrusters.

## RESPONSE\_THRUSTER\_STANDARD: 0x02

The standard thruster response is typically used in conjunction with the multicast PROPULSION COMMAND to retrieve data from each thruster in the system in a round robin fashion.

When the FLAG byte is set to 0x02 the RESPONSE\_THRUSTER\_STANDARD data payload is sent.

The format of this payload is defined by the following structure:

Response\_Thruster\_Standard {

/\*\* Measured shaft rotational velocity \*/

**float** rpm;

/\*\* Bus voltage (Volts) \*/

**float** bus\_v;

/\*\* Bus current (Amps) \*/

**float** bus\_i;

/\*\* Temperature (Degree C) \*/

**float** temp;

/\*\* fault flags \*/

uint8\_t fault;

}

Please see the example thruster.py for an illustration of how to use the PROPULSION\_COMMAND and parse the RESPONSE\_THRUSTERS\_STANDARD response packet.

# Diagnostic mode

Diagnostic mode is a simple ASCII terminal user interface that allows interaction with the motor controller electronics without requiring any additional topside software other than a serial terminal (such as tera term <http://ttssh2.sourceforge.jp/> )

To enter Diagnostic mode, input ‘+++++’ (5 pluses) within 5 seconds after power up.

If 5 seconds have elapsed or any vrcsr packets have been received diagnostic mode will be locked out until the next power cycle.

NOTE: Since diagnostic mode is a simple ascii terminal it is not appropriate for multiparty communications.

## Motor Actuation Mode

When the motor controller enters diagnostic mode it will go directly to motor actuation mode. In motor actuation mode the motor can be directly controlled by key presses in the terminal window.

The diagnostic menu illustrates the keys which can be used actuate the motor.

**Diagnostics:**

**=: Increase motor (0.1%) increments**

**-: Decrease motor (0.1%) increments**

**0-9: Set motor direct (10% increments)**

**Shift + 0-9: Set motor direct reverse (10% increments)**

**r: Reset motor control algorithm state**

**' ': Print data**

**c: Configuration**

**?: Print this menu**

**x: Exit diagnostic mode**

Pressing ‘x’ will exit diagnostic mode. Diagnostic mode can be re-entered for 5 seconds after exiting.

Pressing ‘c’ will bring up the configuration menu.

Actuation is immediate on each keypress, i.e. the ‘enter’ key does not have to be hit after the desired command input.

## Configuration Mode

The diagnostic configuration menu allows for various operating parameters to be set. The parameters can also be saved in non-volatile storage.

The diagnostic configuration menu displays the motor controller serial number as well as the firmware versions number and firmware inception date.

In diagnostic configuration mode the ‘enter’ key must be hit after each command input.

Pressing ‘x’ followed by enter will exit diagnostic configuration mode and return to the diagnostic menu

Pressing ‘s’ followed by enter will save the current parameters in non-volatile memory.

Pressing ‘f’ followed by enter will reset the all parameters to their hardcoded factory defaults.

Entering a number followed by enter will prompt for a new setting for the configuration parameter.

If ‘enter’ is pressed before a new value has been entered the current value will remain.

**Config: (Press enter after command)**

**Motor Controller: TH00015**

**SW Version: 0.2.1 Sep 3 2014 13:28:09**

**1: Motor control flags: 0x0**

**2: Motor poles: 14**

**3: PWM deadband: 32**

**4: Commutation threshold: 0.001000**

**5: Commutation loss timeout: 50**

**6: Startup duty cycle: 0.100000**

**7: Startup initial rpm: 50**

**8: Startup end rpm: 500**

**9: Ramp startup time: 60.000000**

**10: Negative deadband: -0.035000**

**11: Positive deadband: 0.035000**

**12: Negative power limit: -0.990000**

**13: Positive power limit: 0.990000**

**14: Ramp-up rate: 0.000025**

**15: Ramp-down rate: 0.000025**

**16: Fault control: 0x1f**

**17: Minimum voltage: 19.200001**

**18: Max bus voltage: 50.000000**

**19: Max current: 20.000000**

**20: Max temperature: 100.000000**

**21: Max stall count: 100**

**22: Node Id: 2**

**23: Group Id: 129**

**24: Motor Id: 0**

**25: System control flags: 1**

**26: Motor fault interlock: 0**

**f: reset parameters to factory default**

**s: save parameters**

**x: exit parameter mode**

## Configuration fields

### 1: Motor control flags:

The motor control flags are a bitfield that sets binary operation parameters.

Currently defined:

**CONTROL\_FLAG\_REVERSE\_ROTATION (1 << 0)**

When set reverses the rotation of the motor.

### 2: Motor poles:

Defines the number of poles in the attached motor.

### 3: PWM deadband:

Defines the time that band that insures that the motor driving FETs are off. This prevents both the high and low driving complementary FETs to be on at the same time. Units are in processor cycles.

### 4: Commutation threshold:

This defines the trigger level at which the commutation algorithm determines it is time to commutate the motor.

### 5: Commutation loss timeout:

If the time between detected commutations is greater than this value, in mS, then the algorithm will assume the motor is stalled and reset the commutation algorithm and switch to open loop startup mode.

### 6: Startup duty cycle:

The duty cycle that will be used to power the motor when in initial open-loop startup mode.

### 7: Startup initial rpm:

The initial startup RPM that will be used when the motor is in initial open-loop startup mode. The algorithm will attempt to commutate at this RPM using the Startup Duty Cycle PWM. The open loop phase will ramp up from this RPM to the startup end RPM at a rate defined by the Ramp Startup time.

### 8: Startup final rpm:

The ending startup RPM that will be used when the motor is in initial open-loop startup mode. The algorithm will attempt to commutate at this RPM using the Startup Duty Cycle PWM. The open loop phase will ramp up to this RPM from the startup initial RPM at a rate defined by the Ramp Startup time.

### 9: Startup duration:

This parameter defines the duration required to ramp from startup initial rpm to startup end rpm. This value is in units of ????

### 10: Negative deadband:

This defines the largest (closest to 0) negative value that can be set as a motor thruster setpoint. It is assumed that due to stiction the motor will not spin at power levels closer to 0 than this.

### 11: Positive deadband:

This defines the smallest positive (closest to 0) value that can be set as a motor thruster setpoint. It is assumed that due to stiction the motor will not spin at power levels closer to 0 than this.

### 12: Negative power limit:

This value is the limit in the negative direction of applicable power to the motor. The command setpoint will never be less than this value.

### 13: Positive power limit:

This value is the limit in the positive direction of applicable power to the motor. The command setpoint will never be greater than this value

### 14: Slew rate up:

This value defines the rate at which the motor will change from the current power setpoint to the target setpoint. This value defines the rate when the absolute value of the target setpoint is larger than the current setpoint. The units of this value are change in normalized power per mS.

### 15: Slew rate down:

This value defines the rate at which the motor will change from the current power setpoint to the target setpoint. This value defines the rate when absolute value of the target setpoint is less than the current setpoint. The units of this value are change in normalized power per mS.

### 16: Fault control:

This is a bit-field mask is used to enable or disable the de-energizeing of the motor on the detection of a fault. Faults will still be detected, but if the bit field is cleared the controller will not de-energize on that specific fault condition.

Current fault conditions are defined as :

FAULT\_UNDERVOLT (1<<0)

FAULT\_OVERRVOLT (1<<1)

FAULT\_OVERCURRENT (1<<2)

FAULT\_OVERTEMP (1<<3)

FAULT\_STALL (1<<4)

FAULT\_STALL\_WARN (1<<5)

### 17: Minimum voltage:

Sets the undervoltage fault trip point in Volts.

### 18: Max bus voltage:

Sets the over voltage fault trip point in Volts.

### 19: Max current:

Sets the overcurrent fault trip point in Amps.

### 20: Max temperature:

Sets the overtemp fault trip point in degrees Celsius

### 21: Max stall count:

Sets the threshold for the number of stall event detections at which to declare a fault.

### 22: Node Id:

This is the communication protocol network node id. This is used to designate this device on a multiparty rs485 communication network.

### 23: Group Id:

This is the communication protocol network node id. This is used to designate this device on a multiparty rs485 communication network.

### 24: Motor Id:

This is the communication protocol network node id. This is used to designate this device on a multiparty rs485 communication network.

### 25: System control flags:

The motor control flags are a bitfield that sets binary operation parameters.

Currently defined:

**ENABLE\_LOSS\_OF\_COMMS\_TIMOUT (1 << 0)**

When set the controller will set the power setpoint to 0 if communications have been lossed for 1 second. Clearing this flag allows for a greatly reduced communications bandwidth. However the motor will continue to spin even if communications have been lost.

### 26: Motor fault interlock:

This value must be set to the proper password to allow the faults control mask to be changed.

The password is 0xdeadbeef

## Upgrade Firmware

See the manual for updating firmware