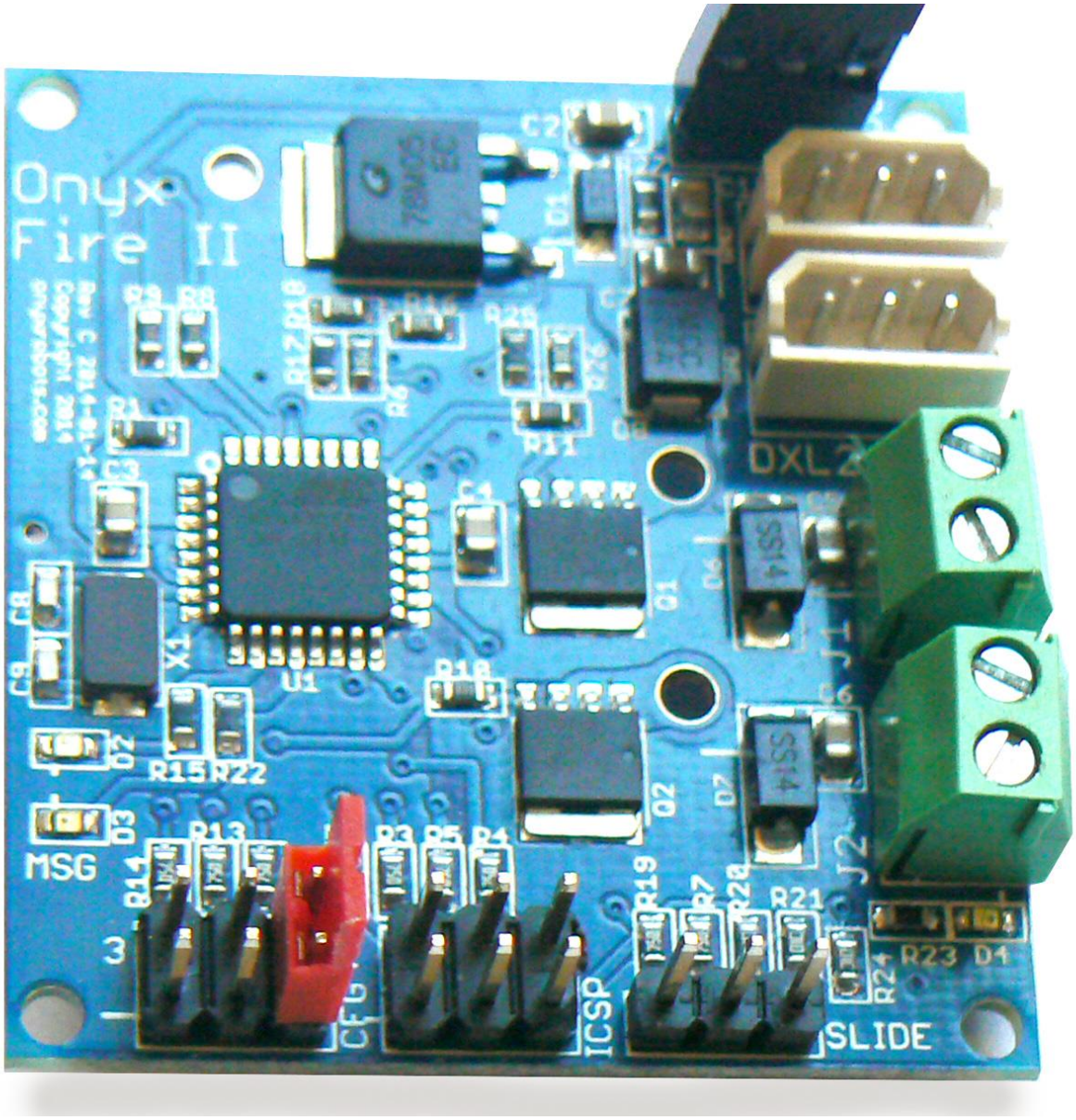
Onyx Fire II Manual



Overview

The Onyx Fire II is a high-performance electrically controlled switch for driving hobby size brushed DC gearmotors, solenoids, airsoft AEGs, 12V LED lights, and other equipment that can be switched on by supplying between 8V and 17V of voltage at up to 20A of peak power draw.

To activate the Onyx Fire II switches, you can use a 5V TTL single-ended serial bus, compatible with the Dynamixel line of servos from Robotis, up to 2 Mbps communication speed. Alternative configurations (selectable through header jumpers) allow you to also control it using 3.3V to 5V logic level voltage, or through standard RC-style PWM control pulses (simultaneous activation of both switches is not possible with RC input.)

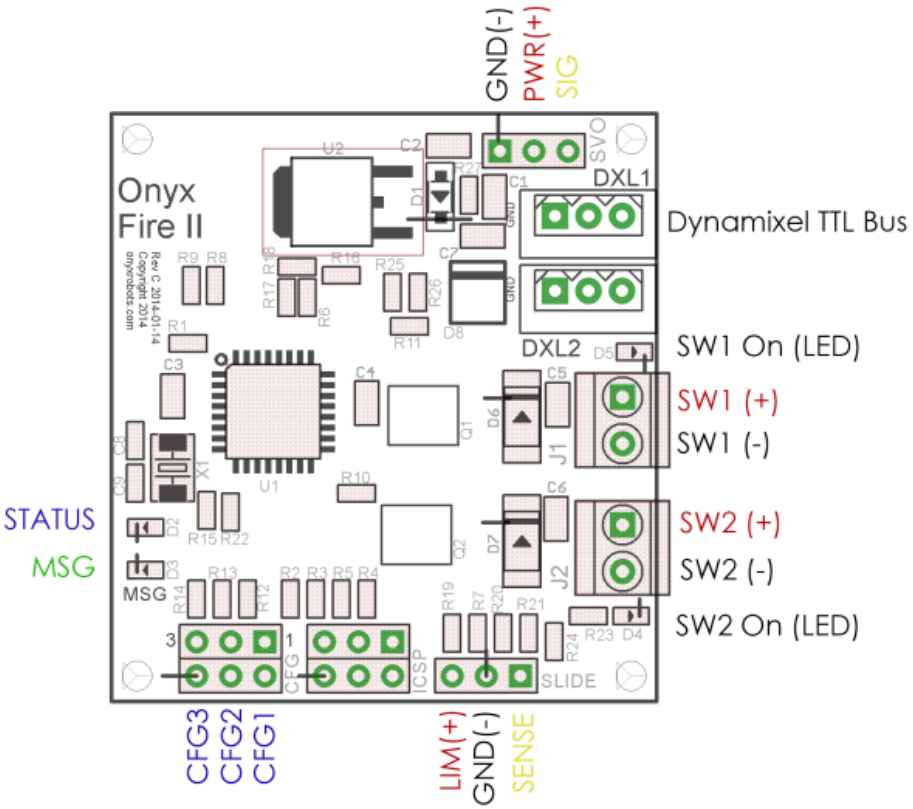
The Onyx Fire II is not a “H bridge” that can make motors spin in two directions. Instead, each Onyx Fire II contains two single-ended switches that can drive two separate devices, and one SLIDE input that can make the Onyx Fire synchronize to an external mechanism using an optional Onyx Sense sensor.

To make sure that you get optimal performance and lifetime out of the Onyx Fire II, please read this complete manual, as it contains important instructions about the connection and operation of the product.

Important Disclaimer

The Onyx Fire II is intended to be one component as part of a larger integrated system. Because there exists a large number of possibilities for catastrophic failure in a bigger system, the purchaser and user are expected to be capable of making appropriate effectiveness and safety determination related to using the Onyx Fire II in any particular system. Onyx Robots does not take responsibility for the use or failure of an Onyx Fire II board in an integrated system that is not under control of Onyx Robots. Onyx Fire II is not intended for, appropriate for, and must not be used in any safety-critical application or any application where personal injury or property destruction may be possible.

Connections and Mounting



Value	Min	Typ	Max
Input Voltage, Power	7V	8V-18V	19V
Load Current (total J1 and J2)	10 mA	5A	20A (peak)
Duty Cycle (at max load, minute)			15%
Baud Rate, TTL Serial Mode	9600 bps	1 Mbps	2 Mbps
Pulse Width, RC PWM Mode	820 µs	1500 µs	2180 µs
Input Voltage, High Signal	3.2V	4.9V	5.1V
Input Voltage, Low Signal	-0.1V	0.0V	0.8V
Operating Temperature, Ambient	0 °C	20 °C	40 °C
Dimensions (W x H x D)		41.4 x 44.0 x 12 mm	
Mounting Hole Diameter		2.15 mm	
Mounting Hole Pattern (W x H)		36.8 X 40.0 mm	
Minimum Clearance Above Board		20 mm	
Minimum Clearance Below Board		4 mm	

Use metric M2 nylon hardware for mounting the Onyx Fire II to your project. The mounting holes are not conductive, so metal hardware can also be used, but as always with metal near electronics, watch out for unintended short circuits that may cause unintended operation or damage the Onyx Fire II. There are four mounting holes intended for mounting, 36.8 mm wide and 40.0 mm high apart.

#2-56 nylon machine screw can also be used, although the fit may be tight.

There are three large, copper plated holes in the center of Onyx Fire II, next to the components labeled Q1, Q2, and U2. These are part of the electrical wiring of the board, and must not be used for mounting, or any other purpose. Doing so may short out the Onyx Fire II and destroy it.

The bottom of the Onyx Fire II board contains protruding soldered connections for components on top of the board. When mounting to another system, it is recommended to use plastic or another non-conducting material below the Onyx Fire II to avoid electrical shorts.

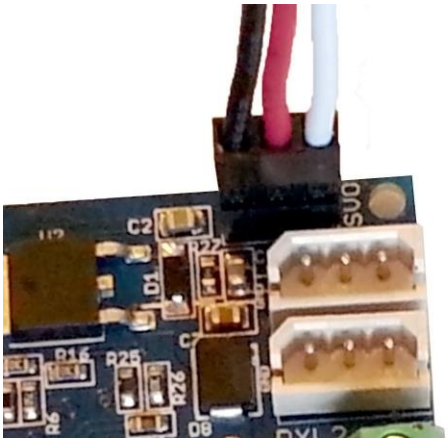
When typical cable connectors/housings are plugged into the male pin headers on the Onyx Fire II (such as SLIDE, SVO, or CFG,) a minimum of 20 mm of space needs to be available above the board surface to allow sufficient clearance for these connectors. Double this clearance is needed if you want to plug/disconnect connectors without disassembly.

Power and Signal

The Onyx Fire II board can plug right into an existing Dynamixel TTL based serial bus for power and communications, as long as the supplied power is between 8V and 18V and the power source can supply sufficient amperage for the switched load attached to the board. The Onyx Fire II has two beige connectors for this bus, labeled DXL1 and DXL2, allowing daisy-chaining. The total number of devices in any one daisy chain should be kept small; a system with many servos, switch boards, or other devices should be connected with a hub that is in turn connected directly to a power source to avoid too much power loss and signal degradation.

Power can also be supplied through a RC-style servo connection, to the pin header labeled SVO at the top right of the board. The leftmost pin, marked with a white bar on the circuit board, is ground, and should be connected to the black or brown wire on an incoming servo wire. The center pin is power, and should be connected to the red wire. The rightmost pin is signal, and should be connected to the white or yellow wire.

The cable that comes with the Onyx Fire II has two different ends, which makes it suitable for connecting either to a Dynamixel bus, or to a RC servo controller. When using the three-pin header of the cable, it is important to connect the black (ground) wire to the leftmost (GND) pin of the power connector, which is marked with a line (bar) next to the pin.



Similarly, when connecting the three-pin header to another board or controller, make sure to line up the black (ground) wire to ground (or negative.)

The SVO, DXL1 and DXL2 connectors are connected on the Onyx Fire II, so which one of these connectors you use is a matter of preference and convenience.

When you first turn on power to the Onyx Fire II, pay close attention to the message LEDs (blue and green) on the left of the board. If you do not immediately see a blue LED light up, immediately turn off power and check connections to avoid damage to the Onyx Fire II. It is not designed to withstand periods of reverse polarity power connections. To help with correct polarity connections, “ground” pins (that usually connect to black wires,) are marked with a “bar” or “line” indicating the pin on the Onyx Fire II circuit board.

Load

The equipment you want to switch on and off (called the “load”) is connected to the two green screw terminals to the right of the circuit board. These screw terminals are labeled “J1” and “J2” to correspond with load 1 and load 2, and are also referred to as “switch 1” and “switch 2,” or “SW1” and “SW2” for short. The positive output of the terminals is permanently connected to the power source supplied to the board; the negative output of the terminal is switched on and off under user control.

Equipment that has polarity, such as LED lights or motors that should spin a particular direction, should be connected with red (positive, anode) wire to the top screw (closest to the white DXL headers) and black (negative, cathode) wire to the bottom screw (closest to the “slide” pin header.)

The load connectors have kickback protection diodes and capacitors that safely dissipate the inductive “kickback” from relays, motors, and solenoids. Additional external kickback protection should not be necessary in most cases.

By default, the Onyx Fire II applies a PWM duty cycle to the switches controlling the loads, so the full power available through the power input will not be seen by the controlled load. This PWM duty cycle can be adjusted, up to “full on,” using serial commands.

If the power source used is not sufficient to power the load attached to the switches, activating the switches may make the voltage sag, which may in turn lock up or reset the controller on the Onyx Fire II board. Cutting power to the board may be the only way to recover from this condition. Make sure the amperage capacity of your power source is sufficient for the load, and that the amperage draw of your load is not too high for the Onyx Fire II.

Control / Signal

Depending on configuration, the Onyx Fire II can be actuated by TTL serial commands, by RC PWM pulses, or by pulling pins CFG1 or CFG2 low using a microcontroller or switch.

The serial interface is most powerful, and allows re-configuration of the board from a host computer. Because it is single-ended, “TXD” and “RXD” are tied together with current limiting resistors, and the device that sends data on the bus must temporarily ignore the data it sends, as it will also be available on the receiver side. The Onyx Fire II is compatible with Dynamixel interface equipment, including the USB2DXL from Robotis, the USB2AX from Xevel, and the OpenCM 9.04 controller from Robotis. It is also compatible with the Version 1 Dynamixel protocol implemented by the Dynamixel Arduino library. For more information about the serial protocol, see below.

With no configuration headers installed, the Onyx Fire II starts up in TTL serial mode. See Serial Commands for more information.

With one header installed on the CFG3 jumper pin, the Onyx Fire II starts up in pin-triggered mode. See Logic Level Commands for more information.

With one header installed on the CFG2 jumper pin, the Onyx Fire II starts up in RC PWM mode. See PWM Commands for more information.

With one header installed on the CFG1 jumper pin, the Onyx Fire II starts up in TTL serial mode and slightly modifies the serial protocol to allow separate activation of the J1 and J2 switches. See Serial Commands for more information.

Slide

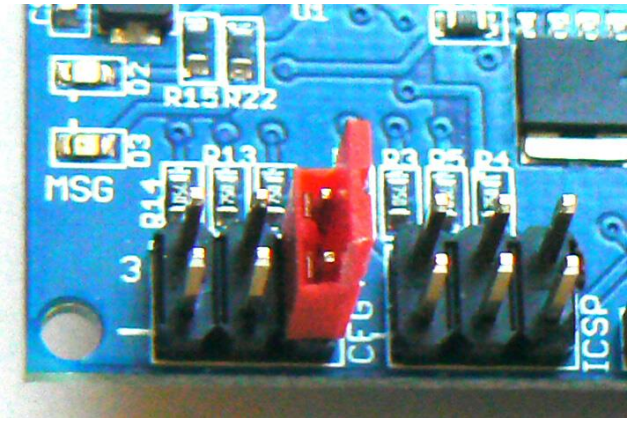
An optional Onyx Slide sensor can be connected to the SLIDE pin header in the lower right of the board. This allows the Onyx Fire II to actuate loads for a single cycle of the operation of a mechanism, and then automatically turn off until actuated again. The center pin of SLIDE is ground (black wire.) The left pin (closest to the ICSP header) is current-limited power (red wire.) The right pin (closest to the mounting hole) is an active-low sensor input (white wire) with a pull-up resistor on the Onyx Fire II board.

Control

Serial Commands

In serial command mode, with no jumpers installed on the CFG header, the Onyx Fire II will blink the blue LED on the left side when turned on. The Onyx Fire II will then make sure the load switches are turned off, and await commands on the serial bus. The default baud rate used on the serial bus is 1 Mbps, and this can be adjusted to many popular baud rates using a configuration utility such as Dynamixel Wizard from Robotis, or a small amount of code running on a microcontroller. For supported baud rates, see the configuration for register 0x04 (Baud Rate.) The default ID of the Onyx Fire II is 0x00, but this can be changed by writing to register 0x03 (ID.)

When the CFG1 jumper is installed, the Onyx Fire II starts up in serial mode, with “separate fire” enabled. It indicates this mode by turning on the blue LED and blinking the green LED during start-up.



Packet Format

A packet sent to the Onyx Fire II has the following format, which is compatible with Version 1 of the Dynamixel serial protocol:

```
0xFF 0xFF <ID> <Length> <Instruction> <Data1> … <DataN> <Checksum>
```

The Onyx Fire II will pay attention to packets where the <ID> matches the configured ID of the board (default 0x00,) or where the <ID> is the broadcast address 0xFE.

The <Length> field contains the number of bytes that follow the length field in the packet, up to and including the <Checksum>.

The <Instruction> is the command in the packet, which is generally either 0x02 for "read registers" or 0x03 for "write registers." The Onyx Fire II recognizes the following commands:

Command	Name	Meaning
0x01	PING	Return a status packet
0x02	READ	Return contents of registers
0x03	WRITE	Update contents of registers
0x06	RESET	Reset registers to defaults

The PING command has no additional parameters, so the <Length> field should be 2.

The READ command has two additional parameters, the start register to read from, and the number of bytes to return, so the <Length> field should be 4.

The WRITE command has one additional parameter, the start register to write to, plus some number of bytes to write, so the <Length> field should be 3 plus the number of written data bytes.

The RESET command resets the Onyx Fire II to the default settings. You will need to power cycle the board twice after you have issued this command to make it fully reinitialize itself. Remember that, after a RESET, the default ID is 0x00 and the default baud rate is 1 Mbps.

The <Checksum> is calculated as the sum of all bytes starting with the <ID> and ending with the last data byte, and then invert all bits. For example, to calculate the checksum for a command to read register 0x04 (baud rate,) here is some example code:

```
//  construct the packet you want to send
unsigned char packet[8] = {
    0xFF,   //  attention
    0xFF,
    0x00,   //  ID
    0x04,   //  length: 4 bytes follow this
    0x02,   //  command: read
    0x04,   //  data1: register start
    0x01,   //  data2: number of bytes
    0x00    //  space for checksum
};

//  calculate the checksum
unsigned char cs = 0x00;
unsigned char len = 3 + packet[3];
for (unsigned char i = 2; i != len; ++i) {
  cs = cs + packet[i];
}
packet[len] = ~cs;  //  write checksum into packet

// call some function to send the bytes on the TTL bus
send_bytes(packet, len+1);
```

If using a pre-existing Dynamixel interface library, the checksum and packet format is made for you in the library.

The response packet will have the following format:

```
0xff 0xff <ID> <Length> <Status> <Data1> .. <DataN> <Checksum>
```

When responding to a command other than READ, the <Length> field will have the value 2, and there will be no <Data> bytes. When responding to a READ command, the <Length> field will have the value 2, plus the number of bytes read. The checksum is calculated in the same way as indicated above.

The <Status> value is OR-ed together bits indicating errors that have occurred since a previous status packet was generated. The values of these bit flags can be found in the table for the ALARMLED register.

Registers

Communication with the Onyx Fire II is mainly through writing values to registers of the controller. For example, if you want to turn on the message LED of the Onyx Fire II, you write the value 0x01 to the LED register (register number 0x19.)

In firmware version 2, the Onyx Fire II uses the following registers:

Register	Name	Description
0x00	MODELLO	Low part of model number
0x01	MODELHI	High part of model number
0x02	FIRMWAREVERSION	Firmware version
0x03	ID	ID recognized on TTL bus
0x04	BAUDRATE	Communication speed on TTL bus
0x05	RETURNDELAY	Delay before returning response to command
0x06	SW1CYCLELO	Low byte of SW1 PWM duty cycle
0x07	SW1CYCLEHI	High byte of SW1 PWM duty cycle
0x08	SW2CYCLELO	Low byte of SW2 PWM duty cycle
0x09	SW2CYCLEHI	High byte of SW2 PWM duty cycle
0x0E	TIMEOUTINIT	Timeout for first activation cycle
0x0F	TIMEOUTNEXT	Timeout for successive activation cycles
0x10	STATUSRETURNLEVEL	Which commands to acknowledge
0x11	ALARMLED	Which conditions to turn on the LED for
0x12	ALARMSHUTDOWN	Which conditions to disable power for
0x18	ENABLE	Set to 1 to enable commands driving the outputs
0x19	LED	Set to 1 to turn on the message LED
0x1E	NUMSHOTSLO	Low byte of how many cycles to turn on SW1
0x1F	NUMSHOTSHI	High byte of how many cycles to turn on SW1
0x20	SEPSHOTSLO	Low byte of how many cycles to turn on SW2
0x21	SEPSHOTSHI	High byte of how many cycles to turn on SW2
0x2F	LOCK	Set to 1 to prevent writing to the EEPROM area

The registers shaded in gray (between 0x00 and 0x17) are stored in EEPROM so their values persist when power is removed. Those registers may also affect the operation of the Onyx Fire II in RC PWM or Logic Level Trigger mode. If the LOCK register (0x2F) is set to a non-zero value, WRITE commands that would modify the EEPROM registers are ignored. The non-EEPROM registers (including the ENABLE register) will have the value 0x00 when power is turned on.

To quickly actuate the J1 output (and J2 output unless in “separate shots” mode) write register 0x18 to the value 0x01 to turn on control, and then write register 0x1E to 0x01 to activate the switch for one cycle. See below about the timeout used instead of “cycles” when not using the optional Onyx Sense sensor.

MODELLO, MODELHI

These registers identify the Onyx Fire II as an Onyx Fire II, and not some other device, on the TTL bus. The value of these registers is 0x01, 0x01.

FIRMWAREVERSION

This register identifies the version of the firmware loaded into the Onyx Fire II. A higher version means a newer firmware, presumably with bug fixes or enhancements compared to previous versions. The list of firmware versions can be found on the web site http://onyxrobots.com/

ID

This register tells the Onyx Fire II which ID on the serial bus to pay attention to. The default value is 0x00 (ID 0) and the allowed range is 0x00 to 0xFC. When sending a packet to the Onyx Fire II, use this ID in the <ID> field. When the Onyx Fire II responds with a status packet, it uses this ID for the <ID> field. Changes to this register take effect immediately.

BAUDRATE

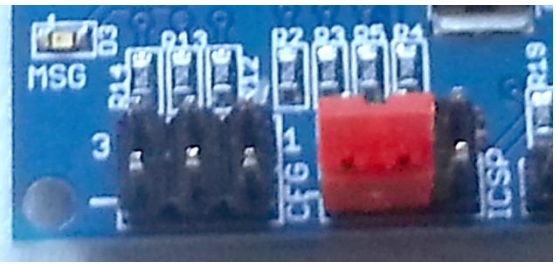
This register tells the Onyx Fire II which speed to use for communications on the TTL bus. Common values are found in this table:

Value	Speed
0x00	2000000 bps
0x01	1000000 bps
0x08	230400 bps
0x10	115200 bps
0x22	57600 bps
0x33	38400 bps
0xCF	9600 bps

1 Mbps is easy to generate from most microcontrollers such as Arduino, OpenCM 9.04, or the Teensy, and is also standard on Dynamixel interface devices such as USB2Dynamixel or USB2AX. Some cheap PC serial adapters are not capable of running at 1 Mbps, instead falling back to 921 kbps or 1.15 Mbps, which will not work correctly.

The Onyx Fire II uses 8 bits, no parity, 1 start bit, and 1 stop bit, treats a “high” logic level as a 1 value, and sends/receives the lowest-valued bit first. This is the default and standard on almost any modern device using TTL serial communications, including the Arduino, the OpenCM 9.04, and most FTDI-style serial USB adapters that provide pin headers, rather than a DB-9 style connector.

If you cannot generate 1 Mbps from your computer or controller, and need to configure the Onyx Fire II registers or control it using serial commands, you can install the jumper on the GND and MOSI pins of the ICSP header, as seen below:



Note that this jumper is installed sideways, as opposed to the CFG jumpers which are installed vertically.

Installing this header will force the Onyx Fire II to use 57600 bps communication rate. This is usually easy to generate from any computer or microcontroller. Note that you can write another serial rate to the BAUDRATE register, and if you remove this header and re-start the board, that change will take effect.

RETURNDELAY

When the Onyx Fire II has processed a command, it delays some amount of time before sending the acknowledgement or response packet back. This gives the transmitting controller time to turn around any buffers or drivers used to manage the bus, and time to disable its own transmitter to avoid multiple chips driving the bus at the same time. The default value is 0x05, and the unit is two microseconds, so the default time after processing is 10 microseconds.

SW1CYCLELO, SW1CYCLEHI

These two registers control the duty cycle of the J1 switch. The valid range is 0x000 through 0xFFF. The default value is 0x800 (50% duty cycle) and is stored as 0x00, 0x08 (little endian order.) The value 0xFF would be stored as 0xFF, 0xF. Setting the duty cycle controls how much power will be output through the switch when it is activated. Because of timer resolution limitations, small adjustments (less than +/- 0x10) may not result in any visible duty cycle change.

SW2CYCLELO, SW2CYCLEHI

These two registers control the duty cycle of the J2 switch. The valid range is 0x000 through 0xFFF. The default value is 0x800 (50% duty cycle) and is stored as 0x00, 0x08 (little endian order.) The value 0xFF would be stored as 0xFF, 0xF. Setting the duty cycle controls how much power will be output through the switch when it is activated. Because of timer resolution limitations, small adjustments (less than +/- 0x10) may not result in any visible duty cycle change.

TIMEOUTINIT

When a switch is actuated, it will stay actuated until the SENSE port completes one cycle, signaling that the activated mechanism is complete. If nothing is connected to the SENSE port, the switch activation value will start counting down after an amount of time controlled by TIMEOUTINIT. The allowed range is 0x00 through 0xFF, and the calculated timeout is (TIMEOUTINIT + 1) * 4.096 milliseconds.

TIMEOUTNEXT

After the first timeout decrement of the switch control value, the control value will be repeatedly decremented at an interval controlled by this register. The unit is about 4.1 milliseconds. The allowed range is 0x00 through 0xFF, and the calculated timeout is (TIMEOUTNEXT + 1) * 4.096 milliseconds.

STATUSRETURNLEVEL

This register controls when the Onyx Fire II board will send a return packet on the TTL bus. The Onyx Fire II will never send a return packet in response to a command to the broadcast id (0xFE) even if it recognizes and acts on such a command. The default status return level is 0x02.

Return Level	Action
0x00	Only respond to PING and RESET

0x01	Respond to PING, RESET, and READ
0x02	Respond to PING, RESET, READ, and WRITE

ALARMLED

The value of this register controls what status conditions will turn on the message LED. The value of this register is a bitwise OR of the conditions that should turn on the message LED. The default is all the conditions ON.

Value	Status
0x08	Range error – the values in a command are outside allowed range
0x10	Checksum error – the checksum in a received packet was incorrect
0x40	Instruction error – an instruction was received that was not recognized

ALARMSHUTDOWN

The value of this register controls what status conditions will automatically disable the ENABLE register. The value of this register is a bitwise OR of the conditions that should auto-disable the ENABLE register. For values, see the ALARMLED register. The default is no condition disables the ENABLE register.

ENABLE

This register must be written as 0x01 before the Onyx Fire II will activate the switches. The default on power-on is 0x00. This allows the controller to safely detect and control the Onyx Fire II after power-on. In RC PWM and Logic Level Trigger mode, this register has no effect.

LED

When this register is non-zero, the message LED will be turned on.

NUMSHOTSLO, NUMSHOTSHI

When written to a non-zero value, this register will trigger the output switches. When in single-trigger mode, both switches will be triggered. When in separate-fire mode (CFG1 is jumpered,) only J1 will be triggered. The switch will stay on until the sensor pin has cycled on iteration. If no sensor is connected, or the mechanism cycles too slowly, the switch will be on for a duration of (TIMEOUTINIT + TIMEOUTNEXT x (NUMSHOTS-1)) x 4.1 milliseconds. If you want the switch to be on for an extended duration, keep setting this register to a non-zero value about every half second (or whatever you have configured in TIMEOUTINIT and TIMEOUTNEXT.)

SEPSHOTSLO, SEPSHOTSHI

When CFG1 is jumpered (separate-fire mode,) this register controls the triggering of J2 similar to how NUMSHOTSLO,NUMSHOTSHI controls the triggering of J1. When CFG1 is not jumpered, J2 is triggered at the same time as J1, and both are controlled by NUMSHOTS.

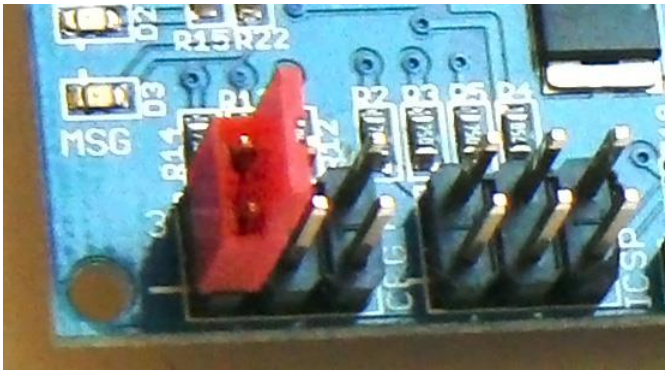
LOCK

When this register is set to a non-zero value, WRITE commands that include the EEPROM register range (0x00 through 0x17) will not be allowed. This can be a safety feature to prevent inadvertent triggering of

configuration changes that may be hard to recover from, such as erroneous commands changing the ID or BAUDRATE registers.

Logic Level Commands

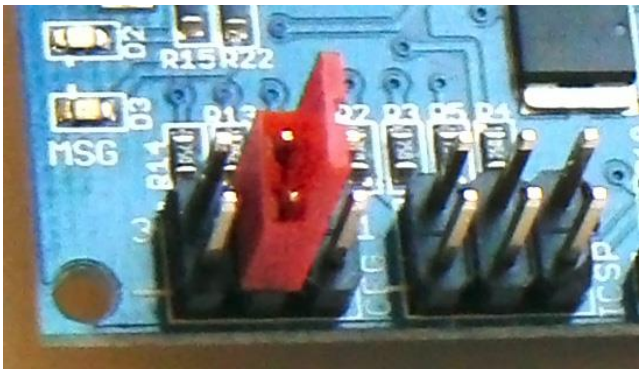
When the CFG3 pin has a header installed (pulled to ground) during start-up, the Onyx Fire II starts up in logic-level-triggered mode. You can tell it does this by watching the blue and green message LEDs during start-up; the green and blue LEDs should light up at the same time.



In Logic Level Commands mode, the pins CFG1 and CFG2 are trigger inputs. Pulling either pin low will activate the corresponding output switch (J1 and J2.) If activated by a short pulse, the output switches will stay active for one slide sensor cycle or about 819 milliseconds. If held low for a longer time, the switches will keep active until released and a sensor cycle completes or the timeout expires.

PWM Commands

The signal pin on SVO and DXL connectors is normally low. When the pin is high for a time between 820 and 1250 microseconds, corresponding to pushing a remote control joystick one way, the J1 switch is activated. When the pin is high for a time between 1750 and 2180 microseconds, corresponding to pushing a remote control joystick the other way, the J2 switch is activated. If the pulse is shorter than 820 microseconds, or longer than 2180 microseconds, the switches will not activate, to prevent accidental triggering from improper connections.

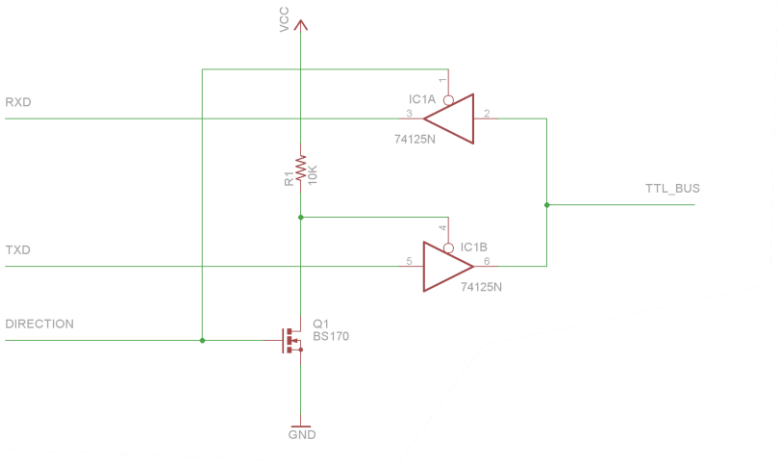


If activated by a short pulse, the output switches will stay active for one slide sensor cycle or about 819 milliseconds. If held low for a longer time, the switches will keep active until released and a sensor cycle completes or the timeout expires.

Serial Connection Troubleshooting

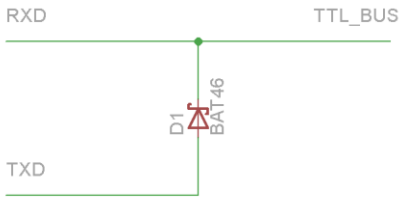
To configure Onyx Fire II, and to get the most control out of it, you need to read and write registers on the single-ended TTL serial bus, as described above. It's important that the devices on the bus take turns to drive the bus. A TXD pin on a typical microcontroller or USB serial adapter will stay high with low impedance when it is not transmitting any data, which will make it impossible for the Onyx Fire II to send data back. There are multiple ways to interface to a single-ended bus like this.

Easiest is to use a device already built for this. This includes the USB2Dynamixel USB adapter by Robotis, the USB2AX USB adapter by Xevel, or the OpenCL 9.04 microcontroller by Robotis. These adapters either automatically release the bus when not transmitting, or have a special chip that controls the direction of the bus. A typical circuit to interface this way includes the 74HCT125 tri-state buffer. A schematic for this connection can be found below.



When transmitting data, set the direction control pin high, send the data as usual, wait for the last byte to be completely sent, and then set the direction pin low. The Dynamixel interface libraries that come with the OpenCM 9.04 board does this for you.

Another option is to insulate the transmit pin from the bus with a diode. This will let the transmit circuitry pull the bus low when it is transmitting data, but won't force the bus high when it's not transmitting. This works because the Onyx Fire II pulls the bus high using a very weak pull-up when it is itself not transmitting. A schematic for such a connection can be found below.

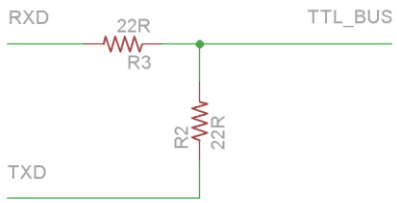


This circuit is appropriate for FTDI and other similar USB-to-TTL-serial adapters used from personal computers. The draw-back of this interface is that the "receive" part of the adapter will receive all the bytes sent by the "send" part, so software needs to expect and discard this data, before it receives the

data from the other end. A Schottky-type small-signal diode with a low voltage drop should be used, such as the BAT46.

The final option, which is appropriate for microcontrollers where you have full control of the serial interfacing circuitry, is to disable the transmitter whenever you are not transmitting, and disable the receiver whenever you are transmitting. You can then simply tie the TXD and RXD pins together, and connect them to the bus, optionally through a low-value resistor (between 10 and 100 Ohms) to prevent over-current in case of an electrical short.

A schematic for such a connection can be found below.



This circuit, as well as the diode-based one, works with the sample Arduino sketch available in the appendinx and on the onyxrobots.com website. This code will work even just tying the TXD and RXD to the TTL bus.

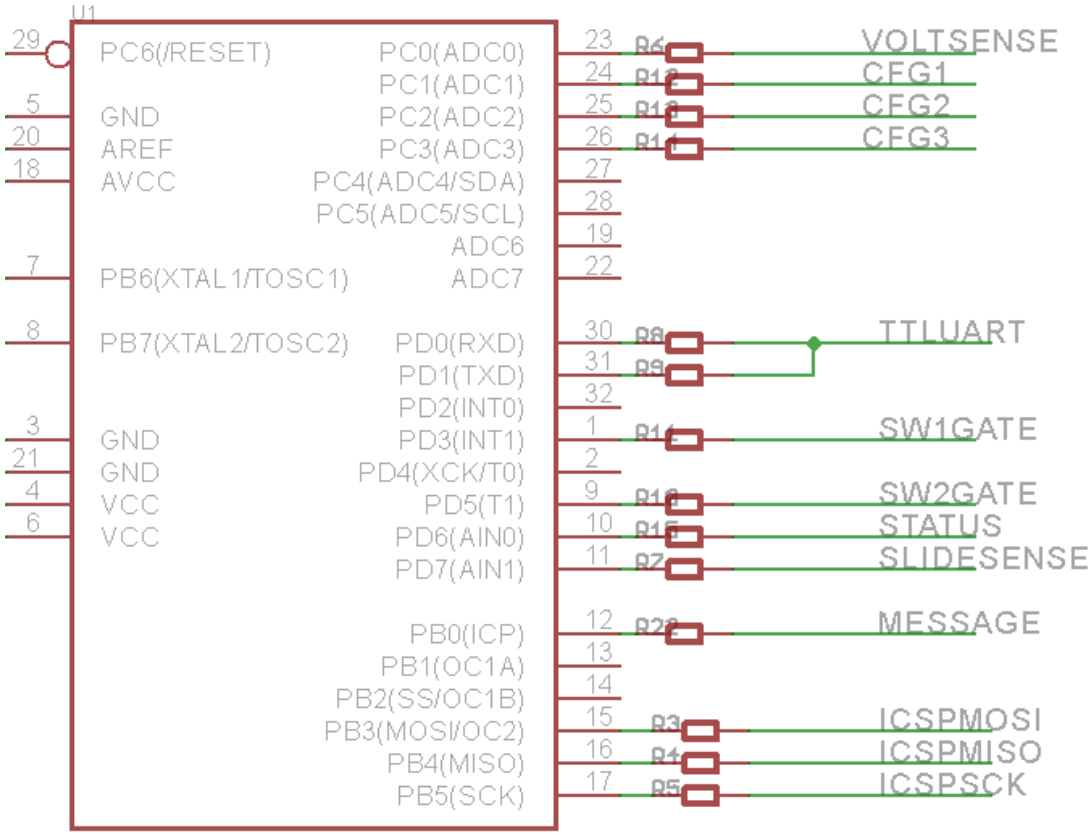
Hackable Firmware

The microcontroller in the Onyx Fire II is an Atmel Atmega328, which is very similar to the controller used by the Arduino Uno development board. Using a “programmer” device connected to the ICSP header, you can re-program the Onyx Fire II board to run software that you develop using the Arduino IDE, the Atmel Studio software, or the avr-gcc command-line tool. The Onyx Fire II does not implement a “bootloader” over the serial bus, so a dedicated programmer is required.

Note that all the functions documented in this manual will go away if you choose to load your own software into the controller, and Onyx Robots cannot take responsibility for any mishap that may happen from your own software – doing this voids the warranty!

Only do this if you feel confident in your ability to develop embedded software to run in the Onyx Fire II environment. If you want to restore the original Onyx Fire II firmware, Onyx Robots can re-load the original firmware for a fee; see contact information below for more information.

Here is a schematic that shows the connections between the Atmeta328 processor and the various peripherals:



Contact Information

For the latest version of this manual, other Onyx Robots products, and general information, please visit our website:

http://onyxrobots.com

For customer service, please file a request on the web site, or send e-mail to:

helpme@onyxrobots.com

Warranty and Limitations of Liability

We've done a lot of work to develop a high quality product, and run tests on every item before it leaves the factory. Despite this, a small percentage of early failures are impossible to avoid in modern electronics, unless you have the budget and resources of NASA or the US Air Force. We do not, and neither do our customers.

The Onyx Fire II board is warranted against defects in materials and workmanship for a period of 90 days after purchase. Should the board fail to operate as documented in this manual when correctly installed and configured by the user, we will gladly replace it with a working version. All you pay is postage to send it to our service center; we pay for any repairs or replacement and shipping back to you. Please contact us using one of the methods above and we will promptly arrange for replacement or refund.

The above warranty does not hold in case we've made an embarrassing and impossible-to-fulfill typo in the manual, in which case your sole recourse is to shame us in public forums until and unless we are told by such a typo and correct it in an updated version of the manual.

The Onyx Fire II board is an electronic component that is intended to be integrated into a larger system by the end user. As such, it is impossible for the manufacturer to foresee all possible problems and dangers that could arise in the end system. The purchaser should be capable of making appropriate engineering and safety considerations to use the Onyx Fire II component. In all engineering, there is danger of improper operation, fire, electrocution, and lost hair, and the purchaser must accept and control for such danger. In other words, we do our best to provide a high-quality product that operates as per this manual; if you should use this product to shoot eggs at neighbors, trigger flammable rockets, or build a robot army that rises up and exterminates humanity, those are your actions, not ours, and you will have to live with yourself afterwards.

If you believe you do not possess the appropriate level of engineering and safety awareness to integrate the Onyx Fire II board in your system, we will happily refund your purchase price if you return the board and its accessories undamaged in its original packaging to us within 30 days of purchase. Please contact us at the above locations if you need to arrange for this.

Appendix: Arduino Functions for Reading and Writing Registers

```
//  The default ID of the Onyx Fire II is 0.
//  This can be changed by setting register 0x03, ID
unsigned char id = 0;

void setup() {
  //  soft pull-up on RXD
  pinMode(0, INPUT);
  digitalWrite(0, HIGH);

  //  setup talking to the Onyx Fire II
  Serial.begin(1000000);
  //  turn off the transmitter so it doesn't pull the receiver high
  UCSR0B = (UCSR0B | (1 << RXEN0)) & ~(1 << TXEN0);

  //  blink the Arduino LED to show something's happening
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);
}
```

```
//  You can use write_packet() directly if you want; this function
//  writes a packet composed of two separate buffers to the TTL bus
//  and adds the checksum. This function is also used by the set_reg()
//  and get_reg() functions.
//
void write_packet(unsigned char const *b1, unsigned char l1, unsigned char
const *b2, unsigned char const l2) {
  unsigned char cs = 0;
  //  Assuming you have tied TXD and RXD together, you want to
  //  turn off the receiver while sending, such that you don't have
  //  to decode the bytes that you just sent.
  //  For maximum performance, disable interrupts while sending
  //  the packet (this avoid interference with the "underflow
  //  interrupt" in the Arduino Serial library.)
  cli();
  UCSR0B = (UCSR0B | (1 << TXEN0)) & ~(1 << RXEN0);
  UCSR0A |= (1 << TXC0);
  for (unsigned char ch = 0; ch != l1; ++ch) {
    UDR0 = b1[ch];
    if (ch >= 2) { // don't checksum the initial 0xff 0xff
      cs += b1[ch];
    }
    while (!(UCSR0A & (1 << TXC0))) {
      ;  //  wait for complete
    }
    UCSR0A |= (1 << TXC0);
  }
  for (unsigned char ch = 0; ch != l2; ++ch) {
    UDR0 = b2[ch];
    cs += b2[ch];
    while (!(UCSR0A & (1 << TXC0))) {
      ;  //  wait for complete
    }
    UCSR0A |= (1 << TXC0);
  }
  UDR0 = ~cs;
  while (!(UCSR0A & (1 << TXC0))) {
    ;  //  wait for complete
  }
  UCSR0A |= (1 << TXC0);
  //  turn off the transmitter so it doesn't pull the receiver high
  UCSR0B = (UCSR0B | (1 << RXEN0)) & ~(1 << TXEN0);
  sei();
}

//  flush_serial() drains the serial input buffer.
//
void flush_serial() {
  while (Serial.available()) {
```

```
    Serial.read(); // throw away the data
  }
}

//  dump() is a convenience function for receiving and ignoring
//  whatever response is sent by the Onyx Fire II.
void dump() {
  delay(1);
  flush_serial();
}

//  set_reg() sends a packet to update one or more registers, starting
//  at register ‘reg’ and incrementing up to ‘reg’+’n’-1, with the
//  data pointed at by ‘data’.
//
void set_reg(unsigned char reg, unsigned char n, unsigned char const *data) {
  unsigned char cmdbuf[6] = {
    0xff,
    0xff,
    id,
    n + 3, // cmd, reg, checksum
    3,    //  WRITE
    reg
  };
  write_packet(cmdbuf, 6, data, n);
  dump();  //  in case response level is 2
}

//  get_reg() reads one or more registers, starting at ‘reg’ up to
//  ‘reg’+’n’-1, and puts the received data into the buffer pointed
//  to by ‘obuf.’ You have to make sure that buffer is at least ‘n’
//  bytes in size.
//
bool get_reg(unsigned char reg, unsigned char n, unsigned char *obuf) {
  //  send the "read registers" packet
  unsigned char cmdbuf[7] = {
    0xff,
    0xff,
    id,
    4,    // cmd, reg, len, checksum
    2,    //  READ
    reg,
    n
  };
  write_packet(cmdbuf, 7, 0, 0);
  //  wait for returned packet
  delay(1);
  //  decode the returned packet
  if (Serial.read() != 0xff
    || Serial.read() != 0xff
```

```
    || Serial.read() != id
    || Serial.read() != n + 2) {  //  status, checksum
    flush_serial();
    return false;
  }
  unsigned char calccs = Serial.read() + (n+2) + id;
  for (unsigned char in = 0; in != n; ++in) {
    obuf[in] = Serial.read();
    calccs += obuf[in];
  }
  calccs = ~calccs;
  unsigned char cs = Serial.read();
  if (cs != calccs) {
    //  bad checksum
    return false;
  }
  //  got the data
  return true;
}


//  Example program that blinks the Onyx Fire II blue LED,
//  and verifies that the right value is stored in register
//  25 (to make sure the board is there.)
//

unsigned char val1 = 1, val2 = 0;

void loop() {
  delay(500);
  set_reg(25, 1, &val1);
  delay(100);
  bool ok = get_reg(25, 1, &val2);
  if (!ok) {
    digitalWrite(13, HIGH);
    while (true) {
      //  spin forever -- error!
    }
  }
  val1 = 1 - val1;
}
```

Version History

2014-04-28	Firmware 2	Initial Version