EVShield Interface Specifications

Power Specs:
EVShield can be powered from external power supply.
Max Power Rating: 10.5 Volts DC
Minimum 6.6 Volts DC needed to run NXT/EV3 motors or Servos.

Motor Ports Specs:
EVShield has 4 NXT/EV3 Motor Ports. The ports are rated as follows:
Output Rating: 10.5 V DC, 1 amp, thermally protected.

Sensor Ports Specs:
EVShield has 4 NXT/EV3 Sensor Ports. Each sensor port has two power pins, For power delivery, the ports are rated as follows:
Power Source 1 (pin 1):
Voltage: 9 V
Max current rating: 50mA continuous current per port.
Power 2: 5 V (VCC pin – pin 4)
Max current rating: 100mA per port
Max current not to exceed: 200mA from combined 4 ports.

I2C Ports Specs:
EVShield has 1 I2C Port with male pin headers.
I2C Voltage: 5 Volts
Max current rating: 100mA per port.
Max current not to exceed: 200mA from combined 2 ports.

Servo Ports Specs:
EVShield has 1 set of 6 Servo ports.
Servo Voltage: 5 V
Max Current rating: 1.5 amp peak.

Arduino Software resources Used by EVShield

Timer 2 from Arduino:
EVShield uses Timer 2 from Arduino resources for port resetting if an error occurs. Note that, Tone Library also uses Timer 2. If you are using Tone generation, EVShield ports may generate errors for incorrect data. It is not recommended to use Tone generation when using the EVShield.

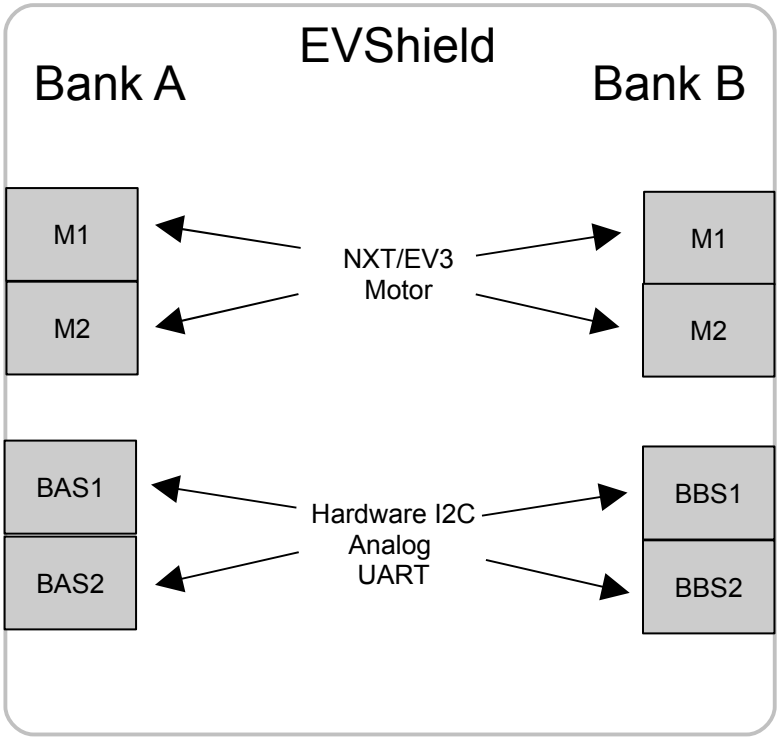
Arduino Hardware resources Used by EVShield

D = Duemilanove, U = Uno, L = Leonardo

Board	PIN	Purpose
D/U	A4	SDA
D/U	A5	SCL
D/U	Di3	Servo 3
D/U/L	Di5	Servo 5
D/U/L	Di6	Servo 6
D/U/L	Di9	Servo 9
D/U/L	Di10	Servo 10
D/U/L	Di11	Servo 11
L	Di13	Servo 3
D/U/L	SDA	SDA
D/U/L	SCL	SCL

Port Features of EVShield

Picture below illustrates the supported devices on prominent Ports.

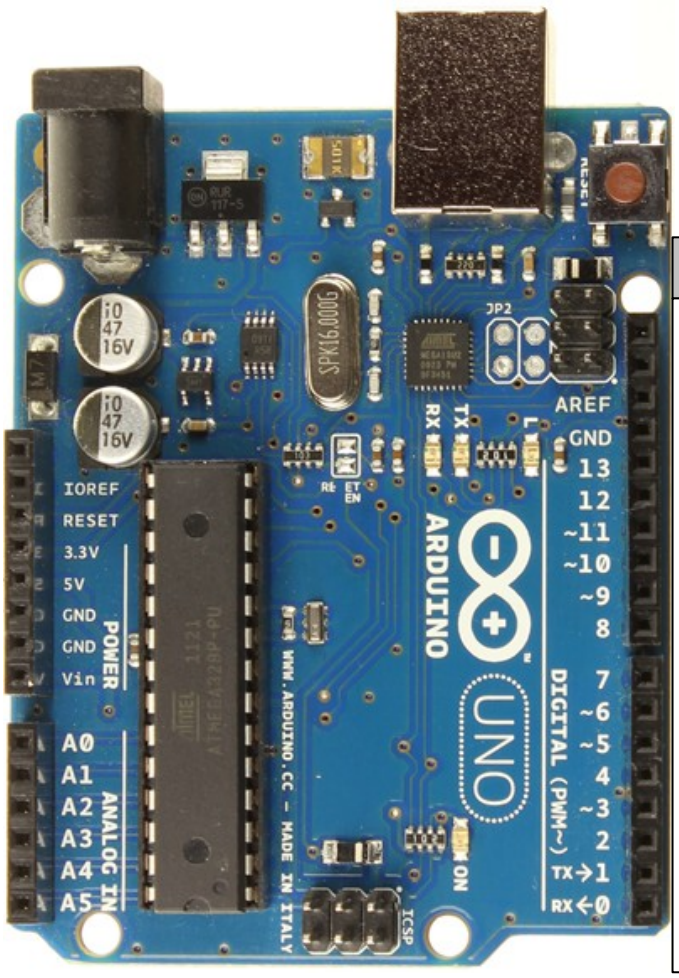


Arduino Uno with Pin mappings for EVShield

Green Labels: These pins could be used for other purposes without affecting any of EVShield functionality.

Red Labels: These pins are required for EVShield functionality. These are used for Hardware I2C, and if you need to use for another I2C device, you may share them as long as that device is compliant to I2C specs and has non-conflicting address.

Pin Usage	
Available (shared)	RESET
Available (shared)	3.3V
Available (shared)	5V
Available (shared)	GND
Available (shared)	GND
Available (shared)	Vin
Available(unconnected)	A0
Available(unconnected)	A1
Available(unconnected)	A2
Available(unconnected)	A3
SDA (UNO)	A4
SCL (UNO)	A5



Pin Usage	
SCL	SCL
SDA	SDA
AREF	Available(unconnected)
GND	Available (shared)
13	Routed to Servo Header 3 (Leonardo)
12	Available(unconnected)
11	Routed to Servo Header 11
10	Routed to Servo Header 10
9	Routed to Servo Header 9
8	Available(unconnected)
7	Available(unconnected)
6	Routed to Servo Header 6
5	Routed to Servo Header 5
4	Available(unconnected)
3	Routed to Servo Header 3 (UNO)
2	Available(unconnected)
1	Available(unconnected)
0	Available(unconnected)

Pros and cons of Hardware I2C:

Hardware I2C: This I2C protocol provides high speed communication.
NXT Ultrasonic sensor will not work with EVShield.

All mindsensors' I2C devices (and most other I2C devices) can be used with this protocol (on any of the sensor ports).

You can connect Port Splitter(s) on any of these ports, and connect several more digital/I2C devices.

I2C Bus address

Factory Default Address of Bank-A: 0x34

Factory Default Address of Bank-B: 0x36

Changing the I2C Bus Addresses:
Not implemented at this time.

I2C Registers:

Each bank of EVShield appears as a set of registers as follows:

Register	Read	Write
0x00-0x07	Firmware version – *Vxxxx*	-
0x08-0x0f	Vendor Id – *openlctrn*	-
0x10-0x17	Device ID – EVShld	-
0x41		Command
	Motor 1 Write Parameters	
0x42	Encoder Target for Motor 1 (long) 0x42: Least Significant Byte 0x43: Byte 2 0x44: Byte 3 0x45: Most Significant Byte	Encoder Target of Motor 1 (long)
0x46	Speed for Motor 1 (byte)	Speed for Motor 1 (byte)
0x47	Time to run in seconds for Motor 1 (byte)	Time to run in seconds for Motor 1 (byte)
0x48	Command register B for Motor 1	Command register B for Motor 1 (set this value to 0 as this is for future use)
0x49	Command register A for Motor 1 (read the description below for details of this register).	Command register A for Motor 1 (read the description below for details of this register).
	Motor 2 Write Parameters	
0x4A	Encoder target for Motor 2 (long) 0x4A: Least Significant Byte 0x4B: Byte 2 0x4C: Byte 3 0x4D: Most Significant Byte	Encoder Value of Motor 2 (long)
0x4E	Speed for Motor 2 (byte)	Speed for Motor 2 (byte)
0x4F	Time to run in seconds for Motor 2 (byte)	Time to run in seconds for Motor 2 (byte)
0x50	Command register B for Motor 2	Command register B for Motor 2
0x51	Command register A for Motor 2 (read the	Command register A for Motor 2 (read the description

	description below for details of this register).	below for details of this register).
	Motor Read Parameters	
0x52	Encoder position of Motor 1 (long) 0x62: Least Significant Byte 0x63: Byte 2 0x64: Byte 3 0x65: Most Significant byte	-
0x56	Encoder position of Motor 2 (long) 0x66: Least Significant Byte 0x67: Byte 2 0x68: Byte 3 0x69: Most Significant Byte	-
0x5A	Status Motor 1 (byte). See section below for details of this register.	
0x5B	Status Motor 2 (byte). See section below for details of this register.	
0x5C	Tasks Running for Motor 1 (byte)	
0x5D	Tasks Running for Motor 2 (byte)	
	Registers for Advanced PID control	**Writing these registers has immediate effect on operation. These registers will be reset to factory default values upon power cycle.**
0x5E	Kp for Encoder Position Control (int) 0x7A: Least Significant Byte 0x7B: Most Significant Byte	Kp for Encoder Position Control (int)
0x60	Ki for Encoder Position Control (int) 0x7C: Least Significant Byte 0x7D: Most Significant Byte	Ki for Encoder Position Control (int)
0x62	Kd for Encoder Position Control (int) 0x7E: Least Significant Byte 0x7F: Most Significant Byte	Kd for Encoder Position Control (int)
0x64	Kp for Speed Control (int) 0x80: Least Significant Byte 0x81: Most Significant Byte	Kp for Speed Control (int)
0x66	Ki for Speed Control (int)	Ki for Speed Control (int)

	0x82: Least Significant Byte 0x83: Most Significant Byte	
0x68	Kd for Speed Control (int) 0x84: Least Significant Byte 0x85: Most Significant Byte	Kd for Speed Control (int)
0x6A	Pass Count – The PID controller repeatedly reads internal encoder ticks, this is the number of times the encoder ticks reading should be within tolerance. (default 5)	Pass Count – Higher Pass count gives more time to position internal encoder, thus providing better accuracy in positioning, but will take longer time. Change this only if you need different motor positioning speeds and accuracy. (For normal usage, default values will be OK).
0x6B	Tolerance – The Tolerance (in ticks) for encoder positioning. (default 80).	Tolerance – the accuracy you desire while positioning. Low number will position the encoders more accurately, but may take longer time. Change this only if you need different motor positioning speeds and accuracy. (For normal usage, default values will be OK).
0x6C	Check Sum	-
0x6E	Input Battery Voltage	-

I2C Registers for Sensor Modes:

See 'Sensor Type Modes' table below for defined sensor type modes.

	Read	Write
0x6F	Mode for Sensor 1	Mode for Sensor 1
0xA3	Mode for Sensor 2	Mode for Sensor 2

Sensor Type Modes:

These modes are to determine sensor type. **Do Not** confuse with EV3 sensor modes.

Mode	Value	Defined Variable for Arduino Code
None	0	SH_Type_NONE
Switch	1	SH_Type_SWITCH
Analog	2	SH_Type_ANALOG
Light Reflected	3	SH_Type_LIGHT_REFLECTED
Light Ambient	4	SH_Type_LIGHT_AMBIENT

I2C	9	SH_Type_I2C
Color Full	13	SH_Type_COLORFULL
Color Red	14	SH_Type_COLORRED
Color Green	15	SH_Type_COLORGREEN
Color Blue	16	SH_Type_COLORBLUE
Color None	17	SH_Type_COLORNONE
EV3 Switch	18	SH_Type_EV3_SWITCH
EV3	19	SH_Type_EV3

I2C Registers for LEGO Analog Sensors:

	Read	Write
0x70	Sensor 1 value (LSB)	-
0x71	Sensor 1 value (MSB)	-
0xA4	Sensor 2 value (LSB)	-
0xA5	Sensor 2 value (MSB)	-

I2C Registers for EV3 Sensors:

See 'EV3 Sensor Modes' table below for defined EV3 sensor modes.

	Read	Write
0x70	Sensor 1 Ready	-
0x71-0x80	Sensor 1 ID (up to 16 characters)	-
0x81	Sensor 1 Mode	Sensor 1 Mode
0x82	Sensor 1 Length	-
0x83-0xA2	Sensor 1 Data (up to 32 characters)	-
0xA4	Sensor 2 Ready	-
0xA5-0xB4	Sensor 2 ID (up to 16 characters)	-
0xB5	Sensor 2 Mode	Sensor 2 Mode
0xB6	Sensor 2 Length	-
0xB7-0xD6	Sensor 2 Data (up to 32 characters)	-

EV3 Sensor Modes:

These are EV3 sensor modes. **Do Not** confuse with sensor type modes. This table **does not** show the EV3 Touch Sensor modes.

Sensor/Mode	Value	Defined Variable for Arduino Code
Color	-	-
Reflected Light	0x00	MODE_Color_Reflected_Light
Ambient Light	0x01	MODE_Color_Ambient_Light
Measure Color	0x02	MODE_Color_MeasureColor
Gyro	-	-

Angle	0x00	MODE_Gyro_Angle
Rate	0x01	MODE_Gyro_Rate
Infrared	-	-
Proximity	0x00	MODE_Infrared_Proximity
Beacon	0x01	MODE_Infrared_Beacon
Remote	0x02	MODE_Infrared_Remote
Ultrasonic	-	-
Proximity in Centimeters	0x00	MODE_Sonar_CM
Proximity in Inches	0x01	MODE_Sonar_Inches
Presence	0x02	MODE_Sonar_Presence

I2C Registers for LEGO Color Sensor:

	Read	Write
0x70	Sensor 1 LEGO Color	-
0x71	Sensor 1 Red calibrated value	-
0x72	Sensor 1 Green calibrated value	-
0x73	Sensor 1 Blue calibrated value	-
0x74	Sensor 1 No Light calibrated value	-
0x75	Sensor 1 Red raw value	-
0x76	Sensor 1 Green raw value	-
0x77	Sensor 1 Blue raw value	-
0x78	Sensor 1 No Light raw value	-
0xA4	Sensor 2 LEGO Color	-
0xA5	Sensor 2 Red calibrated value	-
0xA6	Sensor 2 Green calibrated value	-
0xA7	Sensor 2 Blue calibrated value	-
0xA8	Sensor 2 No Light calibrated value	-
0xA9	Sensor 2 Red raw value	-
0xAA	Sensor 2 Green raw value	-
0xAB	Sensor 2 Blue raw value	-
0xAC	Sensor 2 No Light raw value	-

I2C Registers for Buttons and LEDs:

See 'Key Press Values' Section on page ? for defined button values.
Rear 2 LEDs are Bank specific.

	Read	Write
0xD7	Rear LED Red value	Front LED Red value
0xD8	Rear LED Green value	Front LED Green value
0xD9	Rear LED Blue value	Front LED Blue value
0xDA	KeyPress value corresponding to button	-

0xDB	Go Button press count	-
0xDC	Left Button press count	-
0xDD	Right Button press count	
0xDE	Front LED Red value	Back LED Red value
0xDF	Front LED Green value	Back LED Green value
0xE0	Front LED Blue value	Back LED Blue value

Key Press Values:

Button	Value	Defined Variable for Arduino Code
Go	2	BTN_GO
Left	1	BTN_LEFT
Right	4	BTN_RIGHT

Supported I2C Commands:

CMD	Hex	Description
R	0x52	Reset all Encoder values and motor parameters. (This does not reset the PID parameters).
S	0x53	Issue commands to both motors at the same time, for Synchronized starting of both motors.
		Motor Stopping Commands
a	0x61	Motor 1: Float while stopping.
b	0x62	Motor 2: Float while stopping.
c	0x63	Motor 1 and 2: Float while stopping.
A	0x41	Motor 1: Brake while stopping.
B	0x42	Motor 2: Brake while stopping.
C	0x43	Motor 1 & 2: Brake while stopping.
		Encoder Reset Commands
r	0x72	Motor 1: Reset Encoder to zero
s	0x73	Motor 2: Reset Encoder to zero

These commands are issued on command register (0x41).

Motor Command Register Explained

Each motor has two command registers (Register A and Register B). In current release Register B is reserved for future use and must be set to zero.

Bits in Register A should be set to 1 to avail functionality as described below.

Register Bit	Turn this bit to 1 for following functionality
Least significant	**Speed control of your motor.** The EVShield will honor speed values specified in the speed register for the

Register Bit	Turn this bit to 1 for following functionality
bit (bit 0)	respective motor.
Bit 1	**Ramp the speed up or down.** While starting the motor or changing speed, the EVShield will ramp up or ramp down the power to new value. If this bit is zero, the power changes are sudden, e.g. full power is applied to motors as they start.
Bit 2	**Relative change based on encoder values.** This is useful when Bit 3 is turned on, and in this case, the EVShield will make a relative movement from last seen Encoder position (it will add the new Encoder position to old position and move to that resulting position). Useful when turning by degrees or rotations. If this bit is 0, the Encoder positions are taken as absolute values.
Bit 3	**Encoder control of your motor.** The EVShield will honor Encoder values specified in Encoder position register for respective motor. If speed values are also specified, and speed bit is set on, motors will rotate to new encoder position with the specified speed.
Bit 4	**Brake or Float at the completion of motor movement.** If this bit is 1, motor will Brake at the completion, otherwise it will float.
Bit 5	**Encoder active feedback.** This bit is used when Encoder control is used. If this bit is set to 1 at the completion of motor movement, the EVShield will continue to hold the Encoder position (i.e. if motor is turned by external force, EVShield will try to restore it to last specified Encoder position). If this bit is zero, the motor may float.
Bit 6	**Timed control of your motor.** The EVShield will honor the time specified value in 'Time to run' register and run the motor for specified time. (If Time control bit as well as Encoder Control bit is on, the Timed control has precedence over encoder control).
Most significant bit (bit 7)	**GO.** When this bit is set to 1, all above bit values are brought into effect. This is useful to synchronized starting of both motors. As it takes some time to write each motor's register values. The I2C command 'S' can be used to change these bits to 1 for both motors at once.

Motor Status Register Explained

Each motor has one status register. Each bit in status register indicates various situations with the motor as explained below.

Register Bit	Value 1 indicates the situation is true
Least significant bit (bit 0)	**Speed Control is ON.** Motor is programmed to move at a fixed speed.
Bit 1	**Motor is Ramping (up or down).** If the Power ramp is enabled, this bit is 1 while the motor is ramping (while changing its speed).
Bit 2	**Motor is powered.** (This may not mean motor is moving.)
Bit 3	**Positional Control is ON.** The motor is either moving towards desired encoder position or holding its position.
Bit 4	**Motor is in Brake mode.** (0 value of this bit means motor is floating).
Bit 5	**Motor is overloaded.** If the external load prevents motor from achieving desired speed, this bit is set to 1.
Bit 6	**Motor is in timed mode.** This bit is 1 while the motor is programmed to move for given duration.
Most significant bit (bit 7)	**Motor is stalled.** The external load caused the motor to stop moving.

Running Motors for Unlimited Duration

Not specifying Encoder Control or Timed Control bit will result in unlimited running of motor, (the speed control is honored if specified). To stop motors started with 'Unlimited Duration' use respective Stop command from the command set.

⚠ **NOTE**

When motors are set to run for 'Unlimited Duration', they will continue to run until a Stop command is issued (or power is disconnected). In other words, after starting the motors for 'Unlimited Duration' if your program does something else without stopping the motors, they will continue to run.

How to detect if motor is moving or not moving:

In General:

Anytime when the 'Stalled' bit is 1, the motor is not moving.

Running in encoder mode:

During moving:

Position Control bit (bit 3) is 1

'Motor is Powered' bit (bit 2) is 1

Finished moving normally:
All bits are zero
Stopped due to stall:
Position Control bit (bit 3) is 1
'Motor is Powered' bit (bit 2) is 1
Stalled bit (bit 7) is 1

Running in timed mode:
During moving:
Timed Mode bit (bit 6) is 1
'Motor is Powered' bit (bit 2) is 1
Finished moving normally:
All bits are zero
Stopped due to stall (while time is not over):
Timed Mode bit (bit 6) is 1
'Motor is Powered' bit (bit 2) is 1
Stalled bit (bit 7) is 1
Stopped due to stall (after time is over):
All bits are zero.

Using RCX Motors with EVShield:

You can connect the RCX motors (71427) to EVShield, using NXT Conversion cable (part #1676).