What is NXTMMX-v2

NXTMMX is a multiplexer to control additional NXT (or EV3) Motors from your NXT or EV3.

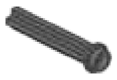
Connections and Placement

NXTMMX can be connected to any of the four sensor ports of NXT or EV3 by using standard cables from NXT set, or FlexiCable from mindsensors.com.
Connect your motors to ports specified as M1 and M2.
Connect NXT or EV3 to port specified as 'NXT'.
Additional Digital Sensors (or another NXTMMX) can be connected to port specified as 'Sensor'.

⚠ NOTE
Port specified as 'Sensor' on NXTMMX only supports I2C or Digital sensors or multiplexers.
If you intend to daisy chain several NXTMMX's or other digital sensors, please read the relevant section from the Advanced Information Appendix of this doc.

Mounting NXTMMX on your contraption

The holes on the NXTMMX enclosure are designed for tight fit of Technic pins (or axles) with '+' cross section.
The holes however are not designed for repeated insertions/removals of these pins.



To mount NXTMMX on your contraption we suggest that you use two dark gray 'Technic Axle 3 with Stud' as shown.
Insert axles from the top of the device and secure with a bushing on the back or mount it on your contraption directly.
Alternately, you may use blue 'Technic Axle Pin with Friction', as shown.
While disassembling contraption, leave the pins on the device.

Supplying power to your NXTMMX

The NXTMMX has Green terminals to connect external battery.

⚠ **NOTE**
NXTMMX is rated for external power supply of 10 volts DC max. Ensure to not exceed this value. While connecting external battery, ensure correct polarity.

Battery Options:
6AA batteries (rechargeable or non-rechargeable).
To hold and mount these batteries, you can purchase a holder with NXT mounts from the following location.

http://www.mindsensors.com/ev3-and-nxt/57-6aa-battery-holder-with-ev3-mount

RC rechargeable battery - 7.2Volts or 9.6Volts Ni-CD or Ni-MH RC rechargeable battery.
You can buy such battery and it's charger in local toy store.

To connect these batteries to NXTMMX, you can purchase a connector cable from the following location:

http://www.mindsensors.com/rpi/10-rc-battery-pack-connector-cable

Lithium-Polymer Battery (also known as Li-Po or LiPo batteries)
With NXTMMX, use 7.4 Volts battery.
Select the capacity based on your load and anticipated duration of use. (For nominal load, we recommend 7.4Volts, 800 mAh battery.)

You can purchase these batteries in electronic hobby stores, or on-line RC toys stores. An example url is here:

To connect these batteries to your NXTMMX, you can purchase connector pins from the following location:

http://www.mindsensors.com/ev3-and-nxt/120-lipo-battery-adapter-pins?search_query=lipo&results=1

Feature Highlights

Table below lists the important features provided by the NXTMMX. To use these features in your specific programming environment, please read your API help file (or NXT-G block help file) or program header file.

Feature	**Description**
Timed Control of each motor	Each motor can be run for a specified duration of time.
Encoder control of each motor	Each motor can be run from its current Encoder position to a new position (with or without a specific speed).
Speed control of each motor	Speed of each motor can be controlled in timed run or encoder based run.
Brake Vs Float while stopping the motors	Each motor can be set to Brake (where motor shaft can not be turned easily), Vs Float (where motor shaft is free to rotate by external force.
Holding Encoder position	At the end of run, hold the encoder position (i.e. motor turned by external force is restored to last set encoder position).
Turning motor by degrees	Move it in forward or reverse direction.
Turning motor by rotations	360 degrees makes one rotation.
Running operations asynchronously	While a motor is running other operations may be performed.
Running motors for unlimited duration.	While motors are running, you can also perform other operations. 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 exits without stopping the motors, they will continue to run.
Stopping motors abruptly.	
Reading Motor Encoders	You can read the value of each encoder from the NXTMMX.

Programming Techniques for NXTMMX

EV3 Method:
To use capabilities of the sensor, please download EV3 blocks available at the following URL:
http://www.mindsensors.com/index.php?controller=attachment&id_attachment=156

Installation instructions for EV3 block are available at:
http://www.mindsensors.com/content/13-how-to-install-blocks-in-ev3

Download EV3 sample program from following URL and modify it to suit your needs.
http://www.mindsensors.com/index.php?controller=attachment&id_attachment=155

NXT-G Method:
Please download custom NXT block from following URL and import it into your NXT-G IDE.

http://www.mindsensors.com/index.php?controller=attachment&id_attachment=96

If you need instructions on how to import this block in NXT-G, visit following url:
http://www.mindsensors.com/content/21-nxt-g-blocks-how-to-install-blocks

Also download sample NXT-G programs from following URL and modify them to suit your needs:
http://www.mindsensors.com/index.php?controller=attachment&id_attachment=106

⚠ **NOTE**
Ensure to use latest LEGO firmware on NXT (1.29 or higher).

RobotC Method:
Download the library file from following URL, and at the top of your RobotC program file, include it with following directive:

```
#include "NXTMMX-lib.h"
```

Download library file and sample program from following URL:
http://sourceforge.net/projects/rdpartyrobotcdr/

Also, ensure to download the help file, **NXTMMX-README-RobotC-api.html,** this file explains various functions available for your program.

NXC Method:
Download the library file from following URL, and at the top of your NXC program file, include it with following directive:

```
#include "NXTMMX-lib.nxc"
```

Library file URL:
http://www.mindsensors.com/index.php?controller=attachment&id_attachment=68

Ensure to download the help file, **NXTMMX-README-nxc-api.html,** this file explains various functions available for your program.
You can also download sample program from this location, and modify it to meet your needs.

Current Characteristics

Average current consumption of this device is about 5.0 mA.
NXTMMX can deliver upto 1 Amp current per NXT Motor attached.
Drawing of excessively large current (such as incorrect motors or stalled motors) will result in internal shutdown until the situation is corrected.

I2C Bus address

Factory Default Address: 0x06

Changing the I2C Bus Address:
The I2C bus address of NXTMMX can be changed. To set an address different from default address, send sequence of following commands on the command register:
0xA0, 0xAA, 0xA5, <new I2C address>

Note: Send these commands with no break/read operation in between. This new address is effective immediately. Please note down your address carefully for future reference.

Instructions for changing the I2C address can be found at:

NXT and EV3

http://www.mindsensors.com/blog/how-to/change-i2c-device-address.

PiStorms
http://www.mindsensors.com/blog/how-to/change-i2c-device-address-with-pistorms

APPENDIX A - Advanced Information

I2C Registers:

The NXTMMX appears as a set of registers as follows:

Register	Read	Write
0x00-0x07	Firmware version – *Vxxxx*	-
0x08-0x0f	Vendor Id – *mndsnsrs*	-
0x10-0x17	Device ID – NxTMMX	-
0x41	You can read the NXTMMX battery voltage at this register. (voltage in milli-volts = register value * 37). for EV3 compatible firmware, voltage is relocated to register 0x90	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 description below for details of this register).	Command register A for Motor 2 (read the description below for details of this register).
	Motor Read Parameters	
0x62	Encoder position of Motor 1 (long) 0x62: Least Significant Byte 0x63: Byte 2 0x64: Byte 3 0x65: Most Significant byte	-
0x66	Encoder position of Motor 2 (long) 0x66: Least Significant Byte 0x67: Byte 2 0x68: Byte 3 0x69: Most Significant Byte	-
0x72	Status Motor 1 (byte). See section below for details of this register.	
0x73	Status Motor 2 (byte). See section below for details of this register.	
0x76	Tasks Running for Motor 1 (byte)	
0x77	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.**
0x7A	Kp for Encoder Position Control (int) 0x7A: Least Significant Byte 0x7B: Most Significant Byte	Kp for Encoder Position Control (int)
0x7C	Ki for Encoder Position Control (int) 0x7C: Least Significant Byte 0x7D: Most Significant Byte	Ki for Encoder Position Control (int)

0x7E	Kd for Encoder Position Control (int) 0x7E: Least Significant Byte 0x7F: Most Significant Byte	Kd for Encoder Position Control (int)
0x80	Kp for Speed Control (int) 0x80: Least Significant Byte 0x81: Most Significant Byte	Kp for Speed Control (int)
0x82	Ki for Speed Control (int) 0x82: Least Significant Byte 0x83: Most Significant Byte	Ki for Speed Control (int)
0x84	Kd for Speed Control (int) 0x84: Least Significant Byte 0x85: Most Significant Byte	Kd for Speed Control (int)
0x86	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).
0x87	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).
0x90	Voltage value	

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.

CMD	Hex	Description
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 bit (bit 0)	**Speed control of your motor.** The NXTMMX will honor speed values specified in the speed register for the respective motor.
Bit 1	**Ramp the speed up or down.** While starting the motor or changing speed, the NXTMMX 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 NXTMMX 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 NXTMMX 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 NXTMMX will continue to hold the Encoder position (i.e. if motor is

Register Bit	**Turn this bit to 1 for following functionality**
	turned by external force, NXTMMX 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 NXTMMX 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 exits 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.

Connecting other sensors on daisy chain port of NXTMMX:

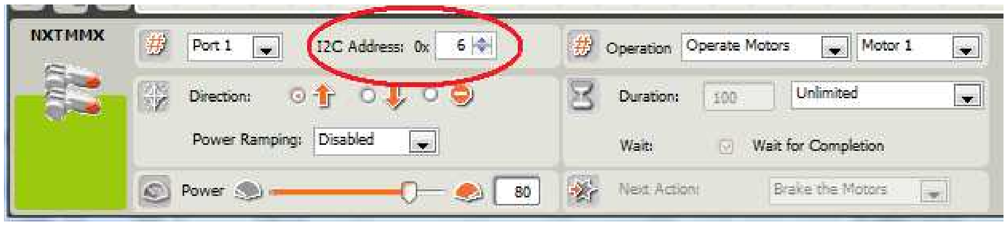
You can connect only I2C digital sensors (or controllers) to the daisy chain port of NXTMMX.
In other words, you can not connect Touch sensors (which are Analog) or most of the LEGO EV3 sensors (which are UART).

Daisy chaining NXTMMX:

If you are daisy chaining upto 5 NXTMMX, you should change I2C address of all but one NXTMMX.
For example, change the second NXTMMX to address to 0x08, third to 0x10 and so on. (or something that does not conflict with any other device address that you are attaching on the same port).

See the 'I2C Bus Address' section above for information on changing the I2C address.

Install the program on your NXT, attach the NXTMMX to Port 1 (attach only the NXTMMX for which you need to change the address), and run the program, follow on screen instructions to select new address and update it.
(With a pencil make a note of the new address on the NXTMMX).
While using this NXTMMX in your program use the new address (picture below shows where to use the new address in NXT-G program):



NXTMMX
Port 1
I2C Address: 0x 6
Operation: Operate Motors
Motor 1
Direction:
Duration: 100 Unlimited
Power Ramping: Disabled
Wait: Wait for Completion
Power 80
Next Action: Brake the Motors

If you are using any other programming environment, the API reference should indicate where to specify the new address.

If you are daisy chaining 5 or more NXTMMX or other digital sensors, you may have to disable Pull up resistors in some of the devices. The Pull up resistors in NXTMMX can be disabled by installing a special firmware.

For special firmware with pull ups disabled, write an e-mail to support@mindsensors.com.

Follow the firmware upgrade procedure mentioned below.
Upon upgrade, the NXTMMX must always be used with a device with Pull up resistor in it. (e.g. Ultrasonic Sensor)

TIP
Upon upgrade, write on the NXTMMX with a pencil that pull ups in it are disabled.

Theory behind this:

The I2C specification specifies that there be only one pull up resistor on the I2C bus. The specs also recommend that the pull up resistor be in the master device, and slave devices should not have pull up.
In case of NXT, the pull ups are in sensors. When you daisy chain multiple sensors on one I2C bus, the pull ups start to add up, and that deteriorates the quality of communication between the NXT and the sensors. On a smaller scale (usually up to 5 devices) this deterioration has no effect on I2C network performance, however when you add more devices, you may encounter data errors.

Upgrading NXTMMX firmware:

The NXTMMX is shipped with latest stable firmware, if you need to change/upgrade the firmware, follow the procedure described at URL below.
http://www.mindsensors.com/content/62-firmware-upgrader-for-compatible-devices

If the Firmware Upgrader doesn't detect your NXTMMX, try this: When the firmware Upgrader is looking for device, attach the device, remove it momentarily and attach it again.