Version: 1.00

LEGO® MINDSTORMS® NXT Bluetooth® Developer Kit

TABLE OF CONTENTS

HARDWARE SPECIFICATION FOR THE NXT BRICK

The LEGO® MINDSTORMS® NXT brick uses various advanced electronics to yield its broad functionality. For details on the hardware functionality of the LEGO® MINDSTORMS® NXT, see the LEGO MINDSTORMS NXT Hardware Developer Kit document.

Here is a summary list of hardware specifications for the NXT brick:

Main processor:	Atmel® 32-bit ARM® processor, AT91SAM7S256 - 256 KB FLASH - 64 KB RAM - 48 MHz
Co-processor:	Atmel® 8-bit AVR processor, ATmega48 - 4 KB FLASH - 512 Byte RAM - 8 MHz
Bluetooth wireless communication	CSR BlueCore™ 4 v2.0 +EDR System - Supporting the Serial Port Profile (SPP) - Internal 47 KByte RAM - External 8 MBit FLASH - 26 MHz
USB 2.0 communication	Full speed port (12 Mbit/s)
4 input ports	6-wire interface supporting both digital and analog interface - 1 high-speed port, IEC 61158 Type 4/EN 50170 compliant
3 output ports	6-wire interface supporting input from encoders
Display	100 x 64 pixel LCD black & white graphical display - View area 26 x 40.6 mm
Loudspeaker	Sound output channel with 8-bit resolution - Supporting sample rate 2-16 KHz
4 button user-interface	Rubber buttons
Power source	6 AA batteries - Recommend alkaline batteries - Rechargeable Lithium-Ion battery 1400 mAH is available
Connector	6-wire industrial-standard connector, RJ12 Right side adjustment

BLUETOOTH® FUNCTIONALITY WITHIN THE NXT

The NXT brick supports wireless communication using Bluetooth® by including a CSR BlueCore™ 4 version 2 chip. The NXT brick can be connected wirelessly to three other devices at the same time but can only communicate with one device at a time. This functionality has been implemented using the Serial Port Profile (SPP), which can be considered a wireless serial port. The NXT brick can communicate with Bluetooth devices that can be programmed to communicate using the LEGO® MINDSTORMS® NXT Communication Protocol commands and that support the Serial Port Profile (SPP). It's possible to send programs and sound files between NXT bricks and to use wireless communication to send and receive information between bricks during program execution. To reduce the power consumption used by Bluetooth, the technology has been implemented as a Bluetooth® Class II device, which means that it can communicate up to a distance of approximately 10 meters.

BLUETOOTH FUNCTIONALITY WITHIN THE NXT BRICK

The Bluetooth functionality within the NXT brick is set up as a master/slave communication channel. This means that one NXT within the network needs to function as the master device and that other NXT bricks communicate through it if they need to. The figure below shows which NXT devices can communicate directly within a network.

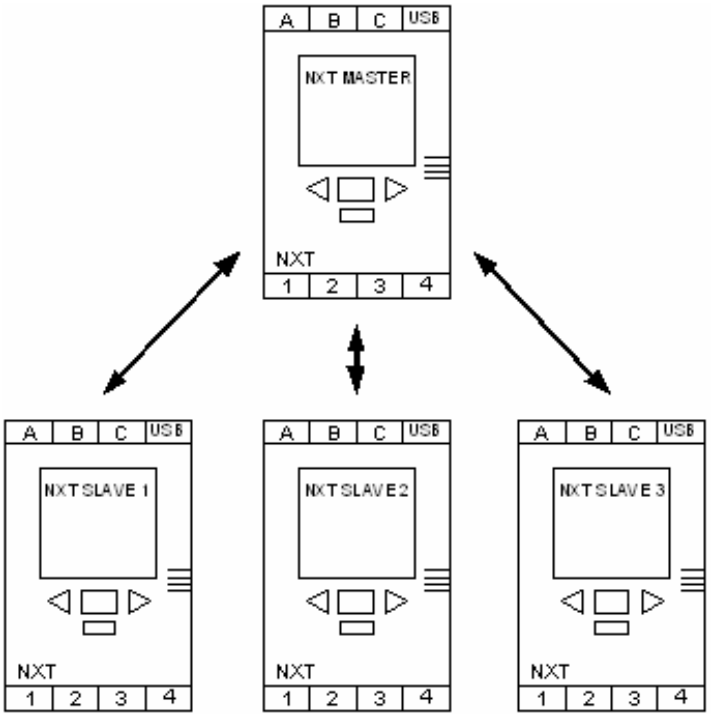

Figure 1: Illustrating 4 NXTs communicating using Bluetooth®

As shown in the figure above, the MASTER NXT can be connected to three other Bluetooth devices at the same time. The MASTER NXT can only communicate with one of the SLAVE devices during a given moment, meaning that if the MASTER NXT is communicating with NXT SLAVE 1 and NXT SLAVE 3 starts sending data to the MASTER NXT, the MASTER NXT will not evaluate the received data until it switches to NXT SLAVE 3.

An NXT is not able to function as both a MASTER and SLAVE device at the same time because this could cause lost data between NXT devices. This functionality (i.e., serving as a master and slave device at the same time) has been disabled in the standard LEGO MINDSTORMS NXT firmware.

Connections to other Bluetooth devices occur through channels. The NXT has four connection channels used for Bluetooth communication. Channel 0 is always used by SLAVE NXT devices in communicating with the MASTER NXT (i.e., *towards* the MASTER NXT) while channels 1, 2 and 3 are used for outgoing communication *from* the MASTER device to the SLAVE devices.

In figure 1 above, NXT MASTER will use communication channels 1, 2 and 3 when communicating respectively with NXT SLAVE1, NXT SLAVE 2 and NXT SLAVE 3. When one of the NXT SLAVEs communicates with the NXT MASTER, it will use communication channel 0.

INTERFACING WITH THE BLUECORE™ CHIP

Bluetooth® functionality within the NXT is implemented using a standalone chip, a CSR BlueCore™ 4 with an external 8 Mbit FLASH memory. The Bluetooth chip from CSR contains all the necessary hardware to run a self-contained Bluetooth node. A 16-bit integrated processor runs the Bluetooth stack implemented by CSR, called Bluelab. The NXT runs version 3.2 of Bluelab. The firmware within the BlueCore™ integrates a user programmable VM-task allowing us to control and run small amounts of application code. A command interpreter is integrated within the VM that is able to decode and respond to commands received through the UART interface from the ARM7 processor.

The VM has a full implementation of both the Bluetooth SPP-A and SPP-B profiles. The SPP-A profile is used when the local BlueCore™ is the connection initiator while the SPP-B profile is used when another Bluetooth device initiates the connection. The BlueCore™ uses what is referred to as "stream-mode" to exchange data at a rate of <= 220 K baud after a connection is established. When BlueCore™ is not in "stream-mode," it is in "command-mode" which is used to control the VM application within BlueCore™ and by extension, the Bluetooth functionality within the NXT. Which communication type the UART includes is controlled by two interface signals (ARM7__CMD & BC4__CMD).

For a detailed description of the communication protocol used between the ARM7 processor and the BlueCore™ chip, see Appendix 3.
The figure below shows the interface between the ARM7 processor and the BlueCore™ chip. (Its functionality is explained below the figure.) For a detailed description of the pin layout, see the hardware schematics for the NXT brick.

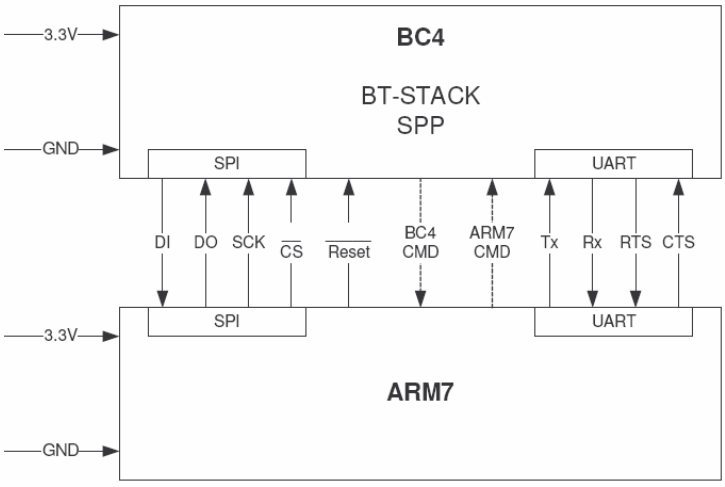

Figure 2: Hardware interface between the ARM7 and BlueCore™ chip

The SPI interface allows the possibility of updating the BlueCore™ chip. It is not in use during normal operation of the NXT brick. The SPI interface is shared with the display within the NXT brick.

The Reset pin is used at startup to re-initialize the chip correctly and to disable Bluetooth.

BC4-CMD: Indication from the BlueCore™ to the ARM7 as to which data type the BlueCore™ expects to send to the ARM7.

ARM7-CMD: Indication from the ARM7 to the BlueCore™ as to which data type the ARM7 expects to send to the BlueCore™.

UART communication is used for both data and command communication between the BlueCore™ and the ARM7 processor.

UART interface between the ARM7 and the BlueCore™ chip

The UART within the BlueCore™ chip is initialized for communication with the ARM7 using the following set-up (both for stream-mode and command-mode):

Communication speed:	460.8 K bit/s
Data bits:	8 bits
Parity:	No parity bits
Stop bit:	One stop bit
Flow control:	Hardware handshake signals (RTS & CTS)

To establish Bluetooth communication between two Bluetooth devices, several steps need to be carried out. Within the NXT brick, all Bluetooth® functionality is handled by the BlueCore™ chip and its functionality is controlled through the command-mode setup and the specific commands, which are described in the LEGO® MINDSTORMS® NXT ARM Bluetooth® Interface document.

All of the commands needed to control the BlueCore™ chip in setting up Bluetooth® communication between devices are handled by the communication module of the standard LEGO® MINDSTORMS® NXT firmware.

During any third-party firmware development for the ARM7 processor, it's important to use the interface provided in the LEGO® MINDSTORMS® NXT ARM7 Bluetooth® Interface document; the virtual machine running in the BlueCore™ firmware is only able to understand these commands. All other commands will cause the BlueCore™ firmware to panic, which will require a hardware reset (i.e., turning the brick off and then on).

BLUETOOTH® DEVICES COMMUNICATING WITH THE NXT

The LEGO® MINDSTORMS® NXT brick can communicate with external Bluetooth devices that use the serial port profile (SPP) and can be programmed to use the LEGO® MINDSTORMS® NXT Communication Protocol. A detailed description of the protocol's commands and their respective functions can be found in the LEGO® MINDSTORMS® NXT Communication Protocol document.

It's also possible to send direct commands to the LEGO® MINDSTORMS® NXT brick. Direct commands are interpreted by the NXT brick and translated into specific functions without additional user interaction. This enables direct control of the NXT brick from an external Bluetooth® device such as a mobile phone or PDA. A detailed description of direct commands and their respective functions is found in the LEGO® MINDSTORMS® NXT Direct Command document.

BLUETOOTH® COMMUNICATION WITH LEGO® MINDSTORMS® NXT PROGRAMS

It's possible to send and receive Bluetooth® messages between Bluetooth® devices while using the LEGO® MINDSTORMS® NXT software. These tasks are performed using the message commands, MessageRead and MessageWrite, described in the LEGO® MINDSTORMS® NXT Direct Command document.

As described earlier, Bluetooth functionality is implemented as a master-slave protocol with all communication controlled by the master device. This allows for reliable communication especially when multiple slave devices are connected to one master device. When the master device is programmed to verify whether it has received data from one of three potential slave devices, it will actually send a "MessageRead" command to the specific slave device to verify that it has data ready for the master device.

Sending Bluetooth® data to external Bluetooth devices

This section describes the data that will be sent from an NXT to another Bluetooth® device (with the NXT functioning as the master during the communication).

MessageWrite
Byte 0: 0x00 or 0x80
Byte 1: 0x09
Byte 2: Inbox number (0 – 9)
Byte 3: Message size
Byte 4 - N: Message data, where N = Message size + 3

Message data is treated as a string; it must include null termination to be accepted. Accordingly, message size must include the null termination byte. Message size must be capped at 59 for all message packets to be legal on USB!

Return package:
Byte 0: 0x02
Byte 1: 0x09
Byte 2: Status Byte

Reading Bluetooth® data from external Bluetooth devices

This section describes the data that will be sent from an NXT to another Bluetooth® device (with the NXT functioning as the master during the communication and polling the slave for data).

First, the NXT will send the following *MessageRead* command to the slave device:

MessageRead
Byte 0: 0x00 or 0x80
Byte 1: 0x13
Byte 2: Remote Inbox number (0 – 9)
Byte 3: Local Inbox number (0 – 9)
Byte 4: Remove? (Boolean; TRUE (non-zero) value clears message from Remote Inbox)

Return package:
Byte 0: 0x02
Byte 1: 0x13
Byte 2: Status Byte
Byte 3: Local Inbox number (0 – 9)
Byte 4: Message size
Byte 5 - 63: Message data (padded)

Message data is treated as a string; it must include null termination. Accordingly, message size includes the null termination byte. Furthermore, return packages have a fixed size, so the message data field will be padded with null bytes.

Note that the remote Inbox number may specify a value of 0-19, while all other mailbox numbers should remain below 9. This is due to the master-slave relationship between connected NXT bricks. Slave devices may not initiate communication transactions with their masters, so they store outgoing messages in the upper 10 mailboxes (indices 10-19). Use the MessageRead command from the master device to retrieve these messages.

When reading remote messages from slave devices, send the following commands:
0x05, 0x00, 0x00, 0x13, 0x0A, 0x00, 0x01 => Read Mailbox 0 from slave and clear message on slave
0x05, 0x00, 0x00, 0x13, 0x0B, 0x01, 0x01 => Read Mailbox 1 from slave and clear message on slave

If data is ready from the slave device, the following message will be send to the master NXT from the slave device:

MessageWrite
Byte 0: 0x00 or 0x80
Byte 1: 0x09
Byte 2: Inbox number (0 – 9)
Byte 3: Message size
Byte 4 - N: Message data, where N = Message size + 3

Message data is treated as a string; it must include null termination to be accepted. Accordingly, message size must include the null termination byte. Message size must be capped at 59 for all message packets to be legal on USB!

Return package:
Byte 0: 0x02
Byte 1: 0x09
Byte 2: Status Byte

APPENDIX

1. LEGO® MINDSTORMS® NXT Communication Protocol
2. LEGO® MINDSTORMS® NXT Direct Commands
3. LEGO® MINDSTORMS® NXT ARM7 Bluetooth® Interface specification