# Contents

1 Purpose .................................................. 3  

2 Rewrites ...................................................... 3  

3 Setting up sensors  
   3.1 Purpose .............................................. 3  
   3.2 Constants ............................................. 3  
   3.3 Methods ................................................. 4  

4 Output port constants  
   4.1 Constants .............................................. 4  

5 Using touch sensors  
   5.1 Purpose .............................................. 5  
   5.2 Methods ............................................... 5  

6 Using light sensors  
   6.1 Purpose .............................................. 5  
   6.2 Methods ............................................... 6  

7 Using colour sensors  
   7.1 Purpose .............................................. 7  
   7.2 Constants ............................................. 7  
   7.3 Methods ............................................... 7  

8 Using sound sensors  
   8.1 Purpose .............................................. 8  
   8.2 Methods ............................................... 8
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Using ultrasonic sensors</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>9.1 Purpose</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>9.2 Methods</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>Using rotation sensors</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>10.1 Purpose</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>10.2 Methods</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>Using motors</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>11.1 Purpose</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>11.2 Methods</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>Using lamps</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>12.1 Methods</td>
<td>11</td>
</tr>
<tr>
<td>13</td>
<td>Making sounds</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>13.1 Constants</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>13.2 Methods</td>
<td>12</td>
</tr>
<tr>
<td>14</td>
<td>Using the LCD</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>14.1 Purpose</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>14.2 Constants</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>14.3 Methods</td>
<td>13</td>
</tr>
<tr>
<td>15</td>
<td>Using buttons</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>15.1 Purpose</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>15.2 Rewrites</td>
<td>14</td>
</tr>
<tr>
<td>16</td>
<td>Waiting for fixed times</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>16.1 Methods</td>
<td>15</td>
</tr>
<tr>
<td>17</td>
<td>Terminating a program</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>17.1 Methods</td>
<td>15</td>
</tr>
<tr>
<td>18</td>
<td>Math</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>18.1 Purpose</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>18.2 Constants</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>18.3 Methods</td>
<td>16</td>
</tr>
<tr>
<td>19</td>
<td>Strings</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>19.1 Purpose</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>19.2 Methods</td>
<td>19</td>
</tr>
<tr>
<td>20</td>
<td>Threads</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>20.1 Purpose</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>20.2 Rewrites</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>20.3 Methods</td>
<td>20</td>
</tr>
</tbody>
</table>
1 Purpose

This environment supports programming a Lego Mindstorms NXT robot, via the Lejos NXJ system.

Requires Lejos-NXJ version 0.9.1.

2 Rewrites

mandatory

void main()

Purpose: A program that is organised into methods must have a main method (a procedure with no arguments). This will be the first method to execute. mashc automatically rewrites this method to conform to standard Java.

3 Setting up sensors

3.1 Purpose

At the start of a program, the sensor ports need to be set up to work with the kind of sensor that is plugged into it. Sensors can be either the old RCX kinds or the new NXT kinds.

The rotation sensor built into an NXT motor does not have to be set up.

Light, sound and proximity sensors benefit from a time delay of about half a second between setting the sensor up and making measurements.

3.2 Constants

final int TOUCH

Purpose: Constant to select sensor type touch (NXT or RCX).

final int LIGHT_FLOOD

Purpose: Constant to select sensor type light (NXT with the flood-light on).

final int LIGHT_NOFLOOD
**Purpose:** Constant to select sensor type light (NXT with the floodlight off).

`final int LIGHT_RCX`

**Purpose:** Constant to select sensor type light (RCX with the floodlight on).

`final int COLOR`

**Purpose:** Constant to select sensor type color (NXT), initially with its floodlight off.

`final int ROTATION_RCX`

**Purpose:** Constant to select sensor type rotation (RCX).

`final int SOUND`

**Purpose:** Constant to select sensor type sound (NXT).

`final int PROXIMITY`

**Purpose:** Constant to select sensor type ultrasonic proximity (NXT).

### 3.3 Methods

`void setUpSensor (int port, int type)`

**Purpose:** Sets up the port to be a sensor of the given type.

**Precondition:** `port` is 1, 2, 3, or 4.

**Precondition:** `type` is TOUCH, LIGHT_FLOOD, LIGHT_NOFLOOD, LIGHT_RCX, COLOR, ROTATION_RCX, SOUND, or PROXIMITY.

### 4 Output port constants

#### 4.1 Constants

`final int A`

**Purpose:** Constant to select port A.

`final int B`

**Purpose:** Constant to select port B.

`final int C`

**Purpose:** Constant to select port C.
5 Using touch sensors

5.1 Purpose

These methods provide either waits for a touch sensor’s state to change or the current state of the touch sensor.

These methods all assume that the \texttt{port} number you provide as a parameter has been set up as a touch sensor of either the RCX or NXT kind.

5.2 Methods

\texttt{void waitForPush (int port)}

\textit{Purpose}: Makes the program wait until the touch sensor on \texttt{port} is pushed.

\textit{Precondition}: \texttt{port} is 1, 2, 3 or 4.

\texttt{void waitForLetGo (int port)}

\textit{Purpose}: Makes the program wait until the touch sensor on \texttt{port} is let go.

\textit{Precondition}: \texttt{port} is 1, 2, 3, or 4.

\texttt{boolean isPushed (int port)}

\textit{Purpose}: Returns \texttt{true} if and only if the touch sensor on \texttt{port} is currently pushed.

\textit{Precondition}: \texttt{port} is 1, 2, 3, or 4.

6 Using light sensors

6.1 Purpose

Most of these methods provide waits for the light level reported by a light sensor to change. In addition the current light level reported by a light sensor can be obtained and (for NXT sensors) the floodlight turned on or off.

These methods all assume that the \texttt{port} number you provide as a parameter has been set up as a light sensor, of either the RCX or NXT kind, or as a colour sensor.
6.2 Methods

void waitForLighter (int port, int dif)

Purpose: Makes the program wait until the light sensor reading on port is increased by dif.

Precondition: port is 1, 2, 3 or 4.

Precondition: dif is between 0 and 100, inclusive. 0 is no wait at all. Real light levels never really change by anything like 100.

void waitForLight (int port, int light)

Purpose: Makes the program wait until the light sensor reading on port is at least the desired light level.

Precondition: port is 1, 2, 3 or 4.

Precondition: light is between 0 and 100, inclusive.

void waitForDarker (int port, int dif)

Purpose: Makes the program wait until the light sensor reading on port is decreased by dif.

Precondition: port is 1, 2, 3 or 4.

Precondition: dif is between 0 and 100, inclusive. 0 is no wait at all. Real light levels never really change by anything like 100.

void waitForDark (int port, int light)

Purpose: Makes the program wait until the light sensor reading on port is at most the desired light level.

Precondition: port is 1, 2, 3 or 4.

Precondition: light is between 0 and 100, inclusive.

int getLight (int port)

Purpose: Returns the current light sensor reading on port. The reading will depend on whether the floodlight is on or not, and what colour it is.

Precondition: port is 1, 2, 3 or 4.

void setFloodlight (int port, boolean floodlight)

Purpose: Turn the floodlight of the light sensor on port on or off. For the colour sensor, turning it on, sets it to red.

Precondition: port is 1, 2, 3 or 4.
7 Using colour sensors

7.1 Purpose
These constants and methods are those specific to the NXT colour sensor. All of the light sensor methods can also be used with the colour sensor.
These methods all assume that the port number you provide as a parameter has been set up as a colour sensor.

7.2 Constants

final int BLACK
Purpose: ID code for a detectable colour.

final int BLUE
Purpose: ID code for a detectable colour and for selecting the floodlight colour.

final int GREEN
Purpose: ID code for a detectable colour and for selecting the floodlight colour.

final int NONE
Purpose: Constant to select the floodlight colour (off).

final int RED
Purpose: ID code for a detectable colour and for selecting the floodlight colour.

final int WHITE
Purpose: ID code for a detectable colour.

7.3 Methods

void setFloodlight (int port, int color)
Purpose: Turn the floodlight of the colour sensor at port on with the selected color, or off with NONE.
Precondition: port is 1, 2, 3 or 4.
Precondition: color is BLUE, GREEN, RED, or NONE (off).

int getColor (int port)
Purpose: This function automatically controls the floodlight colour and reads light levels and decides which colour is under the colour sensor at port. It returns one of BLACK, BLUE, GREEN, RED, or WHITE.
Precondition: port is 1, 2, 3 or 4.
8 Using sound sensors

8.1 Purpose

These methods provide either waits for the sound level reported by a sound sensor to change or the current volume reported by the sound sensor.

These methods all assume that the port number you provide as a parameter has been set up as a sound sensor.

8.2 Methods

void waitForLouder (int port, int dif)

Purpose: Makes the program wait until the sound sensor reading on port is increased by dif.

Precondition: port is 1, 2, 3 or 4.

Precondition: dif is between 0 and 100, inclusive. 0 is no wait at all.

void waitForLoud (int port, int volume)

Purpose: Makes the program wait until the sound sensor reading on port is at least the desired volume level.

Precondition: port is 1, 2, 3 or 4.

Precondition: volume is between 0 and 100, inclusive.

void waitForQuieter (int port, int dif)

Purpose: Makes the program wait until the sound sensor reading on port has decreased by dif.

Precondition: port is 1, 2, 3 or 4.

Precondition: dif is between 0 and 100, inclusive. 0 is no wait at all.

void waitForQuiet (int port, int volume)

Purpose: Makes the program wait until the sound sensor reading on port is at most the desired volume level.

Precondition: port is 1, 2, 3 or 4.

Precondition: volume is between 0 and 100, inclusive.

int getVolume (int port)

Purpose: Returns the current volume sensor reading on port.

Precondition: port is 1, 2, 3 or 4.
9 Using ultrasonic sensors

9.1 Purpose

These methods provide either waits for the distance reported by an ultrasonic sensor to change or the current distance reported by the ultrasonic sensor.

These methods all assume that the port number you provide as a parameter has been set up as an ultrasonic sensor.

A known bug in Lejos 0.85 means the ultrasonic sensor does not work in input port 4.

9.2 Methods

void waitForNearer (int port, int dif)

Purpose: Makes the program wait until the Ultrasonic sensor on port senses that distance has decreased by dif cm.

Precondition: port is 1, 2, 3 or 4.

Precondition: dif is between 0 and 255, inclusive. 0 is no wait at all.

void waitForNear (int port, int distance)

Purpose: Makes the program wait until the ultrasonic sensor on port reports a distance which is at most distance cm. A distance of 255 means no object is in view.

Precondition: port is 1, 2, 3 or 4.

Precondition: distance is between 0 and 255, inclusive.

void waitForFurther (int port, int dif)

Purpose: Makes the program wait until the ultrasonic sensor on port reports the distance has increased by dif cm.

Precondition: port is 1, 2, 3 or 4.

Precondition: dif is between 0 and 255, inclusive. 0 is no wait at all. 255 represents no object currently in view.

void waitForFar (int port, int distance)

Purpose: Makes the program wait until the ultrasonic sensor on port reports a distance which is at least distance cm. A distance of 255 means no object is in view.
**Precondition:** port is 1, 2, 3 or 4.

**Precondition:** distance is between 0 and 255, inclusive.

```c
int getDistance (int port)
```

*Purpose:* Returns the current ultrasonic sensor reading on port in centimetres. 255 means no object is in view.

**Precondition:** port is 1, 2, 3 or 4.

## 10 Using rotation sensors

### 10.1 Purpose

These methods provide a wait for the rotation count reported by a rotation sensor to change, the current rotation count reported by a sensor, or reset the count to zero.

An NXT rotation sensor registers 360 counts per full 360 degree rotation. NXT rotation sensors are built into NXT motors and must be plugged into port A, B or C. There is no need to call `setUpSensor(int,int)` to set up these sensors.

An RCX rotation sensor registers 16 counts per full 360 degree rotation. It must be plugged into port 1, 2, 3 or 4 and set up like other sensor kinds.

### 10.2 Methods

```c
void resetRotation (int port)
```

*Purpose:* Sets the counter in the rotation sensor on port to zero.

**Precondition:** port is 1, 2, 3, 4, A, B, or C.

```c
void waitForRotation (int port, int rotation)
```

*Purpose:* Makes the program wait until the counter in the rotation sensor on port has changed by at least the absolute value of rotation.

This method does not reset the counter in the rotation sensor.

**Precondition:** port is 1, 2, 3, 4, A, B, or C.

```c
int getRotation (int port)
```

*Purpose:* Returns the current rotation sensor reading from port.

**Precondition:** port is 1, 2, 3, 4, A, B, or C.
11 Using motors

11.1 Purpose
The following methods apply to both NXT and RCX motors.

11.2 Methods

void motorForward (int port, int power)

*Purpose:* Make the motor on port go forwards at the given power.

*Precondition:* port is A, B, or C.

*Precondition:* power is between 0 and 100, inclusive.

void motorBackward (int port, int power)

*Purpose:* Make the motor on port go backwards at the given power.

*Precondition:* port is A, B, or C.

*Precondition:* power is between 0 and 100, inclusive.

void motorStop (int port)

*Purpose:* Stop the motor on port. Stopping a motor cuts the power to it and stops it rotating.

*Precondition:* port is A, B, or C.

void motorFloat (int port)

*Purpose:* Float the motor on port. Floating a motor, cuts power to it and lets it rotate freely.

*Precondition:* port is A, B, or C.

12 Using lamps

12.1 Methods

void lampOn (int port, int power)

*Purpose:* Make the lamp on port go on at the given power.

*Precondition:* port is A, B, or C.

*Precondition:* power is between 0 and 100, inclusive.

void lampOff (int port)

*Purpose:* Turns the lamp on port off.

*Precondition:* port is A, B, or C.
13 Making sounds

13.1 Constants

final int[] PIANO

Purpose: Attack, decay, sustain and release shape parameters to emulate a piano. Use with playNote(int[], int, int).

final int[] FLUTE

Purpose: Attack, decay, sustain and release shape parameters to emulate a flute. Use with playNote(int[], int, int).

final int[] XYLOPHONE

Purpose: Attack, decay, sustain and release shape parameters to emulate a xylophone. Use with playNote(int[], int, int).

13.2 Methods

void beep ()

Purpose: Beep once.

void twoBeeps ()

Purpose: Beep twice.

void fallingBeeps ()

Purpose: A few short beeps with descending tones.

void risingBeeps ()

Purpose: A few short beeps with ascending tones.

void buzz ()

Purpose: Makes a low buzz.

void pause (int t)

Purpose: Make no sound for t milliseconds.

void playTone (int frequency, int duration)

Purpose: Plays a tone, given its frequency (Hertz) and duration (milliseconds).

void playNote (int[] instrument, int frequency, int duration)

Purpose: Plays a note attack, decay, sustain and release shape specified by a selected instrument, given its frequency (Hertz) and duration (milliseconds).
14 Using the LCD

14.1 Purpose

The LCD display can be used to display text and graphics.

14.2 Constants

final int LCD_WIDTH

Purpose: The width of the LCD display in pixels, equals 100.

final int LCD_HEIGHT

Purpose: The height of the LCD display in pixels, equals 64.

final int LCD_COLUMNS

Purpose: The width of the LCD display in character spaces, equals 16.

final int LCD_LINES

Purpose: The height of the LCD display in character spaces, equals 8.

14.3 Methods

void clear ()

Purpose: Clears the LCD display to all white.

void setPixel (boolean p, int x, int y)

Purpose: Set the pixel at (x, y). If p is true, the pixel is set to black, white otherwise.

Precondition: 0 ≤ x < LCD_WIDTH.

Precondition: 0 ≤ y < LCD_HEIGHT.

void drawString (String s, int x, int y, boolean invert)

Purpose: Displays a string s on the LCD starting at (x, y). If invert is true then it is drawn as white on black, instead of the usual black on white.

Precondition: 0 ≤ x < LCD_COLUMNS.
Precondition: $0 \leq y < \text{LCD}\_\text{LINES}$.

void drawString (String s, int x, int y)

Purpose: Displays a string $s$ on the LCD starting at column $x$ and line $y$.

Precondition: $0 \leq x < \text{LCD}\_\text{COLUMNS}$.

Precondition: $0 \leq y < \text{LCD}\_\text{LINES}$.

void drawInt (int i, int width, int x, int y)

Purpose: Displays integer 1 on the LCD starting at column $x$ and line $y$ right justified within at least $width$ spaces.

Precondition: $1 \leq width \leq 11$.

Precondition: $0 \leq x < \text{LCD}\_\text{COLUMNS}$.

Precondition: $0 \leq y < \text{LCD}\_\text{LINES}$.

void scroll ()

Purpose: Scrolls the screen up one text line.

15 Using buttons

15.1 Purpose

There are four buttons on the front of an NXT. These rewrites define handers for pressing on or releasing the ENTER, LEFT, RIGHT and ESCAPE buttons. Usually a button should cause an action when it is released, so actions are usually put in an onRelease handler, but programs can also do something while the button is pressed by implementing an onPress handler.

Note: Releasing the ESCAPE button, by default, is the easiest way for users to terminate a program with an endless loop. Think carefully before changing that behaviour.

15.2 Rewrites

void onPressEnter ()

Purpose: Write a procedure of this type to handle presses on the ENTER button (the orange one).

void onReleaseEnter ()
Purpose: Write a procedure of this type to handle releases of the ENTER button.

void onPressLeft ()

Purpose: Write a procedure of this type to handle presses on the LEFT button.

void onReleaseLeft ()

Purpose: Write a procedure of this type to handle releases of the LEFT button.

void onPressRight ()

Purpose: Write a procedure of this type to handle presses on the RIGHT button.

void onReleaseRight ()

Purpose: Write a procedure of this type to handle releases of the RIGHT button.

void onPressEscape ()

Purpose: Write a procedure of this type to handle presses on the ESCAPE button.

void onReleaseEscape ()

Purpose: Write a procedure of this type to handle releases of the ESCAPE button.

16 Waiting for fixed times

16.1 Methods

void sleep (int ms)

Purpose: Makes the program wait for a requested number of ms (milliseconds).

17 Terminating a program

17.1 Methods

void exit ()

Purpose: Terminates the program.
18 Math

18.1 Purpose
The following are some commonly used numeric constants and functions.

18.2 Constants

final int MAX_INT

Purpose: A constant holding the maximum value an int can have, $2^{31} - 1$.

final int MIN_INT

Purpose: A constant holding the minimum value an int can have, $-2^{31}$.

final long MAX_LONG

Purpose: A constant holding the maximum value a long can have, $2^{63} - 1$.

final long MIN_LONG

Purpose: A constant holding the minimum value a long can have, $-2^{63}$.

final double PI

Purpose: The closest double approximation to $\pi$.

final double E

Purpose: The closest double approximation to $e$.

18.3 Methods

double abs (double a)

Purpose: Returns the absolute value of a.

float abs (float a)

Purpose: Returns the absolute value of a.

long abs (long a)

Purpose: Returns the absolute value of a.
int abs (int a)

*Purpose:* Returns the absolute value of `a`.

double ceil (double a)

*Purpose:* Returns the least double value that is greater than or equal to `a` and equal to an integer.

double exp (double x)

*Purpose:* Returns $e^x$, that is Euler’s constant $e$ raised to power $x$.

double floor (double a)

*Purpose:* Returns the greatest double value that is less than or equal to `a` and equal to an integer.

double log (double x)

*Purpose:* Returns the natural logarithm of `x`.

double rint (double x)

*Purpose:* Returns the closest mathematical integer to `x`.

long round (double a)

*Purpose:* Returns the closest `long` to `a`.

int round (float a)

*Purpose:* Returns the closest `int` to `a`.

double sqrt (double a)

*Purpose:* Returns the square root of `a`.

*Precondition:* $a \geq 0.0$.

double pow (double a, double b)

*Purpose:* Returns $a$ raised to the power $b$, $a^b$.

double sin (double a)

*Purpose:* Returns the trigonometric sine of `a` radians.

double cos (double a)
Purpose: Returns the trigonometric cosine of a radians.

double tan (double a)

Purpose: Returns the trigonometric tangent of a radians.

double asin (double a)

Purpose: Returns the trigonometric arc sine of a in radians.

double acos (double a)

Purpose: Returns the trigonometric arc cosine of a in radians.

double atan (double a)

Purpose: Returns the trigonometric arc tangent of a in radians.

double atan2 (double y, double x)

Purpose: Returns the angle theta from the conversion of rectangular coordinates (x, y) to polar coordinates (r, theta).

double max (double a, double b)

Purpose: Returns the greater of a and b.

float max (float a, float b)

Purpose: Returns the greater of a and b.

int max (int a, int b)

Purpose: Returns the greater of a and b.

long max (long a, long b)

Purpose: Returns the greater of a and b.

double min (double a, double b)

Purpose: Returns the lesser of a and b.

float min (float a, float b)

Purpose: Returns the lesser of a and b.

int min (int a, int b)

Purpose: Returns the lesser of a and b.

long min (long a, long b)

Purpose: Returns the lesser of a and b.

double random ()

Purpose: Returns a random value x such that 0.0 ≤ x < 1.0.
19  Strings

19.1  Purpose
The following are methods for working with Strings.

19.2  Methods

int length (String s)

    Purpose: Returns the length of s.

cchar charAt (String s, int i)

    Purpose: Returns the character at position i in s.

    Precondition: 0 ≤ i < length(s).

boolean equals (String a, String b)

    Purpose: Returns true if and only if a contains the same sequence of characters as in b.

boolean parseBoolean (String s)

    Purpose: Returns s converted to a boolean.

int parseInt (String s)

    Purpose: Returns s converted to an int.

long parseLong (String s)

    Purpose: Returns s converted to a long.

float parseFloat (String s)

    Purpose: Returns s converted to a float.

double parseDouble (String s)

    Purpose: Returns s converted to a double.

20  Threads

20.1  Purpose
A program may have up to 5 additional threads of execution.
20.2 Rewrites

void run1 ()

Purpose: Implement a run1() method that defines the actions of additional thread 1. Do no try to call it directly. To start this thread, call start1().

void run2 ()

Purpose: Implement a run2() method that defines the actions of additional thread 2. Do no try to call it directly. To start this thread, call start2().

void run3 ()

Purpose: Implement a run3() method that defines the actions of additional thread 3. Do no try to call it directly. To start this thread, call start3().

void run4 ()

Purpose: Implement a run4() method that defines the actions of additional thread 4. Do no try to call it directly. To start this thread, call start4().

void run5 ()

Purpose: Implement a run5() method that defines the actions of additional thread 5. Do no try to call it directly. To start this thread, call start5().

20.3 Methods

void start1 ()

Purpose: Call start1() to start additional thread 1.

Precondition: run1() has been implemented. If not, nxjc will report errors.

void start2 ()

Purpose: Call start2() to start additional thread 2.

Precondition: run2() has been implemented. If not, nxjc will report errors.

void start3 ()
Purpose: Call \texttt{start3()} to start additional thread 3.

Precondition: \texttt{run3()} has been implemented. If not, \texttt{nxjc} will report errors.

\textbf{void start4 ()}

Purpose: Call \texttt{start4()} to start additional thread 4.

Precondition: \texttt{run4()} has been implemented. If not, \texttt{nxjc} will report errors.

\textbf{void start5 ()}

Purpose: Call \texttt{start5()} to start additional thread 5.

Precondition: \texttt{run5()} has been implemented. If not, \texttt{nxjc} will report errors.

\section{Sensor listeners}

\subsection{Purpose}
An alternative to polling sensors is to register a listener that will respond when the sensor changes state.

\subsection{Rewrites}

\textbf{void onChange1 (int oldValue, int newValue)}

\textit{Purpose}: Write a method with this type to handle changes on sensor port 1. \texttt{oldValue} and \texttt{newValue} are the old and new \texttt{raw} sensor values, respectively.

\textbf{void onChange2 (int oldValue, int newValue)}

\textit{Purpose}: Write a method with this type to handle changes on sensor port 2. \texttt{oldValue} and \texttt{newValue} are the old and new \texttt{raw} sensor values, respectively.

\textbf{void onChange3 (int oldValue, int newValue)}

\textit{Purpose}: Write a method with this type to handle changes on sensor port 3. \texttt{oldValue} and \texttt{newValue} are the old and new \texttt{raw} sensor values, respectively.

\textbf{void onChange4 (int oldValue, int newValue)}

\textit{Purpose}: Write a method with this type to handle changes on sensor port 4. \texttt{oldValue} and \texttt{newValue} are the old and new \texttt{raw} sensor values, respectively.
21.3 Methods

void listenPort1 ()

Purpose: Starts listening to port 1.  
If the state of that sensor changes, onChange1(int, int) will be called.

void listenPort2 ()

Purpose: Starts listening to port 2.  
If the state of that sensor changes, onChange2(int, int) will be called.

void listenPort3 ()

Purpose: Starts listening to port 3.  
If the state of that sensor changes, onChange3(int, int) will be called.

void listenPort4 ()

Purpose: Starts listening to port 4.  
If the state of that sensor changes, onChange4(int, int) will be called.
Index

A, 4
abs, 16, 17
acos, 18
asin, 18
atan, 18
atan2, 18

B, 4
beep, 12
BLACK, 7
BLUE, 7
buzz, 12

C, 4
ceil, 17
charAt, 19
clear, 13
COLOR, 4
cos, 17
drawInt, 14
drawString, 13, 14

E, 16
equals, 19
exit, 15
exp, 17

fallingBeeps, 12
floor, 17
FLUTE, 12

g getColor, 7
g getDistance, 10
g getLight, 6
g getRotation, 10
g getVolume, 8
GREEN, 7

isPushed, 5

lampOff, 11
lampOn, 11

LCD_COLUMNS, 13
LCD_HEIGHT, 13
LCD_LINES, 13
LCD_WIDTH, 13
length, 19
LIGHT_FLOOD, 3
LIGHT_NOFLOOD, 3
LIGHT_RCX, 4
listenPort1, 22
listenPort2, 22
listenPort3, 22
listenPort4, 22
log, 17

main, 3
max, 18
MAX_INT, 16
MAX_LONG, 16
min, 18
MIN_INT, 16
MIN_LONG, 16
motorBackward, 11
motorFloat, 11
motorForward, 11
motorStop, 11

NONE, 7

onChange1, 21
onChange2, 21
onChange3, 21
onChange4, 21
onPressEnter, 14
onPressEscape, 15
onPressLeft, 15
onPressRight, 15
onReleaseEnter, 14
onReleaseEscape, 15
onReleaseLeft, 15
onReleaseRight, 15

parseBoolean, 19
parseDouble, 19
parseFloat, 19
parseInt, 19
parseLong, 19
pause, 12
PI, 16
PIANO, 12
playNote, 12
playTone, 12
pow, 17
PROXIMITY, 4

random, 18
RED, 7
resetRotation, 10
rint, 17
risingBeeps, 12
ROTATION_RCX, 4
round, 17
run1, 20
run2, 20
run3, 20
run4, 20
run5, 20
scroll, 14
setFloodlight, 6, 7
setPixel, 13
setUpSensor, 4
sin, 17
sleep, 15
SOUND, 4
sqrt, 17
start1, 20
start2, 20
start3, 20
start4, 21
start5, 21
tan, 18
TOUCH, 3
twoBeeps, 12

waitForDark, 6
waitForDarker, 6
waitForFar, 9
waitForFurther, 9
waitForLetGo, 5
waitForLight, 6
waitForLighter, 6
waitForLouder, 8
waitForLouder, 8
waitForNear, 9
waitForNearer, 9
waitForPush, 5
waitForQuieter, 8
waitForQuiet, 8
waitForQuieter, 8
waitForRotation, 10
WHITE, 7

XYLOPHONE, 12