

Robot Sensors

Version 0.0

Generated by Doxygen 1.8.4

Thu Jul 18 2013 11:51:48

Contents

| | | |
|----------|---|-----------|
| 1 | Connect to RobotSensors | 1 |
| 2 | Module Index | 3 |
| 2.1 | Modules | 3 |
| 3 | File Index | 5 |
| 3.1 | File List | 5 |
| 4 | Module Documentation | 7 |
| 4.1 | Direction | 7 |
| 4.1.1 | Detailed Description | 7 |
| 4.2 | Status | 8 |
| 4.2.1 | Detailed Description | 8 |
| 4.3 | Mode | 9 |
| 4.3.1 | Detailed Description | 9 |
| 4.4 | Day of Week | 10 |
| 4.4.1 | Detailed Description | 10 |
| 4.5 | Month | 11 |
| 4.5.1 | Detailed Description | 11 |
| 5 | File Documentation | 13 |
| 5.1 | robotSensors.nxc File Reference | 13 |
| 5.1.1 | Function Documentation | 15 |
| 5.1.1.1 | accelerometerRead | 15 |
| 5.1.1.2 | ambientTemperatureRead | 15 |
| 5.1.1.3 | brightnessRead | 15 |
| 5.1.1.4 | cameraLightOff | 15 |
| 5.1.1.5 | cameraLightOn | 16 |
| 5.1.1.6 | cameraRead | 16 |
| 5.1.1.7 | cameraStart | 16 |
| 5.1.1.8 | cameraStop | 16 |
| 5.1.1.9 | gyroscopeRead | 16 |
| 5.1.1.10 | magneticFieldRead | 17 |

| | | |
|----------|-------------------------------------|----|
| 5.1.1.11 | microphoneNoiseLevel | 17 |
| 5.1.1.12 | pictureAccuBrightestPoint | 17 |
| 5.1.1.13 | pictureAccuCopy | 18 |
| 5.1.1.14 | pictureAccuCut | 18 |
| 5.1.1.15 | pictureAccuDarkestPoint | 18 |
| 5.1.1.16 | pictureAccuMeanColor | 19 |
| 5.1.1.17 | pictureAccuShow | 20 |
| 5.1.1.18 | pictureAccuSubtract | 20 |
| 5.1.1.19 | pressureRead | 20 |
| 5.1.1.20 | relativeHumidityRead | 20 |
| 5.1.1.21 | speechRecognizerRead | 21 |
| 5.1.1.22 | speechRecognizerStart | 21 |
| 5.1.1.23 | speechRecognizerStop | 21 |
| 5.1.1.24 | timeRead | 21 |

Index

22

Chapter 1

Connect to RobotSensors

To establish a bluetooth connection between RobotSensors and your NXT follow this instruction. This is only necessary at the first time.

1. Download and install the app on your smartphone.
2. Start the app on your smarthphone. If bluetooth is not yet turned on you will be asked to do so.
3. Go to bluetooth-settings on your smartphone an make it visible.
4. Turn on your NXT.
5. Turn on Bluetooth on your NXT.
6. Search for Bluetooth devices.
7. Choose your smartphone from the list of found devices.
8. Connect to connection slot 1.
9. Set passkey and press ok.
10. Confirm on your smartphone with the same passkey.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

| | |
|-----------------------|----|
| Direction | 7 |
| Status | 8 |
| Mode | 9 |
| Day of Week | 10 |
| Month | 11 |

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

| | |
|--|----|
| robotSensors.nxc | 13 |
|--|----|

Chapter 4

Module Documentation

4.1 Direction

Macros

- `#define FRONT 0x00`
- `#define BACK 0x01`

4.1.1 Detailed Description

4.2 Status

Macros

- `#define SUCCESS 0x00`
- `#define ERROR 0x01`

4.2.1 Detailed Description

4.3 Mode

Macros

- #define **NORMAL** 0x00
- #define **HIGH_PASSED** 0x01
- #define **LOW_PASSED** 0x02

4.3.1 Detailed Description

4.4 Day of Week

Macros

- `#define SUNDAY 1`
- `#define MONDAY 2`
- `#define TUESDAY 3`
- `#define WEDNESDAY 4`
- `#define THURSDAY 5`
- `#define FRIDAY 6`
- `#define SATURDAY 7`

4.4.1 Detailed Description

4.5 Month

Macros

- `#define JANUARY 0`
- `#define FEBRUARY 1`
- `#define MARCH 2`
- `#define APRIL 3`
- `#define MAY 4`
- `#define JUNE 5`
- `#define JULY 6`
- `#define AUGUST 7`
- `#define SEPTEMBER 8`
- `#define OCTOBER 9`
- `#define NOVEMBER 10`
- `#define DECEMBER 11`
- `#define UNDECIMBER 12`

4.5.1 Detailed Description

Chapter 5

File Documentation

5.1 robotSensors.nxc File Reference

Macros

- #define **FRONT** 0x00
- #define **BACK** 0x01
- #define **SUCCESS** 0x00
- #define **ERROR** 0x01
- #define **NORMAL** 0x00
- #define **HIGH_PASSED** 0x01
- #define **LOW_PASSED** 0x02
- #define **SUNDAY** 1
- #define **MONDAY** 2
- #define **TUESDAY** 3
- #define **WEDNESDAY** 4
- #define **THURSDAY** 5
- #define **FRIDAY** 6
- #define **SATURDAY** 7
- #define **JANUARY** 0
- #define **FEBRUARY** 1
- #define **MARCH** 2
- #define **APRIL** 3
- #define **MAY** 4
- #define **JUNE** 5
- #define **JULY** 6
- #define **AUGUST** 7
- #define **SEPTEMBER** 8
- #define **OCTOBER** 9
- #define **NOVEMBER** 10
- #define **DECEMBER** 11
- #define **UNDECIMBER** 12

Functions

- int [gyroscopeRead](#) (int &wX, int &wY, int &wZ, unsigned long ×tampMs)
Read the values of the gyroscope.
- int [cameraStop](#) ()
Stops the camera.

- int [cameraStart](#) (unsigned byte direction)
Starts the camera.
- int [cameraRead](#) (unsigned byte accuIndex, unsigned int &width, unsigned int &height)
Takes a picture from the camera.
- int [pictureAccuShow](#) (unsigned byte accuIndex)
Displays a picture from the picture accu.
- int [pictureAccuBrightestPoint](#) (byte accuIndex, unsigned int &width, unsigned int &height, unsigned int &brightestX, unsigned int &brightestY, unsigned int &intensity)
Finds the brightest point of an image in the picture accu.
- int [pictureAccuDarkestPoint](#) (byte accuIndex, unsigned int &width, unsigned int &height, unsigned int &darkestX, unsigned int &darkestY, unsigned int &intensity)
Finds the darkest point of an image in the picture accu.
- int [pictureAccuMeanColor](#) (byte accuIndex, unsigned byte &r, unsigned byte &g, unsigned byte &b)
Calculates the mean color of an image in the picture accu.
- int [pictureAccuCut](#) (byte accuIndex, unsigned int left, unsigned int right, unsigned int top, unsigned int bottom)
Cuts off some part of an image in the picture accu.
- int [pictureAccuCopy](#) (byte sourceAccuIndex, byte targetAccuIndex)
Copies a picture from one accu place to another.
- int [pictureAccuSubtract](#) (byte accuIndexA, byte accuIndexB)
Subtracts the picture at accuIndexB from the one at accuIndexA.
- int [cameraLightOn](#) ()
Turns on the camera light.
- int [cameraLightOff](#) ()
Turns off the camera light.
- int [microphoneNoiseLevel](#) (unsigned int &noiseLevel)
Returns the surrounding noise level.
- int [brightnessRead](#) (unsigned int &brightness, unsigned long ×tampMs)
Reads the surrounding brightness.
- int [timeRead](#) (int &year, unsigned byte &month, unsigned byte &day, unsigned byte &dayOfWeek, unsigned byte &hour, unsigned byte &minute, unsigned byte &second, unsigned int &millis)
Reads the actual time.
- int [accelerometerRead](#) (unsigned byte mode, int &aX, int &aY, int &aZ, unsigned long ×tampMs)
Read the values of the accelerometer.
- int [magneticFieldRead](#) (int &bX, int &bY, int &bZ, unsigned long ×tampMs)
Read the values of the magnet field sensor.
- int [pressureRead](#) (unsigned int &pressure, unsigned long ×tampMs)
Read the value of the pressure sensor.
- int [relativeHumidityRead](#) (unsigned int &humidity, unsigned long ×tampMs)
Read the value of the humidity sensor.
- int [ambientTemperatureRead](#) (int &temperature, unsigned long ×tampMs)
Read the value of the ambient temperature sensor.
- int [speechRecognizerStart](#) ()
Start the speech recognizer.
- int [speechRecognizerStop](#) ()
Stop the speech recognizer.
- int [speechRecognizerRead](#) (string &word)
Read one word of the speech recognizer.

5.1.1 Function Documentation

5.1.1.1 int accelerometerRead (unsigned byte *mode*, int & *aX*, int & *aY*, int & *aZ*, unsigned long & *timestampMs*)

Read the values of the accelerometer.

For axis definition see <http://developer.android.com/reference/android/hardware/SensorEvent.html>.

Parameters

| | |
|--------------------|---|
| <i>mode</i> | The following modes are supported. <ul style="list-style-type: none"> • NORMAL: raw values • HIGH_PASSED: Values are high-pass filtered; gravity is not visible. • LOW_PASSED: Values are low-pass filtered; only gravity is visible. |
| <i>aX</i> | Rate of acceleration along x axis [cm/s ²]. |
| <i>aY</i> | Rate of acceleration along y axis [cm/s ²]. |
| <i>aZ</i> | Rate of acceleration along z axis [cm/s ²]. |
| <i>timestampMs</i> | Timestamp of the values in milli seconds. |

Returns

Status

5.1.1.2 int ambientTemperatureRead (int & *temperature*, unsigned long & *timestampMs*)

Read the value of the ambient temperature sensor.

Parameters

| | |
|--------------------|---|
| <i>temperature</i> | Value of temperatur [Celsius * 100]. |
| <i>timestampMs</i> | Timestamp of the values in milli seconds. |

Returns

Status

5.1.1.3 int brightnessRead (unsigned int & *brightness*, unsigned long & *timestampMs*)

Reads the surrounding brightness.

Parameters

| | |
|-------------------|---|
| <i>brightness</i> | Returns the brightness value. |
| <i>timestamp</i> | Returns the timestamp of the brightness value in milli seconds. |

Returns

Status

5.1.1.4 int cameraLightOff ()

Turns off the camera light.

cameraStart must have been called before once.

Returns

Status

5.1.1.5 int cameraLightOn ()

Turns on the camera light.

cameraStart must have been called before once.

Returns

Status

5.1.1.6 int cameraRead (unsigned byte *accuIndex*, unsigned int & *width*, unsigned int & *height*)

Takes a picture from the camera.

cameraStart must have been called before once.

Parameters

| | |
|------------------|--|
| <i>accuIndex</i> | The index of the accu to store the picture in. |
| <i>width</i> | Returns the width of the taken picture. |
| <i>height</i> | Returns the height of the taken picture. |

Returns

Status

5.1.1.7 int cameraStart (unsigned byte *direction*)

Starts the camera.

Before you can use the camera, you have to start it.

Parameters

| | |
|------------------|-----------|
| <i>direction</i> | Direction |
|------------------|-----------|

Returns

Status

5.1.1.8 int cameraStop ()

Stops the camera.

Returns

Status

5.1.1.9 int gyroscopeRead (int & *wX*, int & *wY*, int & *wZ*, unsigned long & *timestampMs*)

Read the values of the gyroscope.

For axis definition see <http://developer.android.com/reference/android/hardware/SensorEvent.html>.

Parameters

| | |
|--------------------|---|
| <i>wX</i> | Rate of rotation around the x axis [deg/s]. |
| <i>wY</i> | Rate of rotation around the y axis [deg/s]. |
| <i>wZ</i> | Rate of rotation around the z axis [deg/s]. |
| <i>timestampMs</i> | Timestamp of the values in milli seconds. |

Returns

Status

5.1.1.10 int magneticFieldRead (int & *bX*, int & *bY*, int & *bZ*, unsigned long & *timestampMs*)

Read the values of the magnet field sensor.

For axis definition see <http://developer.android.com/reference/android/hardware/SensorEvent.html>.

Parameters

| | |
|--------------------|---|
| <i>bX</i> | Rate of magnetic field along x axis [uT * 100]. |
| <i>bY</i> | Rate of magnetic field along y axis [uT * 100]. |
| <i>bZ</i> | Rate of magnetic field along z axis [uT * 100]. |
| <i>timestampMs</i> | Timestamp of the values in milli seconds. |

Returns

Status

5.1.1.11 int microphoneNoiseLevel (unsigned int & *noiseLevel*)

Returns the surrounding noise level.

Parameters

| | |
|-------------------|---|
| <i>noiseLevel</i> | Returns the noise level. It's an absolut value and NOT decibel! |
|-------------------|---|

Returns

Status

5.1.1.12 int pictureAccuBrightestPoint (byte *accuIndex*, unsigned int & *width*, unsigned int & *height*, unsigned int & *brightestX*, unsigned int & *brightestY*, unsigned int & *intensity*)

Finds the brightest point of an image in the picture accu.

Parameters

| | |
|-------------------|---|
| <i>accuIndex</i> | The index of the accu to store the picture in. |
| <i>width</i> | Returns the width of the picture. |
| <i>height</i> | Returns the height of the picture. |
| <i>brightestX</i> | Returns the x coordinate of the brightest point. range 0..width |

| | |
|-------------------|--|
| <i>brightestY</i> | Returns the y coordinate of the brightest point. range 0..height |
| <i>intensity</i> | Returns the intensity of the brightest point. range 0..255 |

Returns

Status

5.1.1.13 int pictureAccuCopy (byte *sourceAccuIndex*, byte *targetAccuIndex*)

Copies a picture from one accu place to another.

Parameters

| | |
|-------------------------|--|
| <i>sourceAccu-Index</i> | |
| <i>targetAccuIndex</i> | |

Returns

Status

5.1.1.14 int pictureAccuCut (byte *accuIndex*, unsigned int *left*, unsigned int *right*, unsigned int *top*, unsigned int *bottom*)

Cuts off some part of an image in the picture accu.

Parameters

| | |
|------------------|--|
| <i>accuIndex</i> | The index of the accu to store the picture in. |
| <i>left</i> | Number of Pixels to cut off from the left. |
| <i>right</i> | Number of Pixels to cut off from the right. |
| <i>top</i> | Number of Pixels to cut off from the top. |
| <i>bottom</i> | Number of Pixels to cut off from the bottom. |

Returns

Status

5.1.1.15 int pictureAccuDarkestPoint (byte *accuIndex*, unsigned int & *width*, unsigned int & *height*, unsigned int & *darkestX*, unsigned int & *darkestY*, unsigned int & *intensity*)

Finds the darkest point of an image in the picture accu.

Parameters

| | |
|------------------|--|
| <i>accuIndex</i> | The index of the accu to store the picture in. |
| <i>width</i> | Returns the width of the picture. |
| <i>height</i> | Returns the height of the picture. |
| <i>darkestX</i> | Returns the x coordinate of the darkest point. range 0..width |
| <i>darkestY</i> | Returns the y coordinate of the darkest point. range 0..height |
| <i>intensity</i> | Returns the intensity of the darkest point. range 0..255 |

Returns

Status

5.1.1.16 `int pictureAccuMeanColor (byte accuIndex, unsigned byte & r, unsigned byte & g, unsigned byte & b)`

Calculates the mean color of an image in the picture accu.

Parameters

| | |
|------------------|--|
| <i>accuIndex</i> | The index of the accu to store the picture in. |
| <i>r</i> | Returns the mean red value. range 0..255 |
| <i>g</i> | Returns the mean green value. range 0..255 |
| <i>b</i> | Returns the mean blue value. range 0..255 |

Returns

Status

5.1.1.17 int pictureAccuShow (unsigned byte *accuIndex*)

Displays a picture from the picture accu.

Parameters

| | |
|------------------|------------------------------------|
| <i>accuIndex</i> | The index of the accu to be shown. |
|------------------|------------------------------------|

Returns

Status

5.1.1.18 int pictureAccuSubtract (byte *accuIndexA*, byte *accuIndexB*)

Subtracts the picture at *accuIndexB* from the one at *accuIndexA*.

The result is stored at *accuIndexA* in picture accu. $r = rA - rB$ $g = gA - gB$ $b = bA - bB$

Parameters

| | |
|-------------------|--|
| <i>accuIndexA</i> | |
| <i>accuIndexB</i> | |

Returns

Status

5.1.1.19 int pressureRead (unsigned int & *pressure*, unsigned long & *timestampMs*)

Read the value of the pressure sensor.

Parameters

| | |
|--------------------|---|
| <i>pressure</i> | Rate of ambient pressure [hPa]. |
| <i>timestampMs</i> | Timestamp of the values in milli seconds. |

Returns

Status

5.1.1.20 int relativeHumidityRead (unsigned int & *humidity*, unsigned long & *timestampMs*)

Read the value of the humidity sensor.

Parameters

| | |
|--------------------|---|
| <i>humidity</i> | Value of relative humidity [rH * 100]. |
| <i>timestampMs</i> | Timestamp of the values in milli seconds. |

Returns

[Status](#)

5.1.1.21 int speechRecognizerRead (string & word)

Read one word of the speech recognizer.

[speechRecognizerStart](#) must have been called before once.

Returns

[Status](#)

5.1.1.22 int speechRecognizerStart ()

Start the speech recognizer.

Returns

[Status](#)

5.1.1.23 int speechRecognizerStop ()

Stop the speech recognizer.

Returns

[Status](#)

5.1.1.24 int timeRead (int & year, unsigned byte & month, unsigned byte & day, unsigned byte & dayOfWeek, unsigned byte & hour, unsigned byte & minute, unsigned byte & second, unsigned int & millis)

Reads the actual time.

Parameters

| | |
|------------------|---|
| <i>year</i> | Returns the year. |
| <i>month</i> | Returns the Month . |
| <i>day</i> | Returns the day. range 1..31 |
| <i>dayOfWeek</i> | Returns the Day of Week . |
| <i>hour</i> | Returns the hour. range 0..23 |
| <i>minute</i> | Returns the minue. range 0..59 |
| <i>second</i> | Returns the second. range 0..59 |

| | |
|---------------|---|
| <i>millis</i> | Returns the milli seconds. range 0..999 |
|---------------|---|

Returns

[Status](#)

Index

- accelerometerRead
 - robotSensors.nxc, [15](#)
- ambientTemperatureRead
 - robotSensors.nxc, [15](#)
- brightnessRead
 - robotSensors.nxc, [15](#)
- cameraLightOff
 - robotSensors.nxc, [15](#)
- cameraLightOn
 - robotSensors.nxc, [16](#)
- cameraRead
 - robotSensors.nxc, [16](#)
- cameraStart
 - robotSensors.nxc, [16](#)
- cameraStop
 - robotSensors.nxc, [16](#)
- Day of Week, [10](#)
- Direction, [7](#)
- gyroscopeRead
 - robotSensors.nxc, [16](#)
- magneticFieldRead
 - robotSensors.nxc, [17](#)
- microphoneNoiseLevel
 - robotSensors.nxc, [17](#)
- Mode, [9](#)
- Month, [11](#)
- pictureAccuBrightestPoint
 - robotSensors.nxc, [17](#)
- pictureAccuCopy
 - robotSensors.nxc, [18](#)
- pictureAccuCut
 - robotSensors.nxc, [18](#)
- pictureAccuDarkestPoint
 - robotSensors.nxc, [18](#)
- pictureAccuMeanColor
 - robotSensors.nxc, [18](#)
- pictureAccuShow
 - robotSensors.nxc, [20](#)
- pictureAccuSubtract
 - robotSensors.nxc, [20](#)
- pressureRead
 - robotSensors.nxc, [20](#)
- relativeHumidityRead
 - robotSensors.nxc, [20](#)
- robotSensors.nxc, [13](#)
 - accelerometerRead, [15](#)
 - ambientTemperatureRead, [15](#)
 - brightnessRead, [15](#)
 - cameraLightOff, [15](#)
 - cameraLightOn, [16](#)
 - cameraRead, [16](#)
 - cameraStart, [16](#)
 - cameraStop, [16](#)
 - gyroscopeRead, [16](#)
 - magneticFieldRead, [17](#)
 - microphoneNoiseLevel, [17](#)
 - pictureAccuBrightestPoint, [17](#)
 - pictureAccuCopy, [18](#)
 - pictureAccuCut, [18](#)
 - pictureAccuDarkestPoint, [18](#)
 - pictureAccuMeanColor, [18](#)
 - pictureAccuShow, [20](#)
 - pictureAccuSubtract, [20](#)
 - pressureRead, [20](#)
 - relativeHumidityRead, [20](#)
 - speechRecognizerRead, [21](#)
 - speechRecognizerStart, [21](#)
 - speechRecognizerStop, [21](#)
 - timeRead, [21](#)
- speechRecognizerRead
 - robotSensors.nxc, [21](#)
- speechRecognizerStart
 - robotSensors.nxc, [21](#)
- speechRecognizerStop
 - robotSensors.nxc, [21](#)
- Status, [8](#)
- timeRead
 - robotSensors.nxc, [21](#)