Wheelphone Library

Generated by Doxygen 1.7.6.1

Fri May 31 2013 16:57:48

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 com.wheelphone.wheelphonelibrary.USBAccessoryManagerMessage.MessageType Enum Reference

Public Attributes

- **READ**
- **ERROR**
- **CONNECTED**
- **DISCONNECTED**

The documentation for this enum was generated from the following file:

- H:/lavoro/local-repo/phonebot-working/_android-side/WheelphoneLibrary/src/com/wheelphone/wheelphonelibrary/USBAccessoryManagerMessage.java

3.2 com.wheelphone.wheelphonelibrary.USBAccessoryManager.RETURN_CODES Enum Reference

Public Attributes

- **DEVICE_MANAGER_IS_NULL**
- **ACCESSORIES_LIST_IS_EMPTY**
- **FILE_DESCRIPTOR_WOULD_NOT_OPEN**
- **PERMISSION_PENDING**

3.2.1 Detailed Description

Enumeration of possible return values for the enable function

The documentation for this enum was generated from the following file:

- H:/lavoro/local-repo/phonebot-working/_android-side/WheelphoneLibrary/src/com/wheelphone/wheelphon
USBAccessoryManager.java

3.3 com.wheelphone.wheelphonelibrary.USBAccessoryManager Class Reference

Classes

- class **ReadThread**
- enum **RETURN_CODES**
- class **USBAccessoryManagerException**

Public Member Functions

- **USBAccessoryManager** (Handler handler, int what)
- **RETURN_CODES enable** (Context context, Intent intent)
- void **disable** (Context context)
- boolean **isConnected** ()
- void **write** (byte[ ] data)
- boolean **isClosed** ()

Package Functions

- void **ignore** (int num)
- int **peek** (byte[ ] array)
- int **available** ()
- int **read** (byte[ ] array)

3.3.1 Detailed Description

A class created to assist in making accessing a USB accessory easier for those that are less familiar with programming in Java, working with threads/handlers/synchronization, and those that are not familiar with the Open Accessory framework interface

Author

Microchip Technology Inc.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 com.wheelphone.wheelphonelibrary.USBAccessoryManager.USBAccessoryManager (Handler *handler*, int *what*)

Public API Creates new USB Accessory Manager

Parameters

handler	The handler where to send USB accessory event messages
what	The "what" value to use for USB accessory event messages

3.3.3 Member Function Documentation

3.3.3.1 int com.wheelphone.wheelphonelibrary.USBAccessoryManager.available ( ) [package]

Indicates the number of bytes that are currently in the read buffer. There will be at least this many bytes to read from the buffer (as long as the accessory has not detach or been closed since the call to this function.

Returns

the number of bytes available in the read queue

3.3.3.2 void com.wheelphone.wheelphonelibrary.USBAccessoryManager.disable (Context *context*)

Disables the USB manager and releases all resources

Parameters

context	The context that the manager was enabled with

3.3.3.3 RETURN_CODES com.wheelphone.wheelphonelibrary.USBAccessoryManager.enable (Context *context*, Intent *intent*)

Enables the

Parameters

context	The context that the USB manager should register to

Returns

RETURN_CODES (p. 5) - the status of the enable request

3.3.3.4 void com.wheelphone.wheelphonelibrary.USBAccessoryManager.ignore (int *num*) `[package]`

I/O API discards the specified number of bytes from the internal read buffer

Parameters

num	the number of bytes to discard

3.3.3.5 boolean com.wheelphone.wheelphonelibrary.USBAccessoryManager.isConnected ( )

Describes if an accessory is attached or not

Returns

boolean - true if one is attached, false otherwise

3.3.3.6 int com.wheelphone.wheelphonelibrary.USBAccessoryManager.peek (byte[ ] *array*) `[package]`

fills the array with data from the read buffer without discarding it.

Parameters

array	the buffer to fill

Returns

the number of bytes copied from the buffer

3.3.3.7 int com.wheelphone.wheelphonelibrary.USBAccessoryManager.read (byte[ ] *array*) `[package]`

Reads bytes from the read buffer, removing them from the buffer once read

Parameters

array	where to copy the data

Returns

the number of bytes copied (maximum will be the length of the array param

3.3.3.8 void com.wheelphone.wheelphonelibrary.USBAccessoryManager.write (byte[ ] *data*)

Writes data to the accessory

Parameters

data	the data to write

Exceptions

InterruptedException	

The documentation for this class was generated from the following file:

- H:/lavoro/local-repo/phonebot-working/_android-side/WheelphoneLibrary/src/com/wheelphone/wheelphonelibrary/USBAccessoryManager.java

3.4 com.wheelphone.wheelphonelibrary.USBAccessoryManager.USBAccessoryManagerException Class Reference

Public Member Functions

- **USBAccessoryManagerException** (String message)
- String **toString** ()

Package Attributes

- String **errorMessage**

3.4.1 Detailed Description

Exception definition section Exception that can be thrown by the manager - currently not used

The documentation for this class was generated from the following file:

- H:/lavoro/local-repo/phonebot-working/_android-side/WheelphoneLibrary/src/com/wheelphone/wheelphonelibrary/USBAccessoryManager.java

3.5 com.wheelphone.wheelphonelibrary.USBAccessoryManagerMessage Class Reference

Classes

- enum **MessageType**

Public Member Functions

- **USBAccessoryManagerMessage** (**MessageType** type)
- **USBAccessoryManagerMessage** (**MessageType** type, byte[] data)
- **USBAccessoryManagerMessage** (**MessageType** type, byte[] data, UsbAccessory accessory)
- **USBAccessoryManagerMessage** (**MessageType** type, UsbAccessory accessory)

Public Attributes

- **MessageType type**
- String **text** = null
- byte[] **data** = null
- UsbAccessory **accessory** = null

3.5.1 Detailed Description

Basic Message class for the **USBAccessoryManager** (p. 6). This is used to send messages from the USB Accessory's read thread to the GUI thread to notify the GUI thread of various USB Accessory events (like data available or device attachment).

Author

Microchip Technologies Inc.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 com.wheelphone.wheelphonelibrary.USBAccessoryManagerMessage.USBAccessoryManagerMessage (MessageType *type*)

Creates new message of specified type

Parameters

type	The type of this message

3.5.2.2 com.wheelphone.wheelphonelibrary.USBAccessoryManagerMessage.USBAccessoryManagerMessage (MessageType *type,* byte[ ] *data*)

Creates a new message of specified type with specified data

Parameters

type	The type of this message
data	The data associated with this message

3.5.2.3 com.wheelphone.wheelphonelibrary.USBAccessoryManagerMessage.USBAccessoryManagerMessage (MessageType *type,* byte[ ] *data,* UsbAccessory *accessory*)

Creates a new message of specified type with specified data

Parameters

type	The type of this message
data	The data associated with this message
accessory	The accessory associated with this message

3.5.2.4 com.wheelphone.wheelphonelibrary.USBAccessoryManagerMessage.USBAccessoryManagerMessage (MessageType *type,* UsbAccessory *accessory*)

Creates a new message of specified type with specified data

Parameters

type	The type of this message
accessory	The accessory associated with this message

The documentation for this class was generated from the following file:

- H:/lavoro/local-repo/phonebot-working/_android-side/WheelphoneLibrary/src/com/wheelphone/wheelphonelibrary/USBAccessoryManagerMessage.java

3.6 com.wheelphone.wheelphonelibrary.WheelphoneRobot Class Reference

Classes

- class **communicationTask**

Public Member Functions

- **WheelphoneRobot** (Activity a, Context c)

  Class constructor.
- void **startUSBCommunication** ()

  To be inserted into the "onStart" function of the main activity class.
- void **resumeUSBCommunication** ()

  To be inserted into the "onResume" function of the main activity class.
- void **pauseUSBCommunication** ()

  To be inserted into the "onPause" function of the main activity class.
- void **setSpeed** (int l, int r)

  Set the new left and right speeds for the robot. The new data will be actually sent to the robot when "sendCommandsToRobot" is called the next time within the timer communication task (50 ms cadence). This means that the robot speed will be updated after at most 50 ms (if the task isn't delayed by the system).
- void **setLeftSpeed** (int l)

  Set the new left speed for the robot. For more details refer to "setSpeed".
- void **setRightSpeed** (int r)

  Set the new right speed for the robot. For more details refer to "setSpeed".
- void **setRawSpeed** (int l, int r)

  Set the new left and right speeds for the robot. For more details refer to "setSpeed".
- void **setRawLeftSpeed** (int l)

  Set the new left speed for the robot. For more details refer to "setSpeed".
- void **setRawRightSpeed** (int r)

  Set the new right speed for the robot. For more details refer to "setSpeed".
- void **enableSpeedControl** ()

  Enable speed control on the robot (controller based on speed measure with back-emf).
- void **disableSpeedControl** ()

  Disable speed control on the robot.
- void **enableSoftAcceleration** ()

  Enable soft acceleration on the robot; this is useful when the robot is started fast (from standstill) because it avoid the robot to wheelie.
- void **disableSoftAcceleration** ()

  Disable soft acceleration on the robot.
- void **enableObstacleAvoidance** ()

  Enable obstacle avoidance onboard.
- void **disableObstacleAvoidance** ()

  Disable obstacle avoidance onboard.
- void **enableCliffAvoidance** ()

  Enable cliff avoidance onboard; when a cliff is detected the robot is stopped until this flag is reset.
- void **disableCliffAvoidance** ()

  Disable cliff avoidance onboard.
- void **calibrateSensors** ()

Start the calibration of all the sensors. Use "isCalibrating" to know when the calibration is done.

- int **getBatteryRaw** ()

  Returns the sampled value of the battery.
- float **getBatteryVoltage** ()

  Returns the current battery voltage.
- byte **getFlagStatus** ()

  Returns the flag byte that the robot set/clear itself.
- boolean **isCharging** ()

  Returns the charging status.
- boolean **isCharged** ()

  Returns the battery charged status.
- int **getLeftEncoder** ()

  The value of the left encoder returned from the robot. The encoders values are based on the measured speed not on a real encoder device. The values given by the robot are the current encoders values, not the absolute value.
- int **getRightEncoder** ()

  The value of the right encoder returned from the robot. For more details refer to "getLeftEncoder".
- int[ ] **getFrontProxs** ()

  The robot has 4 front proximity sensors positioned as follows:
- int **getFrontProx** (int ind)

  Return the corresponding front proximity sensor value. For more details refer to "getFrontProxs".
- int[ ] **getFrontAmbients** ()

  The robot has 4 front ambient sensors, actually they are the front proximity sensors that can measure also the ambient light. The higher the value the lighter the environment.
- int **getFrontAmbient** (int ind)

  Return the corresponding front ambient sensor value. For more details refer to "getFrontAmbients".
- int[ ] **getGroundProxs** ()

  The robot has 4 ground proximity sensors positioned as follows:
- int **getGroundProx** (int ind)

  Return the corresponding ground proximity sensor value. For more details refer to "getGroundProxs".
- int[ ] **getGroundAmbients** ()

  The robot has 4 ground ambient sensors, actually they are the ground proximity sensors that can measure also the ambient light. The higher the value the lighter the environment.
- int **getGroundAmbient** (int ind)

  Return the corresponding ground ambient sensor value. For more details refer to "getGroundAmbients".
- boolean **isUSBConnected** ()

  Indicate whether the robot is connected (and exchanging packets) with the phone or not.
- void **setUSBCommunicationTimeout** (int ms)

This timeout sets how much to wait for a response from the robot before changing to a disconnected state.

- boolean **isCalibrating** ()

  Tell whether the calibration is still in progress or not.
- double[ ] **getOdometry** ()

  Return the odometry information resulting from the encoders values received by the robot. The positive X axis is pointing forward and the positive X axis is pointing to the left side of the robot.
- double **getOdometryX** ()

  Return the x absolute position in mm. For more information refer to "getOdometry".
- double **getOdometryY** ()

  Return the y absolute position in mm. For more information refer to "getOdometry".
- double **getOdometryTheta** ()

  Return the theta absolute angle in radians. For more information refer to "getOdometry".
- void **setOdometry** (double x, double y, double t)

  Set/reset odometry components. For more information refer to "getOdometry".
- void **setOdometryParameters** (double dl, double dr, double wb)

  Set/reset odometry parameters.
- void **appendLog** (String text)

Package Attributes

- int **rightEncoder** = 0
- int **rSpeed** = 0
- double **rightEncSum** = 0.0
- double **rightEncSumPrev** = 0.0
- double **finalTime** = 0.0
- double **totalTime** = 0.0

3.6.1 Constructor & Destructor Documentation

3.6.1.1 com.wheelphone.wheelphonelibrary.WheelphoneRobot.WheelphoneRobot (Activity *a*, Context *c*)

Class constructor.

Parameters

a	pass the main activity instance (this)
c	pass the main activity instance (this)

Returns

WheelphoneRobot (p. 11) instance

3.6.2 Member Function Documentation

3.6.2.1 void com.wheelphone.wheelphonelibrary.WheelphoneRobot.calibrateSensors ( )

Start the calibration of all the sensors. Use "isCalibrating" to know when the calibration is done.

Returns

none

3.6.2.2 void com.wheelphone.wheelphonelibrary.WheelphoneRobot.disableCliffAvoidance ( )

Disable cliff avoidance onboard.

Returns

none

3.6.2.3 void com.wheelphone.wheelphonelibrary.WheelphoneRobot.disableObstacleAvoidance ( )

Disable obstacle avoidance onboard.

Returns

none

3.6.2.4 void com.wheelphone.wheelphonelibrary.WheelphoneRobot.disableSoftAcceleration ( )

Disable soft acceleration on the robot.

Returns

none

3.6.2.5 void com.wheelphone.wheelphonelibrary.WheelphoneRobot.disableSpeedControl ( )

Disable speed control on the robot.

Returns

none

3.6.2.6 void com.wheelphone.wheelphonelibrary.WheelphoneRobot.enableCliffAvoidance ( )

Enable cliff avoidance onboard; when a cliff is detected the robot is stopped until this flag is reset.

Returns

none

3.6.2.7 void com.wheelphone.wheelphonelibrary.WheelphoneRobot.enableObstacleAvoidance ( )

Enable obstacle avoidance onboard.

Returns

none

3.6.2.8 void com.wheelphone.wheelphonelibrary.WheelphoneRobot.enableSoftAcceleration ( )

Enable soft acceleration on the robot; this is useful when the robot is started fast (from standstill) because it avoid the robot to wheelie.

Returns

none

3.6.2.9 void com.wheelphone.wheelphonelibrary.WheelphoneRobot.enableSpeedControl ( )

Enable speed control on the robot (controller based on speed measure with back-emf).

Returns

none

3.6.2.10 int **com.wheelphone.wheelphonelibrary.WheelphoneRobot.getBatteryRaw ( )**

Returns the sampled value of the battery.

Returns

battery level (from 0 to 100)

3.6.2.11 float **com.wheelphone.wheelphonelibrary.WheelphoneRobot.getBatteryVoltage ( )**

Returns the current battery voltage.

Returns

battery voltage (from 3.7 to 4.2 volts)

3.6.2.12 byte **com.wheelphone.wheelphonelibrary.WheelphoneRobot.getFlagStatus ( )**

Returns the flag byte that the robot set/clear itself.

Returns

flag byte:

- bit 5: 1 => robot is charging, 0 => robot not charging
- bit 6: 1 => robot completely charged, 0 => robot not completely charged
- others bits not used

3.6.2.13 int **com.wheelphone.wheelphonelibrary.WheelphoneRobot.getFrontAmbient (** int *ind* **)**

Return the corresponding front ambient sensor value. For more details refer to "getFrontAmbients".

Returns

sensor value

3.6.2.14 int [] **com.wheelphone.wheelphonelibrary.WheelphoneRobot.getFrontAmbients ( )**

The robot has 4 front ambient sensors, actually they are the front proximity sensors that can measure also the ambient light. The higher the value the lighter the environment.

Returns

array of size 4 containing the sensors values

3.6.2.15 int **com.wheelphone.wheelphonelibrary.WheelphoneRobot.getFrontProx** (int *ind*)

Return the corresponding front proximity sensor value. For more details refer to "getFrontProxs".

Returns

sensor value

3.6.2.16 int [ ] **com.wheelphone.wheelphonelibrary.WheelphoneRobot.getFrontProxs** ( )

The robot has 4 front proximity sensors positioned as follows:

```
1 2
0 3
```

The higher the value the nearer the object in front of the sensor.

Returns

array of size 4 containing the sensors values

3.6.2.17 int **com.wheelphone.wheelphonelibrary.WheelphoneRobot.getGroundAmbient** (int *ind*)

Return the corresponding ground ambient sensor value. For more details refer to "getGroundAmbients".

Returns

sensor value

3.6.2.18 int [ ] **com.wheelphone.wheelphonelibrary.WheelphoneRobot.getGroundAmbients** ( )

The robot has 4 ground ambient sensors, actually they are the ground proximity sensors that can measure also the ambient light. The higher the value the lighter the environment.

Returns

array of size 4 containing the sensors values

3.6.2.19 int com.wheelphone.wheelphonelibrary.WheelphoneRobot.getGroundProx (int *ind*)

Return the corresponding ground proximity sensor value. For more details refer to "getGroundProxs".

Returns

sensor value

3.6.2.20 int [ ] com.wheelphone.wheelphonelibrary.WheelphoneRobot.getGroundProxs ( )

The robot has 4 ground proximity sensors positioned as follows:

```
1 2
0 3
```

The higher the value the darker the object in front of the sensor.

Returns

array of size 4 containing the sensors values

3.6.2.21 int com.wheelphone.wheelphonelibrary.WheelphoneRobot.getLeftEncoder ( )

The value of the left encoder returned from the robot. The encoders values are based on the measured speed not on a real encoder device. The values given by the robot are the current encoders values, not the absolute value.

Returns

left encoder (positive or negative)

3.6.2.22 double [ ] com.wheelphone.wheelphonelibrary.WheelphoneRobot.getOdometry ( )

Return the odometry information resulting from the encoders values received by the robot. The positive X axis is pointing forward and the positive X axis is pointing to the left side of the robot.

Returns

array of length 3 containing sequentially x position (mm), y position (mm), theta (radians).

3.6.2.23 double com.wheelphone.wheelphonelibrary.WheelphoneRobot.getOdometryTheta ( )

Return the theta absolute angle in radians. For more information refer to "getOdometry".

Returns

theta (radians)

3.6.2.24 double com.wheelphone.wheelphonelibrary.WheelphoneRobot.getOdometryX ( )

Return the x absolute position in mm. For more information refer to "getOdometry".

Returns

x position (mm)

3.6.2.25 double com.wheelphone.wheelphonelibrary.WheelphoneRobot.getOdometryY ( )

Return the y absolute position in mm. For more information refer to "getOdometry".

Returns

y position (mm)

3.6.2.26 int com.wheelphone.wheelphonelibrary.WheelphoneRobot.getRightEncoder ( )

The value of the right encoder returned from the robot. For more details refer to "getLeftEncoder".

Returns

right encoder (positive or negative)

3.6.2.27 boolean com.wheelphone.wheelphonelibrary.WheelphoneRobot.isCalibrating ( )

Tell whether the calibration is still in progress or not.

Returns

true (calibration in progress), false otherwise

3.6.2.28 boolean com.wheelphone.wheelphonelibrary.WheelphoneRobot.isCharged ( )

Returns the battery charged status.

Returns

true if battery charged, false otherwise

3.6.2.29 boolean com.wheelphone.wheelphonelibrary.WheelphoneRobot.isCharging ( )

Returns the charging status.

Returns

true if charging, false otherwise

3.6.2.30 boolean com.wheelphone.wheelphonelibrary.WheelphoneRobot.isUSBConnected ( )

Indicate whether the robot is connected (and exchanging packets) with the phone or not.

Returns

true (if robot connected), false otherwise

3.6.2.31 void com.wheelphone.wheelphonelibrary.WheelphoneRobot.pauseUSBCommunication ( )

To be inserted into the "onPause" function of the main activity class.

Returns

none

3.6.2.32 void com.wheelphone.wheelphonelibrary.WheelphoneRobot.resumeUSBCommunication ( )

To be inserted into the "onResume" function of the main activity class.

Returns

none

3.6.2.33 void com.wheelphone.wheelphonelibrary.WheelphoneRobot.setLeftSpeed (int *l*)

Set the new left speed for the robot. For more details refer to "setSpeed".

Parameters

l	left speed given in mm/s

Returns

none

3.6.2.34 void com.wheelphone.wheelphonelibrary.WheelphoneRobot.setOdometry (double *x*, double *y*, double *t*)

Set/reset odometry components. For more information refer to "getOdometry".

Parameters

x	x position (mm)
y	y position (mm)
t	theta angle (radians)

Returns

none

3.6.2.35 void com.wheelphone.wheelphonelibrary.WheelphoneRobot.-setOdometryParameters (double *dl*, double *dr*, double *wb*)

Set/reset odometry parameters.

Parameters

dl	left wheel diamater (m)
dr	right wheel diamater (m)
wb	wheels distance (m)

Returns

none

3.6.2.36 void com.wheelphone.wheelphonelibrary.WheelphoneRobot.setRawLeftSpeed (int *l*)

Set the new left speed for the robot. For more details refer to "setSpeed".

Parameters

l	left speed (range is from -127 to 127)

Returns

none

3.6.2.37 void com.wheelphone.wheelphonelibrary.WheelphoneRobot.setRawRightSpeed (int *r*)

Set the new right speed for the robot. For more details refer to "setSpeed".

Parameters

r	right speed (range is from -127 to 127)

Returns

none

3.6.2.38 void com.wheelphone.wheelphonelibrary.WheelphoneRobot.setRawSpeed (int *l*, int *r*)

Set the new left and right speeds for the robot. For more details refer to "setSpeed".

Parameters

l	left speed (range is from -127 to 127)
r	right speed (range is from -127 to 127)

Returns

none

3.6.2.39 void com.wheelphone.wheelphonelibrary.WheelphoneRobot.setRightSpeed (int *r*)

Set the new right speed for the robot. For more details refer to "setSpeed".

Parameters

r	right speed given in mm/s

Returns

none

3.6.2.40 void com.wheelphone.wheelphonelibrary.WheelphoneRobot.setSpeed (int *l*, int *r*)

Set the new left and right speeds for the robot. The new data will be actually sent to the robot when "sendCommandsToRobot" is called the next time within the timer communication task (50 ms cadence). This means that the robot speed will be updated after at most 50 ms (if the task isn't delayed by the system).

Parameters

l	left speed given in mm/s
r	right speed given in mm/s

Returns

none

3.6.2.41 void com.wheelphone.wheelphonelibrary.WheelphoneRobot.setUSBCommunicationTimeout (int *ms*)

This timeout sets how much to wait for a response from the robot before changing to a disconnected state.

Parameters

timeout	in milliseconds

Returns

none

3.6.2.42 void com.wheelphone.wheelphonelibrary.WheelphoneRobot.startUSBCommunication ( )

To be inserted into the "onStart" function of the main activity class.

Returns

none

The documentation for this class was generated from the following file:

- H:/lavoro/local-repo/phonebot-working/_android-side/WheelphoneLibrary/src/com/wheelphone/wheelphonelibrary/**WheelphoneRobot.java**

Chapter 4

File Documentation

4.1 H:/lavoro/local-repo/phonebot-working/_android-side/Wheelphone-Library/src/com/wheelphone/wheelphonelibrary/Wheelphone-Robot.java File Reference

Main Wheelphone class.

Classes

- class **com.wheelphone.wheelphonelibrary.WheelphoneRobot**
- class **com.wheelphone.wheelphonelibrary.WheelphoneRobot.communicationTask**

4.1.1 Detailed Description

Main Wheelphone class.

Author

Stefano Morgani <stefano@gctronic.com>

Version

1.0

Date

29.05.13

Copyright

GNU GPL v3

The WheelphoneRobot class is the main class that need to be instantiated in the application in order to communicate with the robot (receive sensors data and send commands). The low-level communication with the robot (packets exchange) is handled internally by this class.

Index