SynchroTime

Generated by Doxygen 1.8.6

Wed Sep 16 2020 19:08:11

Contents

Chapter 1

SynchroTime - Command-line application (CLI-app)

SynchroTime: Command-line client for adjust the exact time and calibrating the RTC DS3231 module via the serial interface (UART).

Author

SergejBre sergej1@email.ua

Chapter 2

SynchroTime - Command-line client for adjust the exact time and calibrating the RTC DS3231 module

Motivation

The real-time clock module on the DS3231 chip has proven itself well in work with microcontrollers Arduino, Raspberry Pi, etc. According to the declared specification, it has thermal correction, so that the drift of the clock time is within ±2 ppm (ca. 1 minute per year). But a large number of modules on the market do not meet the accuracy declared by the manufacturer, which is undoubtedly upsetting. Nevertheless, the manufacturer has provided for the possibility of correcting the drift of the clock time, which is associated with the aging of the oscillator crystal in the range from -12.8 to +12.7 ppm. This correction value can be written to one of the registers on the DS3231 (See the datasheet for exact ppm values). In addition, the manufacturer has provided a the energy-independent flash memory AT24C256 in the module, into which calibration parameters and correction factors can be placed. The tool below can automatically calibrate the DS3231 module.

About the app

- Command-line application is used for fine tuning and calibration of the RTC DS3231 module.
- The application allows you to:
  - Synchronize the time of the RTC DS3231 with your computer time;
  - Correct the time drift of the DS3231-RTC clock. The algorithm performs correction in the range from -12.8 to +12.7 ppm;
  - The application allows you to evaluate the accuracy and reliability of the RTC oscillator for a particular sample, as well as the chances of successful correction in case of significant time drift;
  - Automatically save parameters and calibration data to the energy-independent flash memory of the type AT24C256. In case there is a power failure to the module.
- Developed in pure Qt, no third party libraries.
- Cross-platform application implementation (Linux and Windows).
- The client communicates with the Arduino server via the serial interface (UART). The application allows you to easily select a serial port for communication with the server and save the port number in the program settings.
- Command Help `~/SynchroTime$ ./synchroTime -h`

Using the app

1. First, you need to load a sketch into Arduino from the `arduino/synchro_RTC.ino` project directory and connect the RTC DS3231 module according to the circuit shown in the specification. Connect your Arduino

to your computer via a free USB port. If there is a necessary driver in the system, a new virtual serial port will appear in the system (under Linux it will be /dev/ttyUSBx, under Windows - COMx). To find the name of this port, call the application with the -d (–discovery) switch: “‘ ∼/SynchroTime$ ./synchroTime -d Serial Port : ttyUSB1 Description : USB2.0-Serial Manufacturer: 1a86 Vendor ID : 1a86 Product ID : 7523 System Locat: /dev/ttyUSB1 Busy : No

Serial Port : ttyUSB0 Description : USB2.0-Serial Manufacturer: 1a86 Vendor ID : 1a86 Product ID : 7523 System Locat: /dev/ttyUSB0 Busy : No

A total of 2 serial ports were found. “‘ And under the Windows OS “‘ C:>synchroTime -d Serial Port : COM5 Description : USB-SERIAL CH340 Manufacturer: wch.cn Vendor ID : 1a86 Product ID : 7523 System Locat: Busy : No

Serial Port : COM3 Description : Agere Systems HDA Modem Manufacturer: Agere Vendor ID : 11c1 Product ID : 1040 System Locat: Busy : No

A total of 2 serial ports were found. “‘

2. To select a virtual Serial Port, enter its system name after the command -p <portName>. The app will automatically create a configuration file, and the next call will contact the selected port. “‘ ∼/SynchroTime$ ./synchroTime -p ttyUSB0 Added new serial interface ttyUSB0. “‘ And under the Windows OS “‘ C:>synchroTime -p COM5 Added new serial interface COM5. “‘

3. Use the -i (–information) command to get the current information from the DS3231 module. If everything is connected correctly, then you will get the current time of both clocks, the difference between the clocks in milliseconds (with an accuracy of ±2 ms), the value written in the offset register and the calculated time drift value in ppm. If the offset register and time drift are zero, then the DS3231 has not yet been calibrated (see step 5.) “‘ ∼/SynchroTime$ ./synchroTime -i DS3231 clock time 1598896552596 ms: 31.08.2020 19-:55:52.596 System local time 1598896589772 ms: 31.08.2020 19:56:29.772 Difference between -37176 ms Offset reg. in ppm 0 ppm Time drift in ppm -8.78162 ppm last adjust of time 1594663200000 ms: 13.07.2020 20:00:00.000 “‘

4. To set the exact time, use the -a (–adjust) command. The module clock will be synchronized with the computer time with an accuracy of ±1 ms. After updating the time, the date of the time setting will be recorded in the module’s memory, which will allow later to determine the exact drift of the clock time. “‘ ∼/SynchroTime$ ./synchroTime -a System local time Mo. 31 Aug. 2020 20:02:52.000 Request for adjustment completed successfully. “‘

5. To calibrate the clock of the DS3231 module, enter the -c (–calibration) command. For the successful execution of this procedure, the module must be activated (see point 4.) and it is necessary that enough time has passed so that the calculated value of the clock drift is well distinguishable from the rounding error (ca 55 hours or 2.3 days, see part **Discussion**). The algorithm of the program will calculate the amount of drift of the clock time and the correction factor, which will be written into the offset register. The clock time will also be updated. If the calibration is successful, the current time, drift and correction factor will be displayed, as in the screenshot. “‘ ∼/SynchroTime$ ./synchroTime -c System local time Mo. 31 Aug. 2020 20:04:14.000 Offset last value 0 Time drift in ppm -2.11938 ppm Offset new value -21 Request for calibration completed successfully. “‘

6. To reset the offset register to its default value and clear the module’s memory of calibration data, enter the -r (–reset) command. The default value will be written to the register, and memory cells will be overwritten with bytes with 0xFF. “‘ ∼/SynchroTime$ ./synchroTime -r

   Request for reset completed successfully. “‘

7. Use the -s (–setreg) command to write the new value to the offset register of the DS3231. Warning: it makes sense to do this operation only in case of resetting all calibration data (see step 6). “‘ ∼/SynchroTime$ ./synchroTime -s

   Request for SetRegister completed successfully. “‘

Specification

- The application allows you to adjust the time with an accuracy of ±1 ms.

- The application allows you to control the time difference between the DS3231 module and the computer with an accuracy of ±2 ms.
- The application allows you to calibrate the module clock within the range from -12.8 to +12.7 ppm.
- The application communicates with the Arduino server through any virtual serial interface (UART). The Baud Rate is assumed unchanged and equals 115200 bps.
- The Arduino is in turn connected to the Precision RTC DS3231 module via the I²C-interface: A4 (SDA), A5 (SCL) pins (SDA - data line and SCL - clock line).
- The interrupt on the port D2 (or D3) serves to count milliseconds by the internal Arduino counter millis.
- The suggested connection to the DS3231 module is according to the Circuit below.

Recommended System Requirements

- For correct work your system time required to be synchronized with NTP. Only in this case the program will work according to the declared specifications. Under Linux, the ntp service is installed by the following command “‘ ∼/$ sudo apt-get install ntp “‘
- Check the correct operation of the service ntp by running the command “‘ ∼/$ ntpq -p

remote refid st t when poll reach delay offset jitter

+gromit.nocabal. 131.188.3.222 2 u 64 64 377 27.218 -3.906 5.643 ∗www.kashra.com .DCFa. 1 u 3 64 377 42.583 -5.584 4.940 +ext01.epiontis. 130.149.17.8 2 u 2 64 177 18.668 -7.450 5.801 +ntp1.hetzner.de 124.216.164.14 2 u 11 64 377 25.011 -5.987 6.489 +chilipepper.can 134.71.66.21 2 u 74 64 376 26.689 -5.881 4.974 “‘

- Look for a table entry ∗: table values offset and jitter (ms), they should be as minimal as possible `max[offset ± jitter] <= 10ms`. If this is not the case, adjust the configuration file `/etc/ntp.conf` in which you enter the local time servers.
- Windows OS has its own specifics. Windows `W32tm` Time Service synchronizes time once a week, which is not enough for fine tuning and calibration. Therefore, it is necessary to adjust the computer time manually before setting up and calibrating, or using the subtleties of the settings in the system registry.

Installing the app

- According to the working platform, download the appropriate archive with the command-line application from the project page.
- Unpack it to your home directory with write access, as the application retains its settings. “‘ ∼/$ tar -x -j -f synchroTime_v.1.0.0_i386.tar.bz2 “‘
- Run the application according to the instructions in the section **Using the app**. “‘ ∼/$ cd SynchroTime ∼/SynchroTime$ ./synchroTime -h “‘

Discussion

Dependencies

Name	Version	Comment
Qt lib 32bit	>= 5.5.1	Didn’t test with older versions, but it may work

Qt lib 64bit	>= 5.6	Didn't test with older versions, but it may work
C++ compiler	supporting C++11 (i.e. gcc 4.6+)	
Arduino IDE	>= 1.8.13	!Replace compilation flags from -Os to -O2

“‘ :~/SynchroTime/build$ ldd synchroTime libQt5SerialPort.so.5 => ./lib/libQt5SerialPort.so.5 libQt5Core.so.5 => ./lib/libQt5Core.so.5 ... libicui18n.so.54 => ./lib/libicui18n.so.54 libicuuc.so.54 => ./lib/libicuuc.so.54 libicudata.so.-54 => ./lib/libicudata.so.54 “‘

Compilation on Linux

- `sudo apt-get install build-essential qt5-default qt5-qmake gdb git`
- `git clone https://github.com/SergejBre/SynchroTime.git`
- `cd ./SynchroTime`
- `QT_SELECT=5 qmake SynchroTime.pro`
- `make && make clean`

License

SynchroTime is licensed under [MIT](LICENSE).

Chapter 3

Todo List

Member InterfaceSP::handleError (QSerialPort::SerialPortError error)

Member InterfaceSP::handleReadyRead ()

Member setCommandLineParser (QCommandLineParser &parser)

Member Settings::~Settings ()

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 6

Class Index

6.1 Class List

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

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Chapter 8

Namespace Documentation

8.1 Ui Namespace Reference

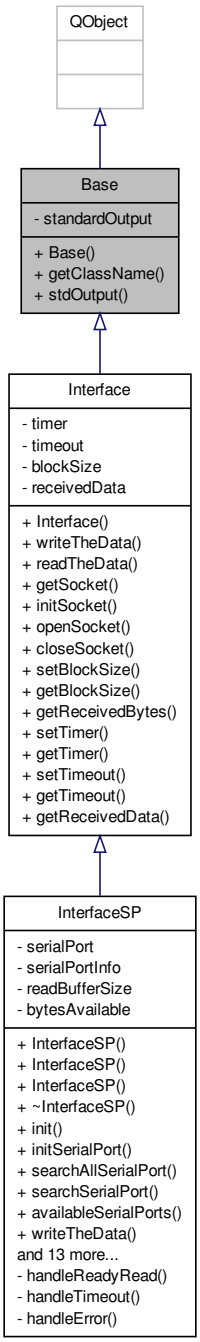
Chapter 9

Class Documentation

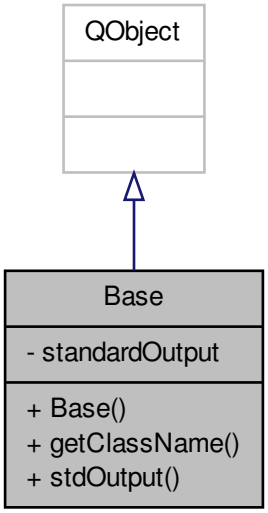
9.1 Base Class Reference

```
#include <base.h>
```

Inheritance diagram for Base:



Collaboration diagram for Base:



Public Member Functions

- Base (QObject ∗parent=0)
  Base::Base.
- virtual QString getClassName (void)
  Base::getClassName.
- QTextStream & stdOutput (void)
  Base::stdOutput.

Private Attributes

- QTextStream standardOutput
  Output stream for all messages.

9.1.1 Constructor & Destructor Documentation

9.1.1.1 Base::Base (QObject ∗ *parent* = 0) [explicit]

Base::Base.

The Constructor for class Basse

9.1.2 Member Function Documentation

9.1.2.1 QString Base::getClassName (void) [virtual]

Base::getClassName.

Get class name as string

Returns

className of the type QString.

9.1.2.2 QTextStream & Base::stdOutput (void)

Base::stdOutput.

Output stream for all messages.

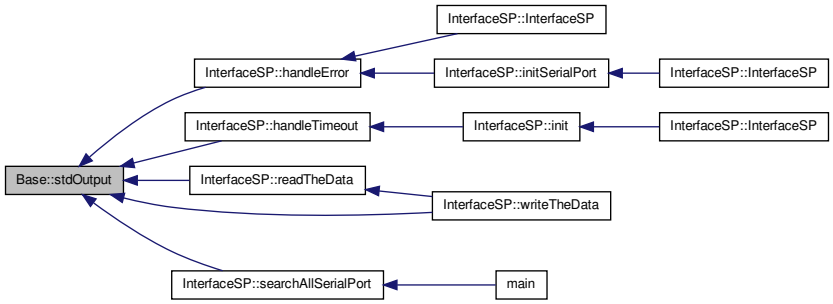
Returns

standardOutput of the type QTextStream&.

References standardOutput.

Referenced by InterfaceSP::handleError(), InterfaceSP::handleTimeout(), InterfaceSP::readTheData(), InterfaceSP::searchAllSerialPort(), and InterfaceSP::writeTheData().

Here is the caller graph for this function:



9.1.3 Member Data Documentation

9.1.3.1 QTextStream Base::standardOutput [private]

Output stream for all messages.

Referenced by stdOutput().

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

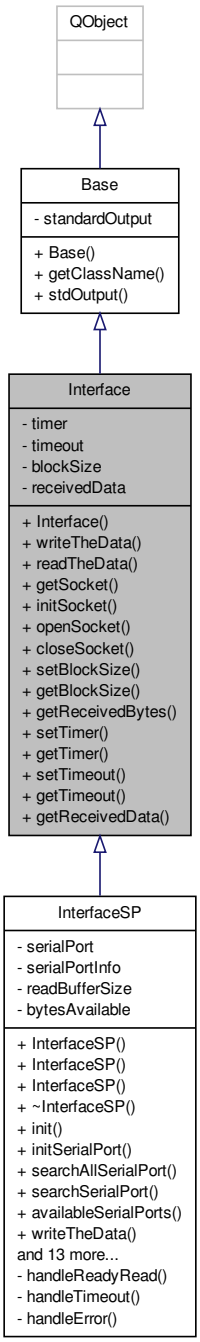
- base.h
- base.cpp

9.2 Interface Class Reference

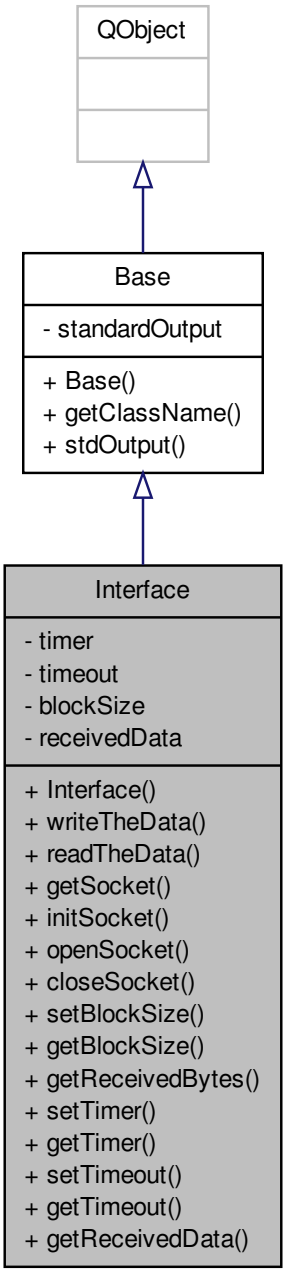
Interface The base class Interface provides for the communication between Bootloader Client and Host Device.

`#include <interface.h>`

Inheritance diagram for Interface:



Collaboration diagram for Interface:



Signals

- void infoMessage (const QVariant &message) const
- void debugMessage (const QVariant &message) const
- void errorMessage (const QVariant &message) const

Public Member Functions

- Interface (QObject ∗parent=0)

  Interface::Interface Constructor of the class Interface for creating objects with default values.
- virtual qint64 writeTheData (const QByteArray &)=0
- virtual bool readTheData (const quint32, const quint32 bytes=0U)=0
- virtual QIODevice ∗ getSocket (void)=0
- virtual void initSocket (void)=0
- virtual bool openSocket (void)=0
- virtual void closeSocket (void)=0
- void setBlockSize (const quint16 size)

  Interface::setBlockSize.
- quint16 getBlockSize (void) const

  Interface::getBlockSize.
- quint32 getReceivedBytes (void) const

  Interface::getReceivedBytes.
- void setTimer (QTimer ∗timer)

  Interface::setTimer.
- QTimer ∗ getTimer (void)

  Interface::getTimer.
- void setTimeout (quint32 time)

  Interface::setTimeout.
- quint32 getTimeout (void) const

  Interface::getTimeout.
- QByteArray & getReceivedData (void)

  Interface::getReceivedData.

Private Attributes

- QTimer ∗ timer

  Timer for Time-out method.
- quint32 timeout

  The timeout interval in milliseconds.
- quint16 blockSize

  Size of the data block for the communication protocol.
- QByteArray receivedData

  The received data.

9.2.1 Detailed Description

Interface The base class Interface provides for the communication between Bootloader Client and Host Device.

Base class interface provides two virtual methods for sending and receiving the instructions and the data. (Interface-::writeTheData Interface::readTheData) These virtual methods are implemented in the other derived classes.

9.2.2 Constructor & Destructor Documentation

9.2.2.1 QT_USE_NAMESPACE Interface::Interface (QObject ∗ *parent* = 0) [explicit]

Interface::Interface Constructor of the class Interface for creating objects with default values.

Create a instance of the class and call following init methods:

1. Interface::init()

Parameters

in	*parent*	Pointer to the parent object used for QObject

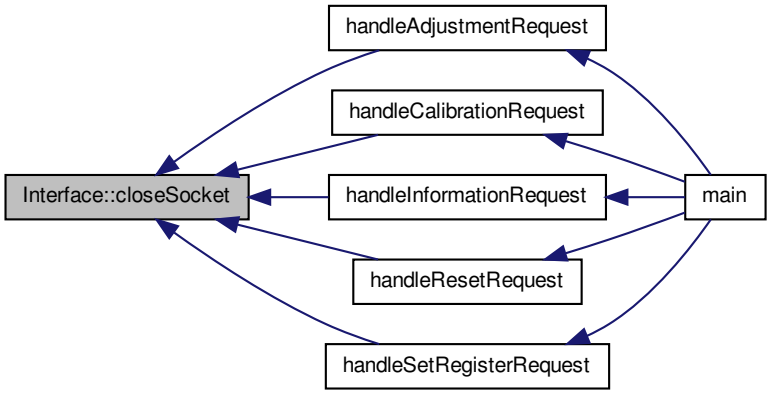
9.2.3 Member Function Documentation

9.2.3.1 virtual void Interface::closeSocket (void) [pure virtual]

Implemented in InterfaceSP.

Referenced by handleAdjustmentRequest(), handleCalibrationRequest(), handleInformationRequest(), handleResetRequest(), and handleSetRegisterRequest().

Here is the caller graph for this function:



9.2.3.2 void Interface::debugMessage (const QVariant & *message*) const [signal]

9.2.3.3 void Interface::errorMessage (const QVariant & *message*) const [signal]

Referenced by InterfaceSP::handleTimeout().

Here is the caller graph for this function:

9.2.3.4 quint16 Interface::getBlockSize (void) const

Interface::getBlockSize.

Gets the size of the data block of the communication protocol.

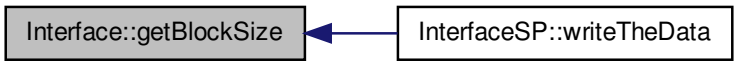
Returns

blockSize of the type quint16.

References blockSize.

Referenced by InterfaceSP::writeTheData().

Here is the caller graph for this function:



9.2.3.5 quint32 Interface::getReceivedBytes (void) const

Interface::getReceivedBytes.

Gets the number of the received bytes.

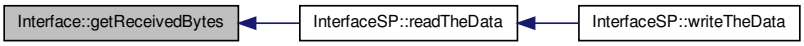
Returns

receivedBytes of the type quint32.

References receivedData.

Referenced by InterfaceSP::readTheData().

Here is the caller graph for this function:



9.2.3.6 QByteArray & Interface::getReceivedData (void)

Interface::getReceivedData.

Returns the received Data from the communication port.

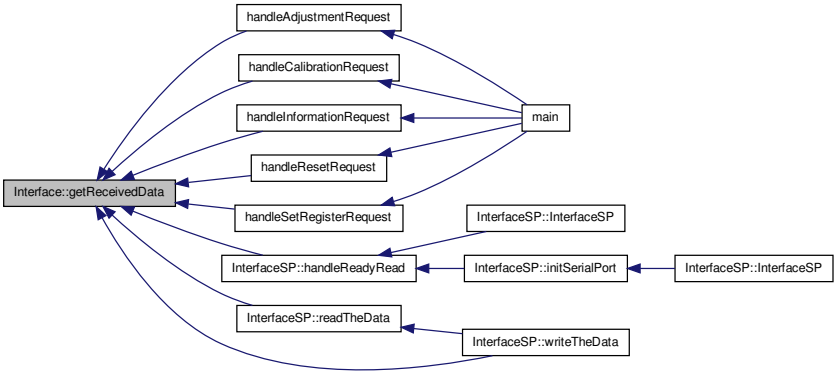
Returns

receivedData of the type QByteArray&.

References receivedData.

Referenced by handleAdjustmentRequest(), handleCalibrationRequest(), handleInformationRequest(), InterfaceSP::handleReadyRead(), handleResetRequest(), handleSetRegisterRequest(), InterfaceSP::readTheData(), and InterfaceSP::writeTheData().

Here is the caller graph for this function:



9.2.3.7 virtual QIODevice* Interface::getSocket (void) [pure virtual]

Implemented in InterfaceSP.

9.2.3.8 quint32 Interface::getTimeout (void) const

Interface::getTimeout.

Returns the Interval for a time-out in milliseconds.

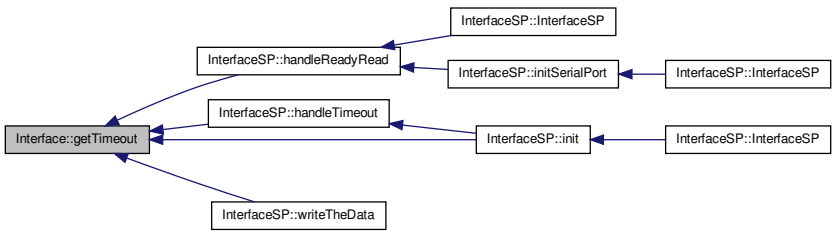
Returns

timeout of the type quint32.

References timeout.

Referenced by InterfaceSP::handleReadyRead(), InterfaceSP::handleTimeout(), InterfaceSP::init(), and InterfaceSP::writeTheData().

Here is the caller graph for this function:



9.2.3.9 QTimer * Interface::getTimer (void)

Interface::getTimer.

Returns the Pointer of object Timer.

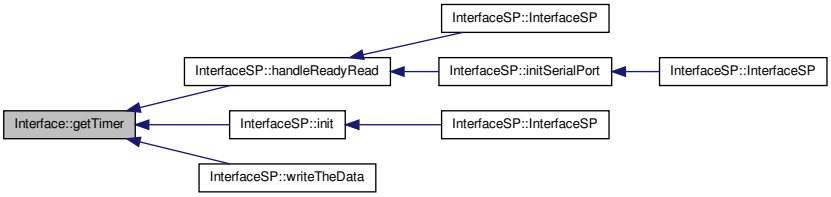
Returns

∗timer of the type QTimer.

References timer.

Referenced by InterfaceSP::handleReadyRead(), InterfaceSP::init(), and InterfaceSP::writeTheData().

Here is the caller graph for this function:



9.2.3.10 void Interface::infoMessage (const QVariant & *message*) const [signal]

9.2.3.11 virtual void Interface::initSocket (void) [pure virtual]

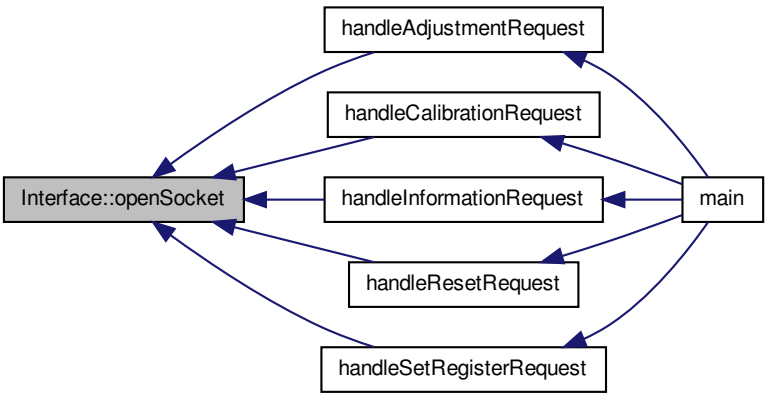
Implemented in InterfaceSP.

9.2.3.12 virtual bool Interface::openSocket (void) [pure virtual]

Implemented in InterfaceSP.

Referenced by handleAdjustmentRequest(), handleCalibrationRequest(), handleInformationRequest(), handleResetRequest(), and handleSetRegisterRequest().

Here is the caller graph for this function:

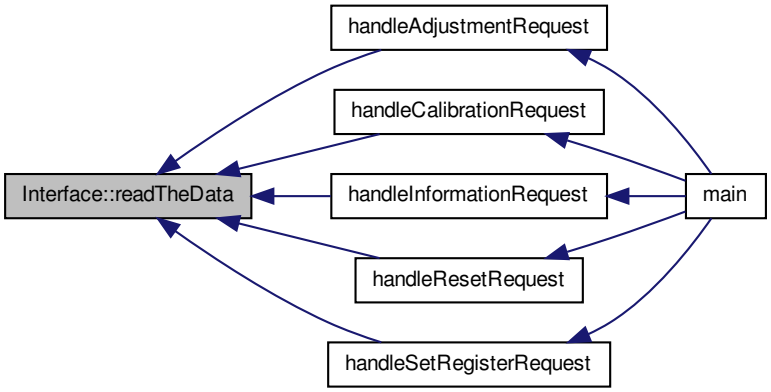


9.2.3.13 virtual bool Interface::readTheData (const quint32 , const quint32 *bytes* = 0U) [pure virtual]

Implemented in InterfaceSP.

Referenced by handleAdjustmentRequest(), handleCalibrationRequest(), handleInformationRequest(), handleResetRequest(), and handleSetRegisterRequest().

Here is the caller graph for this function:



9.2.3.14 void Interface::setBlockSize (const quint16 *size*)

Interface::setBlockSize.

Sets the size of the data block for the communication protocol.

Parameters

size	of typ const quint16

References blockSize.

Referenced by main().

Here is the caller graph for this function:

9.2.3.15 void Interface::setTimeout (quint32 *interval*)

Interface::setTimeout.

Set the Interval for a time-out in milliseconds.

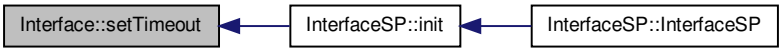
Parameters

interval	of the type quint32.

References timeout.

Referenced by InterfaceSP::init().

Here is the caller graph for this function:



9.2.3.16 void Interface::setTimer (QTimer * *timer*)

Interface::setTimer.

Set the Pointer of new object Timer.

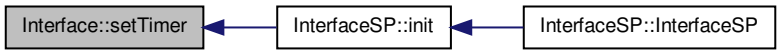
Parameters

timer	of the type QTimer*.

References timer.

Referenced by InterfaceSP::init().

Here is the caller graph for this function:

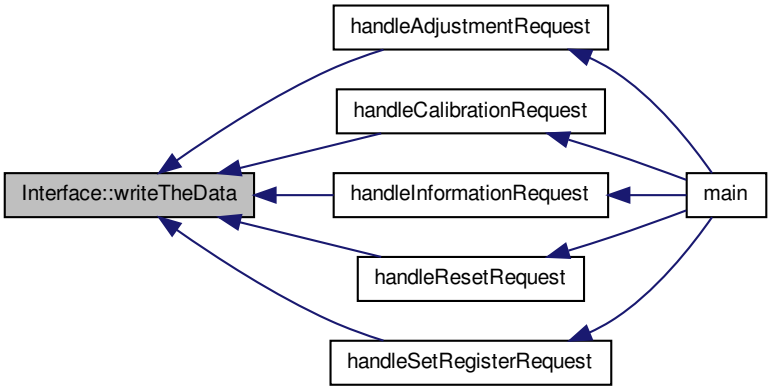


9.2.3.17 virtual qint64 Interface::writeTheData (const QByteArray &) [pure virtual]

Implemented in InterfaceSP.

Referenced by handleAdjustmentRequest(), handleCalibrationRequest(), handleInformationRequest(), handleResetRequest(), and handleSetRegisterRequest().

Here is the caller graph for this function:



9.2.4 Member Data Documentation

9.2.4.1 quint16 Interface::blockSize [private]

Size of the data block for the communication protocol.

Referenced by getBlockSize(), and setBlockSize().

9.2.4.2 QByteArray Interface::receivedData [private]

The received data.

Referenced by getReceivedBytes(), and getReceivedData().

9.2.4.3 quint32 Interface::timeout [private]

The timeout interval in milliseconds.

Referenced by getTimeout(), and setTimeout().

9.2.4.4 QTimer* Interface::timer [private]

Timer for Time-out method.

Referenced by getTimer(), and setTimer().

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

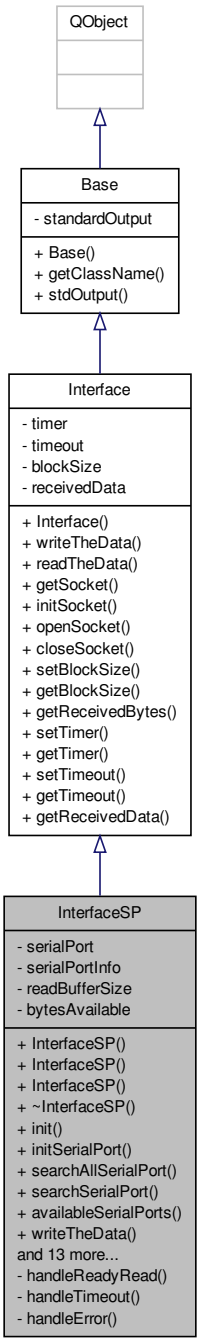
- interface.h
- interface.cpp

9.3 InterfaceSP Class Reference

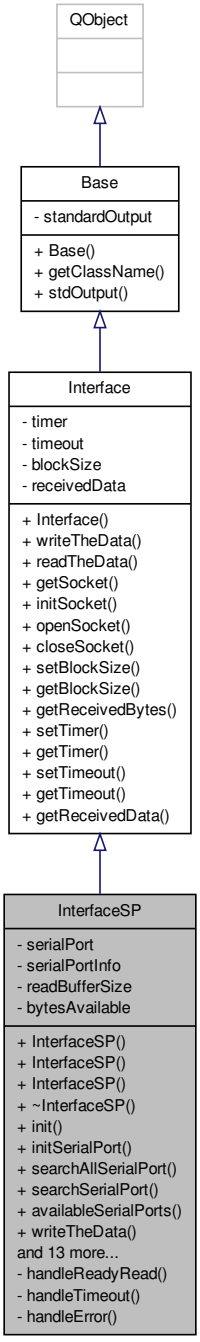
The class InterfaceSP provides methods to access serial ports.

`#include <interface.h>`

Inheritance diagram for InterfaceSP:



Collaboration diagram for InterfaceSP:



Public Member Functions

- InterfaceSP (QObject ∗parent=0)

  InterfaceSP::InterfaceSP Constructor of the class InterfaceSP for creating objects with default values.
- InterfaceSP (QObject ∗parent, const QString &portName)

  InterfaceSP::InterfaceSP Constructor of the class InterfaceSP for creating objects with default values.
- InterfaceSP (QObject ∗parent, QSerialPort ∗port)

InterfaceSP::InterfaceSP Constructor of the class InterfaceSP for creating objects with default values.

- ~InterfaceSP ()

  Destructor for the class InterfaceSP.

- void init (void)

  InterfaceSP::init.

- void initSerialPort (void)

  InterfaceSP::initSerialPort Prepare a instance of the Serial Port.

- void searchAllSerialPort (void)

  InterfaceSP::searchAllSerialPort Info about all available serial ports.

- bool searchSerialPort (const QString &portName)

  InterfaceSP::searchSerialPort This method checks a serial port for existence.

- QStringList availableSerialPorts (void)

  InterfaceSP::availableSerialPorts Return a list with all available serial port names.

- qint64 writeTheData (const QByteArray &data)

  InterfaceSP::writeTheData Writes the content of byteArray to the Serial Port device.

- bool readTheData (const quint32 timewait, const quint32 bytes=0U)

  InterfaceSP::readTheData Reads all remaining data from the UART device.

- QSerialPort * getSocket (void)

  InterfaceSP::getSocket.

- void initSocket (void)

  InterfaceSP::initSocket.

- bool openSocket (void)

  InterfaceSP::openSocket.

- void closeSocket (void)

  InterfaceSP::closeSocket.

- QString getPortName (void) const

  InterfaceSP::getPortName.

- qint32 getPortBaudRate (void) const

  InterfaceSP::getPortBaudRate Returns the baud rate of the serial port.

- QString getDescription (void) const

  InterfaceSP::getDescription.

- QString getManufacturer (void) const

  InterfaceSP::getManufacturer.

- QString getSerialNumber (void) const

  InterfaceSP::getSerialNumber.

- quint16 getProductIdentifier (void) const

  InterfaceSP::getProductIdentifier.

- QSerialPortInfo getSerialPortInfo (void) const

  InterfaceSP::getSerialPortInfo.

- QSerialPort * getSerialPort (void)

  InterfaceSP::getSerialPort.

Private Slots

- void handleReadyRead ()
- void handleTimeout ()

  InterfaceSP::handleTimeout A slot for treatment of the signal Time-Out.

- void handleError (QSerialPort::SerialPortError error)

Private Attributes

- QSerialPort * serialPort

  Serial Port.
- QSerialPortInfo serialPortInfo
- qint64 readBufferSize
- qint64 bytesAvailable

  The number of bytes that are in the buffer for reading.

Additional Inherited Members

9.3.1 Detailed Description

The class InterfaceSP provides methods to access serial ports.

You can get information about the available serial ports using the QSerialPortInfo helper class, which allows an enumeration of all the serial ports in the system. This is useful to obtain the correct name of the serial port you want to use. You can pass an object of the helper class as an argument to the setPort() or setPortName() methods to assign the desired serial device. After setting the port, you can open it in read-only (r/o), write-only (w/o), or read-write (r/w) mode using the open() method.

Note

The serial port is always opened with exclusive access (that is, no other process or thread can access an already opened serial port).

9.3.2 Constructor & Destructor Documentation

9.3.2.1 InterfaceSP::InterfaceSP (QObject * *parent* = 0)

InterfaceSP::InterfaceSP Constructor of the class InterfaceSP for creating objects with default values.

Create a instance of the class and call following init methods:

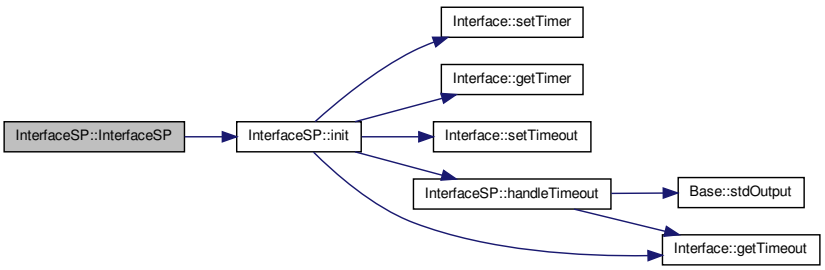
1. InterfaceSP::init()

Parameters

in	*parent*	Pointer to the parent object used for QObject

References init().

Here is the call graph for this function:



9.3.2.2 InterfaceSP::InterfaceSP (QObject * *parent,* const QString & *portName*)

InterfaceSP::InterfaceSP Constructor of the class InterfaceSP for creating objects with default values.

Create a instance of the class and call following init methods:

1. InterfaceSP::init()

2. InterfaceSP::searchSerialPort(const QString & portName)

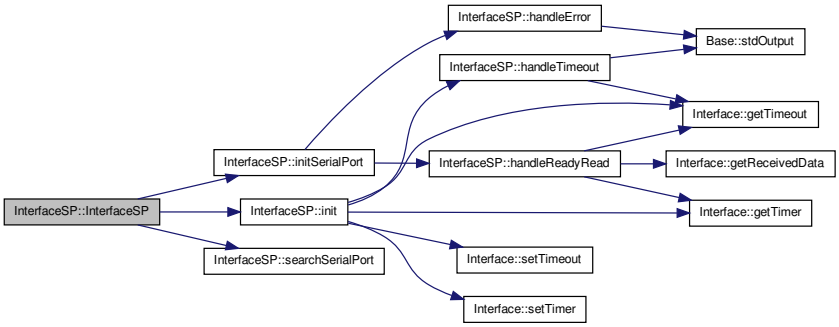
3. InterfaceSP::initSerialPort()

Parameters

in	*parent*	Pointer to the parent object used for QObject.
in	*portName*	a Serial Port Name of the type const QString.

References init(), initSerialPort(), and searchSerialPort().

Here is the call graph for this function:



9.3.2.3 InterfaceSP::InterfaceSP (QObject * *parent,* QSerialPort * *port*)

InterfaceSP::InterfaceSP Constructor of the class InterfaceSP for creating objects with default values.

Create a instance of the class and call following init methods:

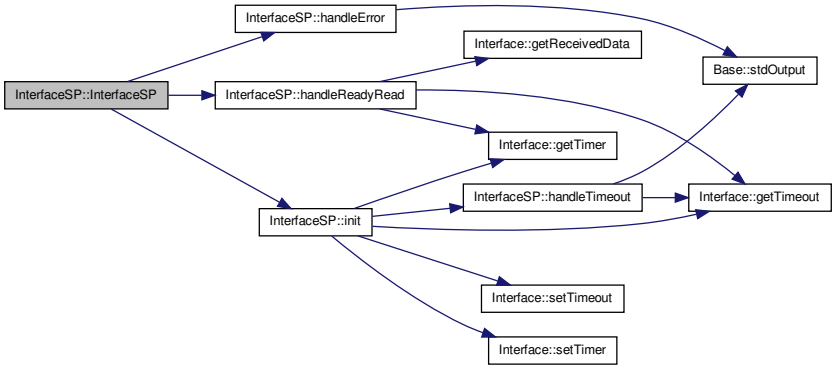
1. InterfaceSP::init()

Parameters

in	*parent*	a pointer an the parent object used for QObject.
in	*port*	a pointer an the Serial Port Name of the type QSerialPort*.

References handleError(), handleReadyRead(), init(), and serialPort.

Here is the call graph for this function:



9.3.2.4 InterfaceSP::~InterfaceSP ( )

Destructor for the class InterfaceSP.

9.3.3 Member Function Documentation

9.3.3.1 QStringList InterfaceSP::availableSerialPorts (void)

InterfaceSP::availableSerialPorts Return a list with all available serial port names.

Returns

serialPorts of the type QStringList

9.3.3.2 void InterfaceSP::closeSocket (void) [virtual]

InterfaceSP::closeSocket.

Implements Interface.

References serialPort.

9.3.3.3 QString InterfaceSP::getDescription (void) const

InterfaceSP::getDescription.

Returns the description string of the serial port.

Returns

description of the type QString.

References serialPortInfo.

9.3.3.4 QString InterfaceSP::getManufacturer (void) const

InterfaceSP::getManufacturer.

Returns the manufacturer string of the serial port.

Returns

manufacturer of the type QString.

References serialPortInfo.

9.3.3.5 qint32 InterfaceSP::getPortBaudRate (void) const

InterfaceSP::getPortBaudRate Returns the baud rate of the serial port.

Warning: Setting the AllDirections flag is supported on all platforms. Windows and Windows CE support only this mode. Warning: Returns equal baud rate in any direction on Windows, Windows CE. The default value is Baud9600, i.e. 9600 bits per second.

Returns

portBaudRate of the type int.

Note

If the setting is set before opening the port, the actual serial port setting is done automatically in the QSerialPort::open() method right after that the opening of the port succeeds.

References serialPort.

9.3.3.6 QString InterfaceSP::getPortName (void) const

InterfaceSP::getPortName.

Returns the name of the serial port.

Returns

portName of the type QString.

References serialPort.

9.3.3.7 quint16 InterfaceSP::getProductIdentifier (void) const

InterfaceSP::getProductIdentifier.

Returns the 16-bit product number for the serial port.

Returns

productIdentifier of the type quint16.

References serialPortInfo.

9.3.3.8 QString InterfaceSP::getSerialNumber (void) const

InterfaceSP::getSerialNumber.

Returns the serial number string of the serial port.

Returns

serialNumber of the type QString.

References serialPortInfo.

9.3.3.9 QSerialPort * InterfaceSP::getSerialPort (void)

InterfaceSP::getSerialPort.

Returns the Information of the serial port.

Returns

serialPort of the type QSerialPort*.

References serialPort.

9.3.3.10 QSerialPortInfo InterfaceSP::getSerialPortInfo (void) const

InterfaceSP::getSerialPortInfo.

Returns the Information of the serial port.

Returns

serialPortInfo of the type QSerialPortInfo.

References serialPortInfo.

9.3.3.11 QSerialPort * InterfaceSP::getSocket (void) [virtual]

InterfaceSP::getSocket.

Returns the Information of the serial port.

Returns

serialPort of the type QSerialPort*.

Implements Interface.

References serialPort.

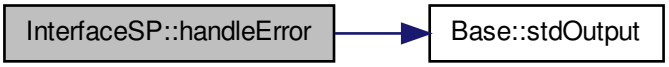
9.3.3.12 void InterfaceSP::handleError (QSerialPort::SerialPortError *error*) [private], [slot]

Todo

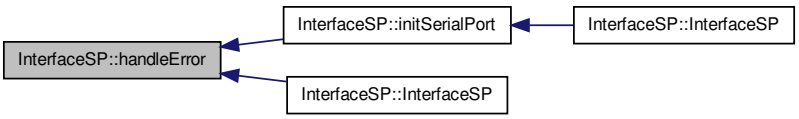
References serialPort, and Base::stdOutput().

Referenced by initSerialPort(), and InterfaceSP().

Here is the call graph for this function:



Here is the caller graph for this function:



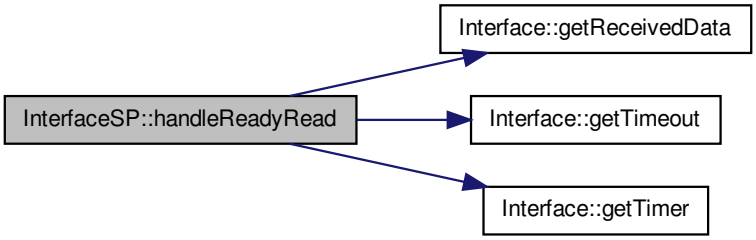
9.3.3.13 void InterfaceSP::handleReadyRead (void) [private], [slot]

Todo

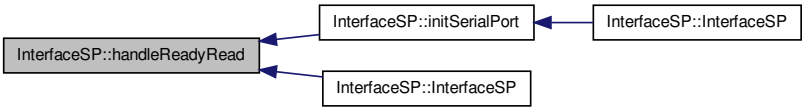
References bytesAvailable, Interface::getReceivedData(), Interface::getTimeout(), Interface::getTimer(), readBufferSize, and serialPort.

Referenced by initSerialPort(), and InterfaceSP().

Here is the call graph for this function:



Here is the caller graph for this function:



9.3.3.14 void InterfaceSP::handleTimeout ( ) [private],[slot]

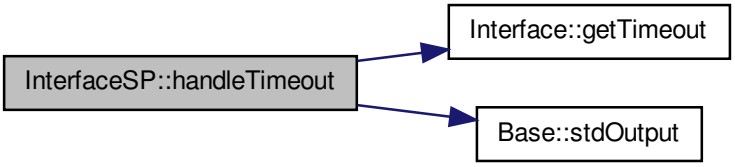
InterfaceSP::handleTimeout A slot for treatment of the signal Time-Out.

The slot terminates the console application or sends another signal to GUI application.

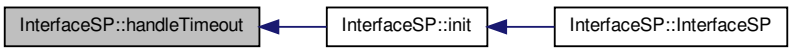
References Interface::errorMessage(), Interface::getTimeout(), serialPort, and Base::stdOutput().

Referenced by init().

Here is the call graph for this function:



Here is the caller graph for this function:



9.3.3.15 void InterfaceSP::init (void)

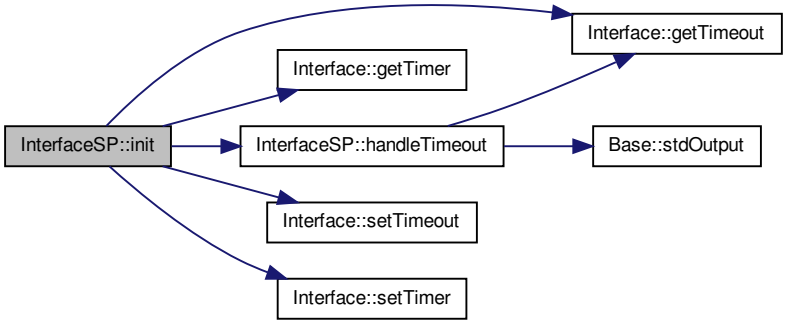
InterfaceSP::init.

This method establishes the necessary fields of the class to read the data, and starts a timeout.

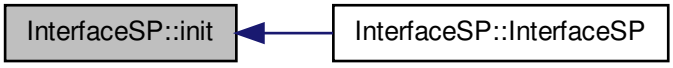
References bytesAvailable, Interface::getTimeout(), Interface::getTimer(), handleTimeout(), readBufferSize, Interface::setTimeout(), and Interface::setTimer().

Referenced by InterfaceSP().

Here is the call graph for this function:



Here is the caller graph for this function:



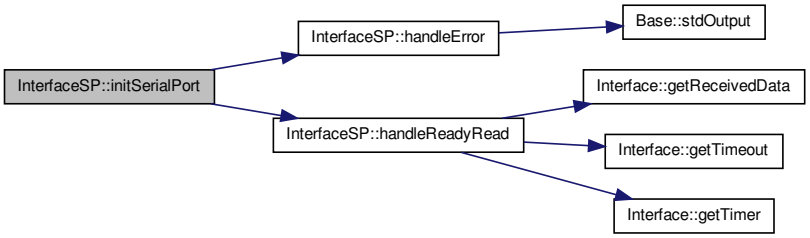
9.3.3.16 void InterfaceSP::initSerialPort (void)

InterfaceSP::initSerialPort Prepare a instance of the Serial Port.

References handleError(), handleReadyRead(), readBufferSize, serialPort, and serialPortInfo.

Referenced by InterfaceSP().

Here is the call graph for this function:



Here is the caller graph for this function:

9.3.3.17 void InterfaceSP::initSocket (void) [virtual]

InterfaceSP::initSocket.

Implements Interface.

9.3.3.18 bool InterfaceSP::openSocket (void) [virtual]

InterfaceSP::openSocket.

Implements Interface.

References REBOOT_WAIT, and serialPort.

9.3.3.19 bool InterfaceSP::readTheData (const quint32 *timewait*, const quint32 *bytes* = 0U) [virtual]

InterfaceSP::readTheData Reads all remaining data from the UART device.

Set a time-out for the read the data from a device.

Parameters

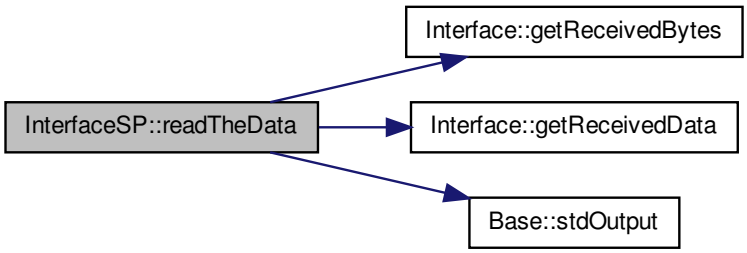
in	*timewait*	of the type quint32.
in	*bytes*	the number of bytes to read from the UART.

Implements Interface.

References Interface::getReceivedBytes(), Interface::getReceivedData(), serialPort, and Base::stdOutput().

Referenced by writeTheData().

Here is the call graph for this function:



Here is the caller graph for this function:

9.3.3.20 void InterfaceSP::searchAllSerialPort (void)

InterfaceSP::searchAllSerialPort Info about all available serial ports.

References Base::stdOutput().

Referenced by main().

Here is the call graph for this function:

Here is the caller graph for this function:

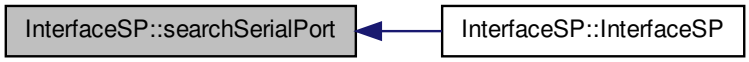
9.3.3.21 bool InterfaceSP::searchSerialPort (const QString & *portName*)

InterfaceSP::searchSerialPort This method checks a serial port for existence.

References serialPortInfo.

Referenced by InterfaceSP().

Here is the caller graph for this function:



9.3.3.22 qint64 InterfaceSP::writeTheData (const QByteArray & *data*) [virtual]

InterfaceSP::writeTheData Writes the content of byteArray to the Serial Port device.

Send the data to a UART device.

Parameters

data	of type const QByteArray.

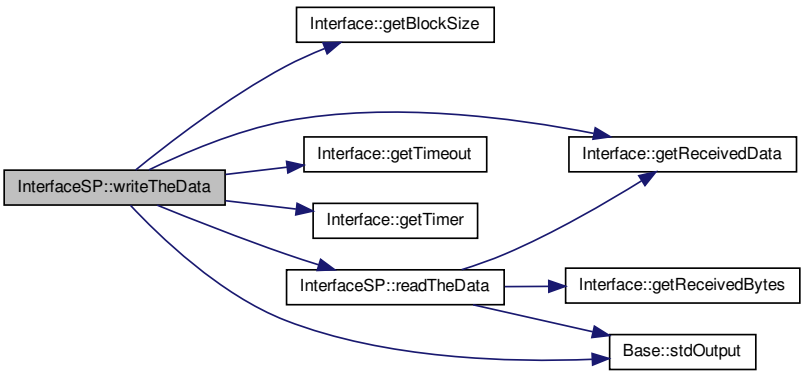
Returns

bytesWritten of the type qint64.

Implements Interface.

References Interface::getBlockSize(), Interface::getReceivedData(), Interface::getTimeout(), Interface::getTimer(), readTheData(), serialPort, and Base::stdOutput().

Here is the call graph for this function:



9.3.4 Member Data Documentation

9.3.4.1 qint64 InterfaceSP::bytesAvailable [private]

The number of bytes that are in the buffer for reading.

Referenced by handleReadyRead(), and init().

9.3.4.2 qint64 InterfaceSP::readBufferSize [private]

The size of the internal read buffer. This limits the amount of data that the client can receive before calling the read() or readAll() methods.

Referenced by handleReadyRead(), init(), and initSerialPort().

9.3.4.3 QSerialPort* InterfaceSP::serialPort [private]

Serial Port.

Referenced by closeSocket(), getPortBaudRate(), getPortName(), getSerialPort(), getSocket(), handleError(), handleReadyRead(), handleTimeout(), initSerialPort(), InterfaceSP(), openSocket(), readTheData(), and writeTheData().

9.3.4.4 QSerialPortInfo InterfaceSP::serialPortInfo [private]

It provides information on this serial ports:

- The name of serial port
- The description string of the serial port
- The manufacturer string of the serial port
- The serial number string of the serial port
- The 16-bit product number for the serial port
- The 16-bit vendor number for the serial port
- The list of available standard baud rates supported by the current serial port

Referenced by getDescription(), getManufacturer(), getProductIdentifier(), getSerialNumber(), getSerialPortInfo(), initSerialPort(), and searchSerialPort().

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

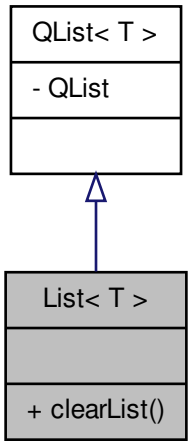
- interface.h
- interface.cpp

9.4 List< T > Class Template Reference

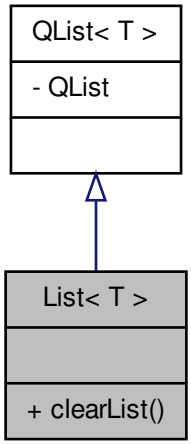
Class List overrides QList class methods.

`#include <base.h>`

Inheritance diagram for List< T >:



Collaboration diagram for List< T >:



Public Member Functions

- void clearList (void)

9.4.1 Detailed Description

template<typename T>class List< T >

Class List overrides QList class methods.

This class is designed to expand the functionality of class QList.

- added method clearList() to free the memory occupied by the objects and elements of the list (via at the QVector).

9.4.2 Member Function Documentation

9.4.2.1 template<typename T > void List< T >::clearList (void)

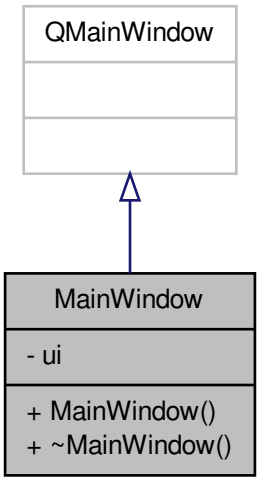
The documentation for this class was generated from the following file:

- base.h

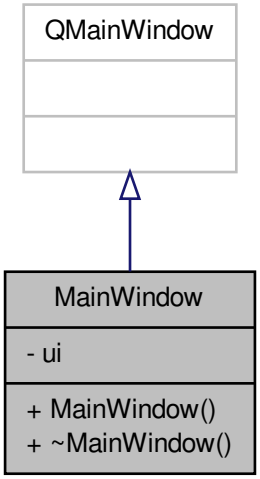
9.5 MainWindow Class Reference

`#include <mainwindow.h>`

Inheritance diagram for MainWindow:



Collaboration diagram for MainWindow:



Public Member Functions

- MainWindow (QWidget ∗parent=0)
- ∼MainWindow ()

Private Attributes

- Ui::MainWindow ∗ ui

9.5.1 Constructor & Destructor Documentation

9.5.1.1 MainWindow::MainWindow (QWidget ∗ *parent* = 0) [explicit]

References ui.

9.5.1.2 MainWindow::∼MainWindow ( )

References ui.

9.5.2 Member Data Documentation

9.5.2.1 Ui::MainWindow∗ MainWindow::ui [private]

Referenced by MainWindow(), and ∼MainWindow().

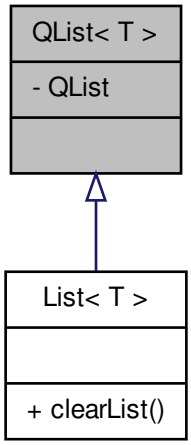
The documentation for this class was generated from the following files:

- mainwindow.h
- mainwindow.cpp

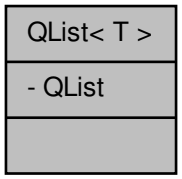
9.6 QList< T > Class Template Reference

`#include <base.h>`

Inheritance diagram for QList< T >:



Collaboration diagram for QList< T >:



Private Attributes

- T QList

9.6.1 Detailed Description

template<class T>class QList< T >

Only to create doxygen collaboration diagram (dummy definition of QList as doxygen naturally does not know these types)

9.6.2 Member Data Documentation

9.6.2.1 template<class T > T QList< T >::QList [private]

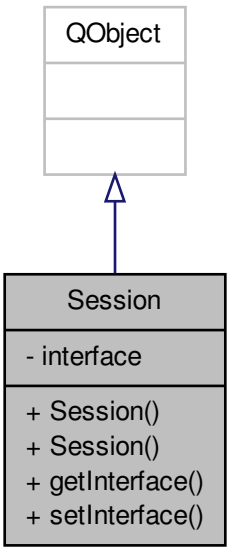
The documentation for this class was generated from the following file:

- base.h

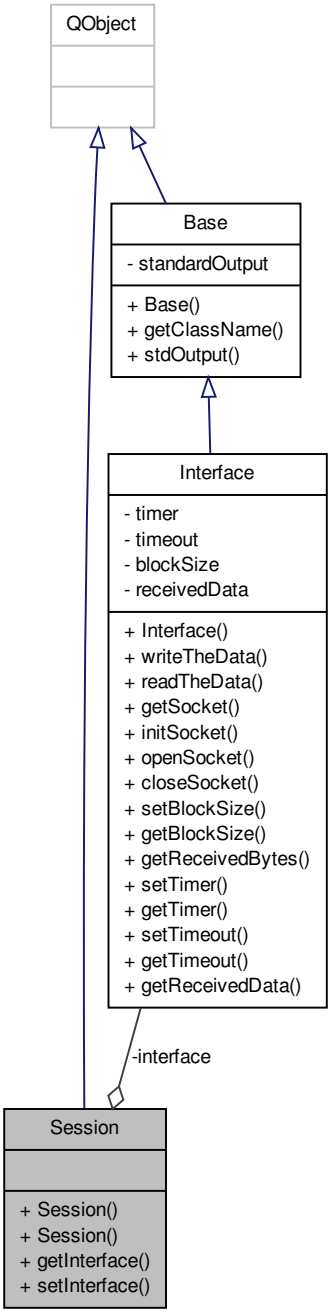
9.7 Session Class Reference

`#include <session.h>`

Inheritance diagram for Session:



Collaboration diagram for Session:



Public Member Functions

- Session (QObject ∗parent=0)
- Session (QObject ∗parent, Interface ∗const interface)
- Interface ∗ getInterface (void)
- void setInterface (Interface ∗const interface)

Private Attributes

- Interface * interface

9.7.1 Constructor & Destructor Documentation

9.7.1.1 Session::Session (QObject * *parent* = 0)

9.7.1.2 Session::Session (QObject * *parent*, Interface *const *interface*)

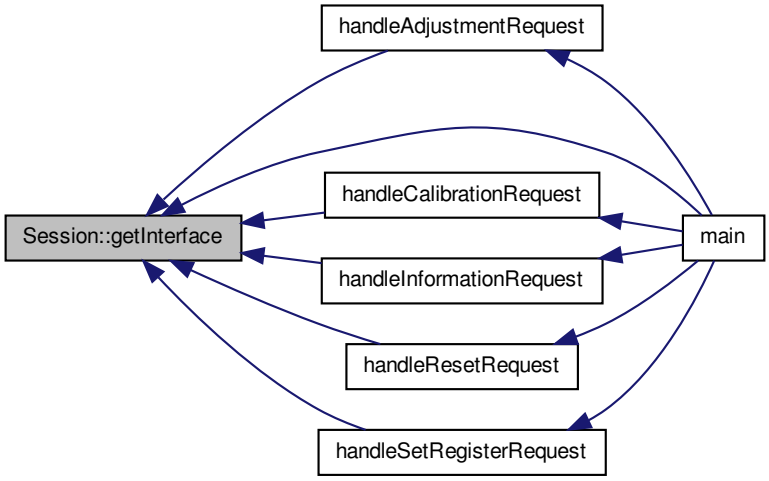
9.7.2 Member Function Documentation

9.7.2.1 Interface * Session::getInterface (void)

References interface.

Referenced by handleAdjustmentRequest(), handleCalibrationRequest(), handleInformationRequest(), handleResetRequest(), handleSetRegisterRequest(), and main().

Here is the caller graph for this function:



9.7.2.2 void Session::setInterface (Interface *const *interface*)

9.7.3 Member Data Documentation

9.7.3.1 Interface* Session::interface [private]

Referenced by getInterface().

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

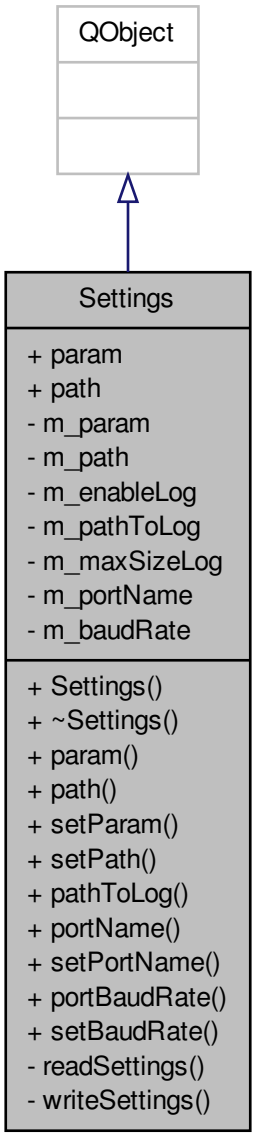
- session.h
- session.cpp

9.8 Settings Class Reference

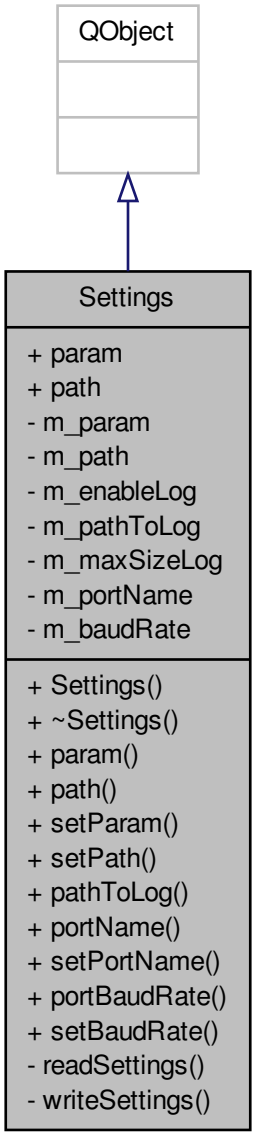
The Settings class.

`#include <settings.h>`

Inheritance diagram for Settings:



Collaboration diagram for Settings:



Signals

- void paramChanged (void)
- void pathChanged (void)

Public Member Functions

- Settings (QObject ∗parent=0)

  Settings::Settings.
- ∼Settings ()

Settings::~Settings.

- QString param (void) const

  Settings::param.
- QString path (void) const
- void setParam (const QString &param)

  Settings::setParam.
- void setPath (const QString &path)

  Settings::setPath.
- QString pathToLog (void) const

  Settings::pathToLog.
- QString portName (void) const

  Settings::portName.
- void setPortName (const QString &port)

  Settings::setPortName.
- qint32 portBaudRate (void) const

  Settings::portBaudRate.
- void setBaudRate (const qint32 baudRate)

  Settings::setBaudRate.

Properties

- QString param
- QString path

  Settings::path.

Private Member Functions

- void readSettings (void)

  The function reads the parameters necessary for the user interface that were saved in the previous session.
- void writeSettings (void) const

  The function saves the user interface parameters that have been changed by the user in the current session.

Private Attributes

- QString m_param
- QString m_path
- bool m_enableLog
- QString m_pathToLog
- quint32 m_maxSizeLog
- QString m_portName
- qint32 m_baudRate

9.8.1 Detailed Description

The Settings class.

9.8.2 Constructor & Destructor Documentation

9.8.2.1 Settings::Settings (QObject * *parent* = 0) [explicit]

Settings::Settings.

Parameters

parent	

References readSettings().

Here is the call graph for this function:

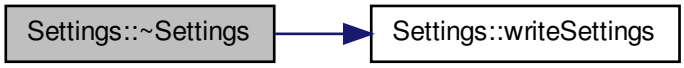
9.8.2.2 Settings::~Settings ( )

Settings::~Settings.

Todo

References writeSettings().

Here is the call graph for this function:



9.8.3 Member Function Documentation

9.8.3.1 QString Settings::param (void) const

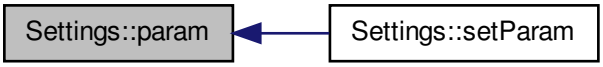
Settings::param.

Returns

References m_param.

Referenced by setParam().

Here is the caller graph for this function:

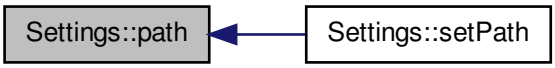


9.8.3.2 void Settings::paramChanged (void) [signal]

9.8.3.3 QString Settings::path (void) const

Referenced by setPath().

Here is the caller graph for this function:



9.8.3.4 void Settings::pathChanged (void) [signal]

9.8.3.5 QString Settings::pathToLog (void) const

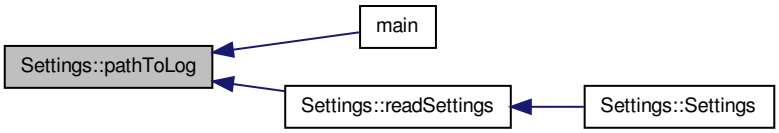
Settings::pathToLog.

Returns

References m_pathToLog.

Referenced by main(), and readSettings().

Here is the caller graph for this function:



9.8.3.6 qint32 Settings::portBaudRate (void) const

Settings::portBaudRate.

Returns

References m_baudRate.

9.8.3.7 QString Settings::portName (void) const

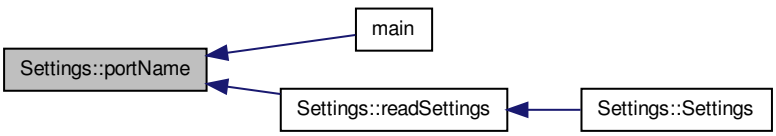
Settings::portName.

Returns

References m_portName.

Referenced by main(), and readSettings().

Here is the caller graph for this function:



9.8.3.8 void Settings::readSettings (void) [private]

The function reads the parameters necessary for the user interface that were saved in the previous session.

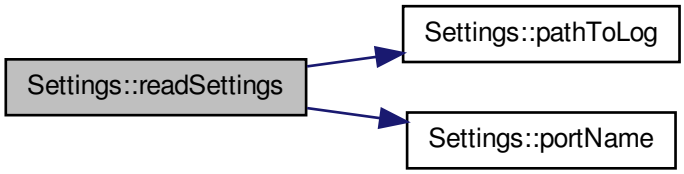
Such important parameters will be read as

- the position of the window on the screen and window size,
- interface font and its size,
- the user interface settings (overwrite of the data, recurse of dir's, etc.)
- error and event logging options.

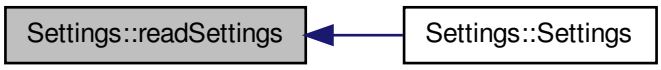
References m_baudRate, m_enableLog, m_maxSizeLog, m_pathToLog, m_portName, pathToLog(), and portName().

Referenced by Settings().

Here is the call graph for this function:



Here is the caller graph for this function:



9.8.3.9 void Settings::setBaudRate (const qint32 *baudRate*)

Settings::setBaudRate.

Parameters

baudRate	

References m_baudRate.

9.8.3.10 void Settings::setParam (const QString & *param*)

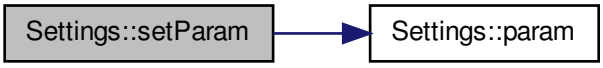
Settings::setParam.

Parameters

param	

References m_param, and param().

Here is the call graph for this function:



9.8.3.11 void Settings::setPath (const QString & *path*)

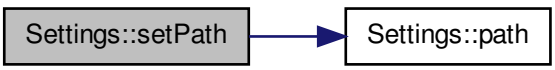
Settings::setPath.

Parameters

path	

References m_path, and path().

Here is the call graph for this function:



9.8.3.12 void Settings::setPortName (const QString & *port*)

Settings::setPortName.

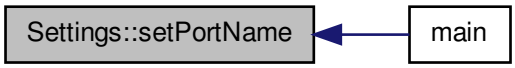
Parameters

port	

References m_portName.

Referenced by main().

Here is the caller graph for this function:

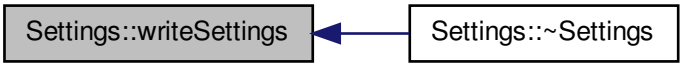


9.8.3.13 void Settings::writeSettings (void) const [private]

The function saves the user interface parameters that have been changed by the user in the current session.
Such parameters will be updated as

- the position of the window on the screen and window size,
- interface font and its size,
- the user interface settings (overwrite of the data, recurse of dir's, etc.)
- error and event logging options.

References m_baudRate, m_enableLog, m_maxSizeLog, m_pathToLog, and m_portName.
Referenced by ~Settings().
Here is the caller graph for this function:



9.8.4 Member Data Documentation

9.8.4.1 qint32 Settings::m_baudRate [private]

Referenced by portBaudRate(), readSettings(), setBaudRate(), and writeSettings().

9.8.4.2 bool Settings::m_enableLog [private]

Referenced by readSettings(), and writeSettings().

9.8.4.3 quint32 Settings::m_maxSizeLog [private]

Referenced by readSettings(), and writeSettings().

9.8.4.4 QString Settings::m_param [private]

Referenced by param(), and setParam().

9.8.4.5 QString Settings::m_path [private]

Referenced by setPath().

9.8.4.6 QString Settings::m_pathToLog [private]

Referenced by pathToLog(), readSettings(), and writeSettings().

9.8.4.7 QString Settings::m_portName [private]

Referenced by portName(), readSettings(), setPortName(), and writeSettings().

9.8.5 Property Documentation

9.8.5.1 QString Settings::param [read], [write]

9.8.5.2 QString Settings::path [read], [write]

Settings::path.

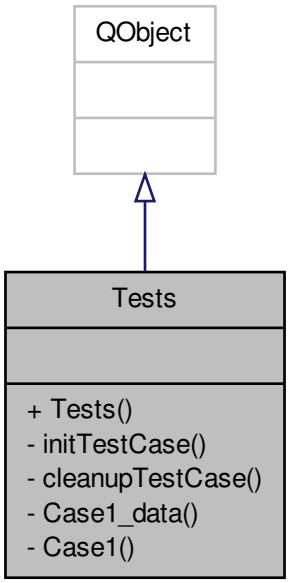
Returns

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

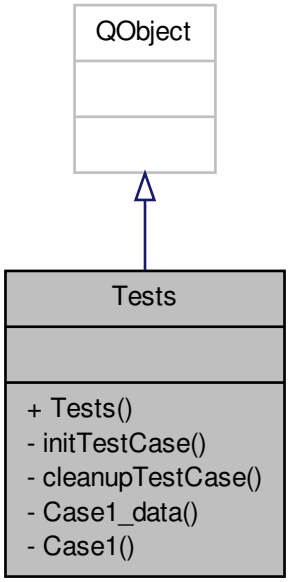
- settings.h
- settings.cpp

9.9 Tests Class Reference

Inheritance diagram for Tests:



Collaboration diagram for Tests:



Public Member Functions

- Tests ()

Private Slots

- void initTestCase ()
- void cleanupTestCase ()
- void Case1_data ()
- void Case1 ()

9.9.1 Constructor & Destructor Documentation

9.9.1.1 Tests::Tests ( )

9.9.2 Member Function Documentation

9.9.2.1 void Tests::Case1 ( ) [private], [slot]

9.9.2.2 void Tests::Case1_data ( ) [private], [slot]

9.9.2.3 void Tests::cleanupTestCase ( ) [private], [slot]

9.9.2.4 void Tests::initTestCase ( ) [private], [slot]

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

- tests.cpp

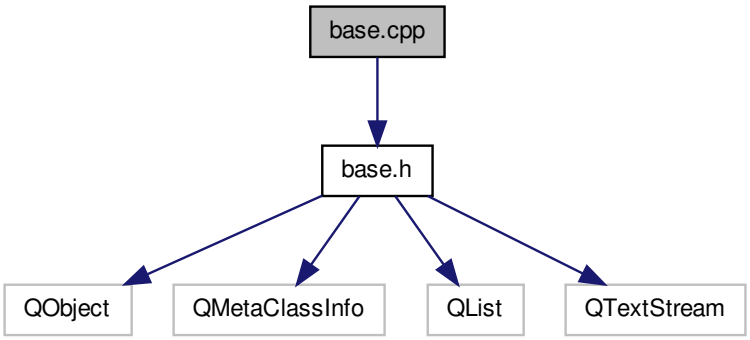
Chapter 10

File Documentation

10.1 base.cpp File Reference

```
#include "base.h"
```

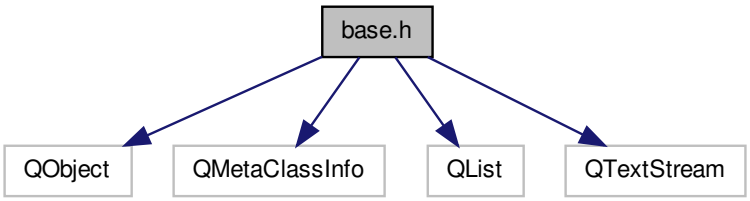
Include dependency graph for base.cpp:



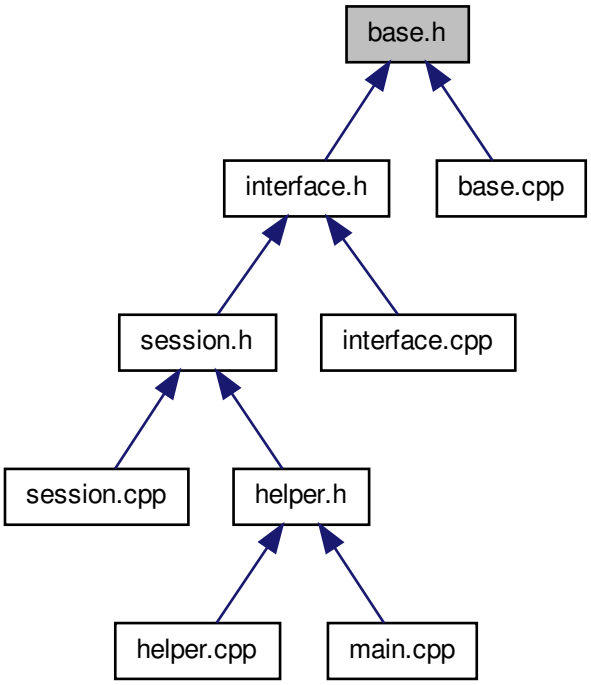
10.2 base.h File Reference

```
#include <QObject>
#include <QMetaClassInfo>
#include <QList>
#include <QTextStream>
```

Include dependency graph for base.h:



This graph shows which files directly or indirectly include this file:



Classes

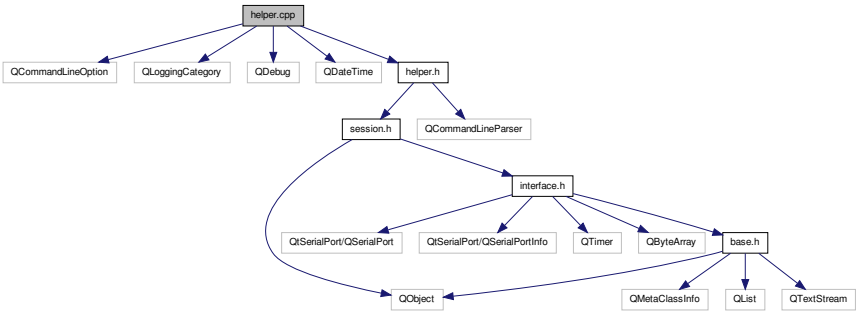
- class QList< T >
- class Base
- class List< T >

  Class List overrides QList class methods.

10.3 helper.cpp File Reference

```
#include <QCommandLineOption>
#include <QLoggingCategory>
#include <QDebug>
#include <QDateTime>
#include "helper.h"
```

Include dependency graph for helper.cpp:



Functions

- static QT_USE_NAMESPACE QTextStream standardOutput (stdout)
- void setCommandLineParser (QCommandLineParser &parser)

  Set Command Line Parser's parameters and optins.
- int handleResetRequest (Session *const session)

  Request for the Reset of the device.
- int handleInformationRequest (Session *const session)

  Request for the Information of the device.
- int handleAdjustmentRequest (Session *const session)

  Request for the Adjustment of the device.
- int handleCalibrationRequest (Session *const session)

  Request for the Calibration of the device.
- int handleSetRegisterRequest (Session *const session, const float value)

  Request to update the register value.

10.3.1 Function Documentation

10.3.1.1 int handleAdjustmentRequest (Session *const *session*)

Request for the Adjustment of the device.

The function creates a connection to the device and sends a request for Adjustment using the Adjustment protocol of the form: {@a|time|ms}, 2+4+2 bytes long.

Parameters

in	*session*	Pointer to the current session.

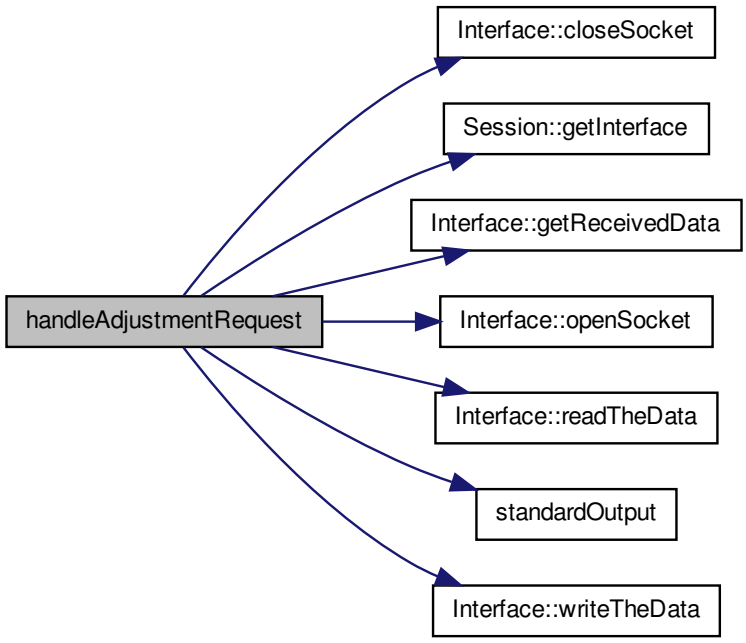
Return values

0	if no error occurs,
1	terminate with an error.

References Interface::closeSocket(), Session::getInterface(), Interface::getReceivedData(), Interface::openSocket(), Interface::readTheData(), RECEIVED_BYTES, standardOutput(), TIME_WAIT, and Interface::writeTheData().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

10.3.1.2 int handleCalibrationRequest (Session *const *session*)

Request for the Calibration of the device.

The function creates a connection to the device and sends a request for Calibration using the Calibration protocol

of the form: {@c|time|ms}, 2+4+2 bytes long.

Parameters

in	*session*	Pointer to the current session.

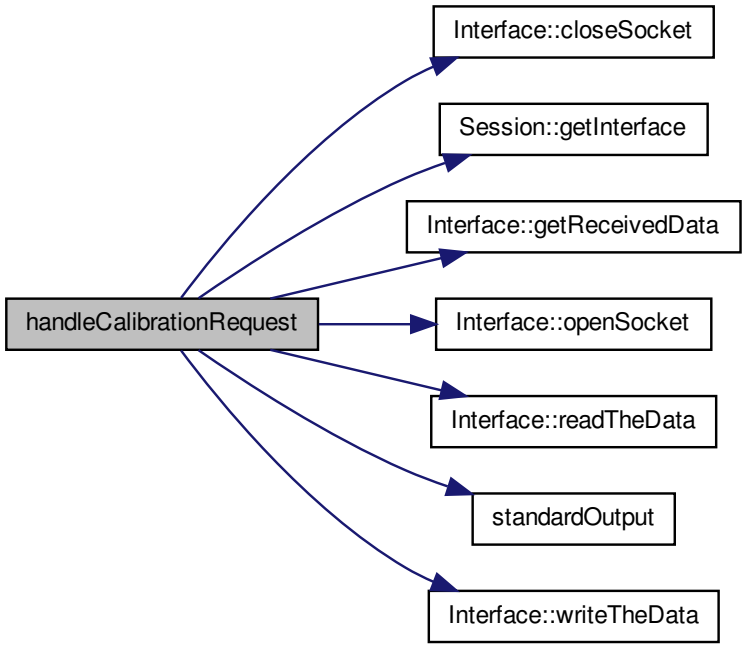
Return values

0	if no error occurs,
1	terminate with an error.

References Interface::closeSocket(), Session::getInterface(), Interface::getReceivedData(), Interface::openSocket(), Interface::readTheData(), RECEIVED_BYTES_CALIBR, standardOutput(), TIME_WAIT, and Interface::writeTheData().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

10.3.1.3 int handleInformationRequest (Session ∗const *session*)

Request for the Information of the device.

The function creates a connection to the device and sends a request for Information using the Information protocol of the form: {@i|time|ms}, 2+4+2 bytes long.

Parameters

in	*session*	Pointer to the current session.

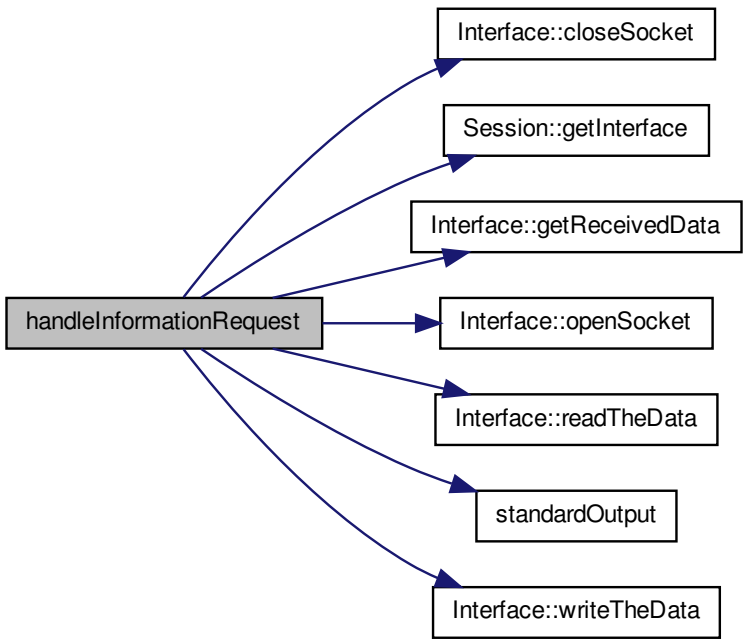
Return values

0	if no error occurs,
1	terminate with an error.

References Interface::closeSocket(), Session::getInterface(), Interface::getReceivedData(), Interface::openSocket(), Interface::readTheData(), RECEIVED_BYTES_INFO, standardOutput(), TIME_WAIT, and Interface::writeTheData().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

10.3.1.4 int handleResetRequest (Session *const *session*)

Request for the Reset of the device.

The function creates a connection to the device and sends a request for Reset using the Reset protocol of the form: {@r}, 2 bytes long.

Parameters

in	*session*	Pointer to the current session.

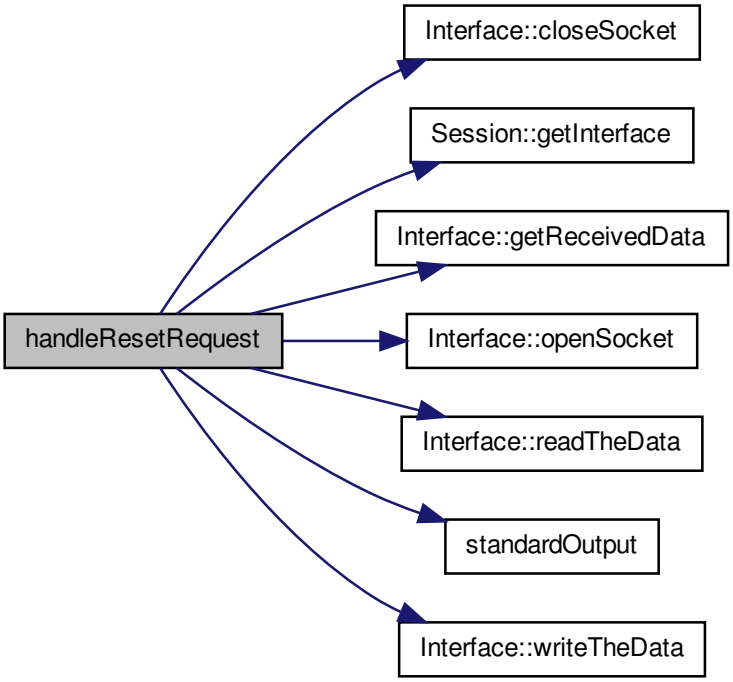
Return values

0	
1	

References Interface::closeSocket(), Session::getInterface(), Interface::getReceivedData(), Interface::openSocket(), Interface::readTheData(), RECEIVED_BYTES, standardOutput(), TIME_WAIT, and Interface::writeTheData().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

10.3.1.5 int handleSetRegisterRequest (Session *const *session, const float *value*)

Request to update the register value.

The function creates a connection to the device and sends a request for SetRegister using the SetRegister protocol of the form: {@s|value}, 2+4 bytes long.

Parameters

in	*session*	Pointer to the current session.
in	*value*	

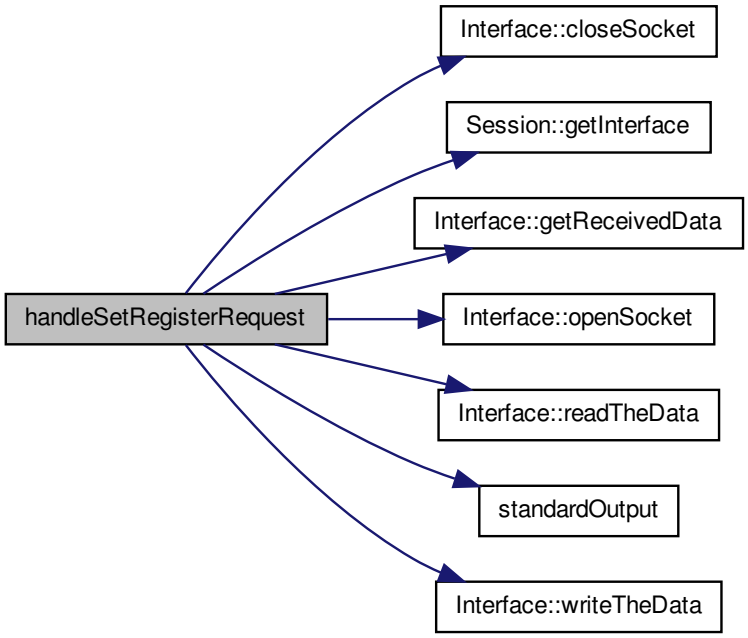
Return values

0	if no error occurs,
1	terminate with an error.

References Interface::closeSocket(), Session::getInterface(), Interface::getReceivedData(), Interface::openSocket(), Interface::readTheData(), RECEIVED_BYTES, standardOutput(), TIME_WAIT, and Interface::writeTheData().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

10.3.1.6 void setCommandLineParser (QCommandLineParser & *parser*)

Set Command Line Parser's parameters and optins.

Todo

Parameters

in,out	*parser*	of the type QCommandLineParser

References ADJUST, CALIBR, DISCOVERY, INFO, PORT, RESET, and SETREG.

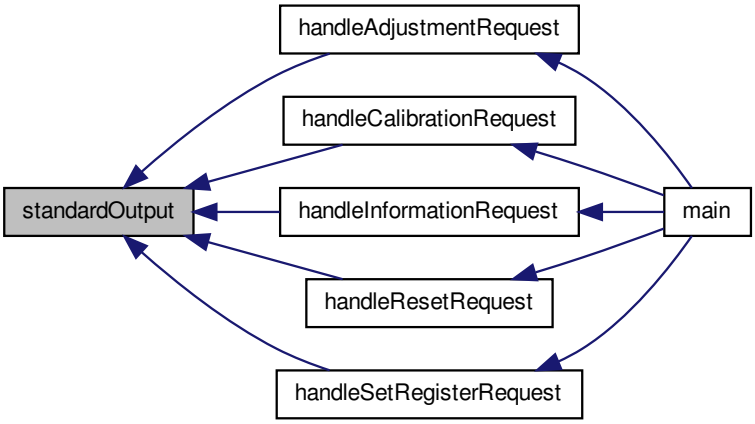
Referenced by main().

Here is the caller graph for this function:

10.3.1.7 static QT_USE_NAMESPACE QTextStream standardOutput (stdout) [static]

Referenced by handleAdjustmentRequest(), handleCalibrationRequest(), handleInformationRequest(), handleResetRequest(), and handleSetRegisterRequest().

Here is the caller graph for this function:

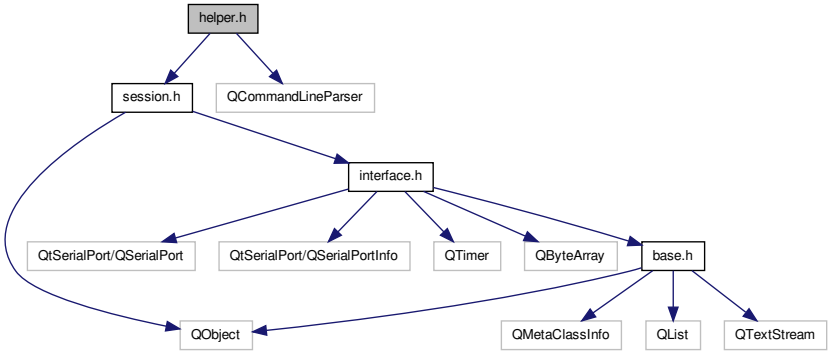


10.4 helper.h File Reference

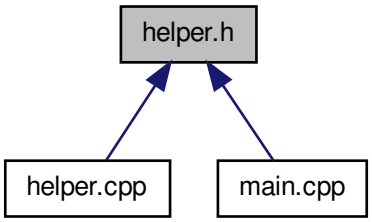
```
#include "session.h"
```

```
#include <QCommandLineParser>
```

Include dependency graph for helper.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define TIME_WAIT 10U
- #define DISCOVERY "d"
- #define PORT "p"
- #define INFO "i"
- #define ADJUST "a"
- #define CALIBR "c"
- #define RESET "r"
- #define SETREG "s"
- #define RECEIVED_BYTES 1U
- #define RECEIVED_BYTES_INFO 15U
- #define RECEIVED_BYTES_CALIBR 7U

Functions

- void setCommandLineParser (QCommandLineParser &parser)

  Set Command Line Parser's parameters and optins.

- int handleInformationRequest (Session *const session)

  Request for the Information of the device.
- int handleAdjustmentRequest (Session *const session)

  Request for the Adjustment of the device.
- int handleCalibrationRequest (Session *const session)

  Request for the Calibration of the device.
- int handleResetRequest (Session *const session)

  Request for the Reset of the device.
- int handleSetRegisterRequest (Session *const session, const float value)

  Request to update the register value.

10.4.1 Macro Definition Documentation

10.4.1.1 #define ADJUST "a"

Referenced by main(), and setCommandLineParser().

10.4.1.2 #define CALIBR "c"

Referenced by main(), and setCommandLineParser().

10.4.1.3 #define DISCOVERY "d"

Referenced by main(), and setCommandLineParser().

10.4.1.4 #define INFO "i"

Referenced by main(), and setCommandLineParser().

10.4.1.5 #define PORT "p"

Referenced by main(), and setCommandLineParser().

10.4.1.6 #define RECEIVED_BYTES 1U

Referenced by handleAdjustmentRequest(), handleResetRequest(), and handleSetRegisterRequest().

10.4.1.7 #define RECEIVED_BYTES_CALIBR 7U

Referenced by handleCalibrationRequest().

10.4.1.8 #define RECEIVED_BYTES_INFO 15U

Referenced by handleInformationRequest().

10.4.1.9 #define RESET "r"

Referenced by main(), and setCommandLineParser().

10.4.1.10 #define SETREG "s"

Referenced by main(), and setCommandLineParser().

10.4.1.11 #define TIME_WAIT 10U

Referenced by handleAdjustmentRequest(), handleCalibrationRequest(), handleInformationRequest(), handleResetRequest(), and handleSetRegisterRequest().

10.4.2 Function Documentation

10.4.2.1 int handleAdjustmentRequest (Session ∗const *session*)

Request for the Adjustment of the device.

The function creates a connection to the device and sends a request for Adjustment using the Adjustment protocol of the form: {@a|time|ms}, 2+4+2 bytes long.

Parameters

in	*session*	Pointer to the current session.

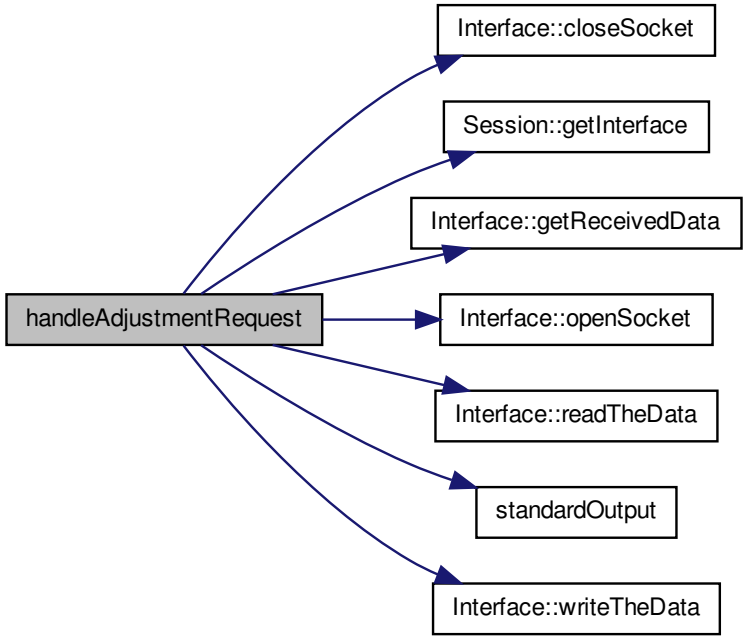
Return values

0	if no error occurs,
1	terminate with an error.

References Interface::closeSocket(), Session::getInterface(), Interface::getReceivedData(), Interface::openSocket(), Interface::readTheData(), RECEIVED_BYTES, standardOutput(), TIME_WAIT, and Interface::writeTheData().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

10.4.2.2 int handleCalibrationRequest (Session *const *session*)

Request for the Calibration of the device.

The function creates a connection to the device and sends a request for Calibration using the Calibration protocol of the form: {@c|time|ms}, 2+4+2 bytes long.

Parameters

in	*session*	Pointer to the current session.

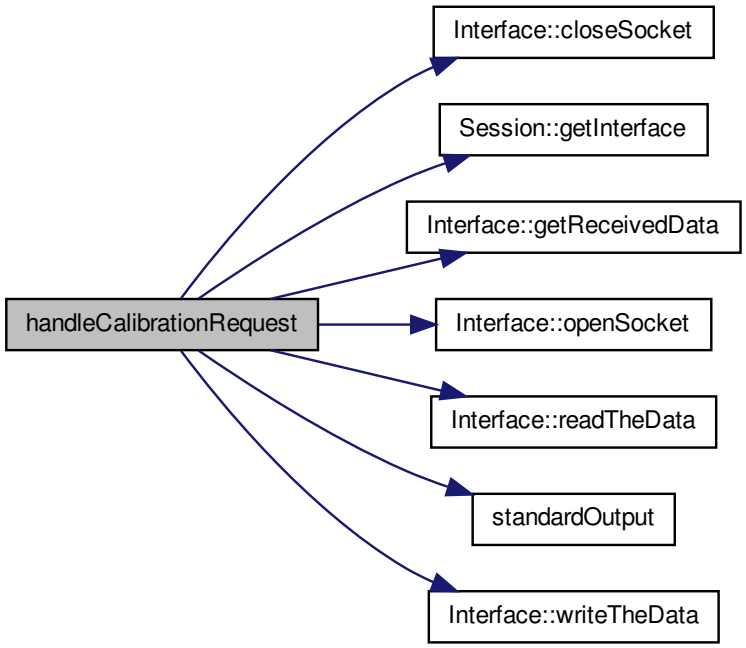
Return values

0	if no error occurs,
1	terminate with an error.

References Interface::closeSocket(), Session::getInterface(), Interface::getReceivedData(), Interface::openSocket(), Interface::readTheData(), RECEIVED_BYTES_CALIBR, standardOutput(), TIME_WAIT, and Interface::writeTheData().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

10.4.2.3 int handleInformationRequest (Session *const *session*)

Request for the Information of the device.

The function creates a connection to the device and sends a request for Information using the Information protocol of the form: {@i|time|ms}, 2+4+2 bytes long.

Parameters

in	*session*	Pointer to the current session.

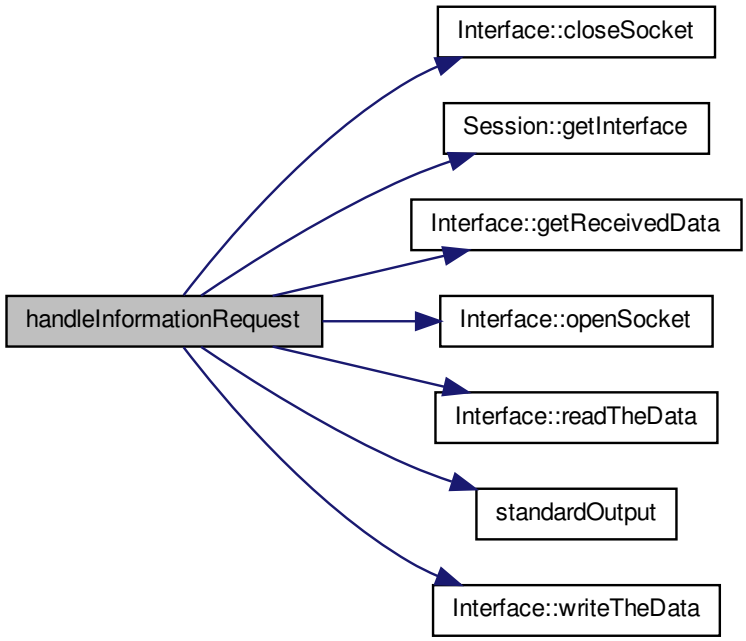
Return values

0	if no error occurs,
1	terminate with an error.

References Interface::closeSocket(), Session::getInterface(), Interface::getReceivedData(), Interface::openSocket(), Interface::readTheData(), RECEIVED_BYTES_INFO, standardOutput(), TIME_WAIT, and Interface::writeTheData().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

10.4.2.4 int handleResetRequest (Session ∗const *session*)

Request for the Reset of the device.

The function creates a connection to the device and sends a request for Reset using the Reset protocol of the form: {@r}, 2 bytes long.

Parameters

in	*session*	Pointer to the current session.

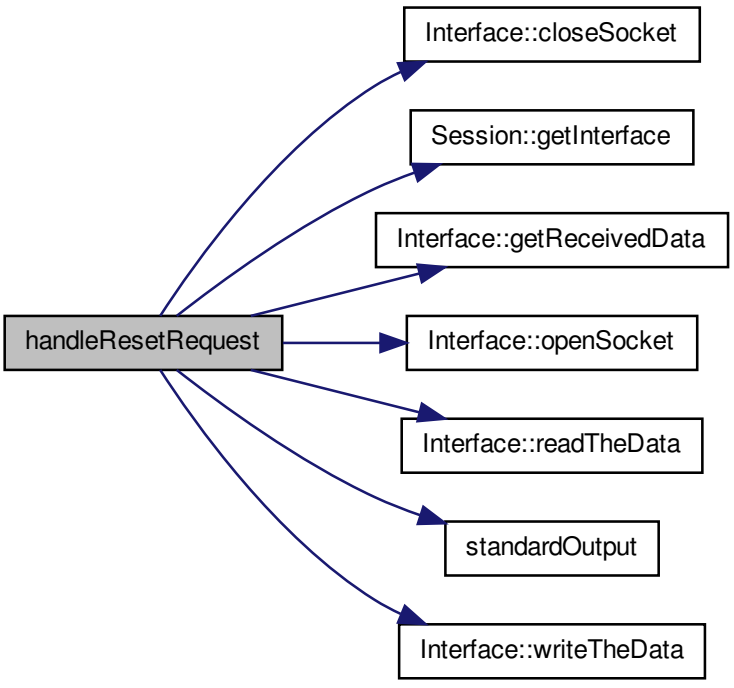
Return values

0	
1	

References Interface::closeSocket(), Session::getInterface(), Interface::getReceivedData(), Interface::openSocket(), Interface::readTheData(), RECEIVED_BYTES, standardOutput(), TIME_WAIT, and Interface::writeTheData().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

10.4.2.5 int handleSetRegisterRequest (Session *const *session, const float *value*)

Request to update the register value.

The function creates a connection to the device and sends a request for SetRegister using the SetRegister protocol of the form: {@s|value}, 2+4 bytes long.

Parameters

in	*session*	Pointer to the current session.
in	*value*	

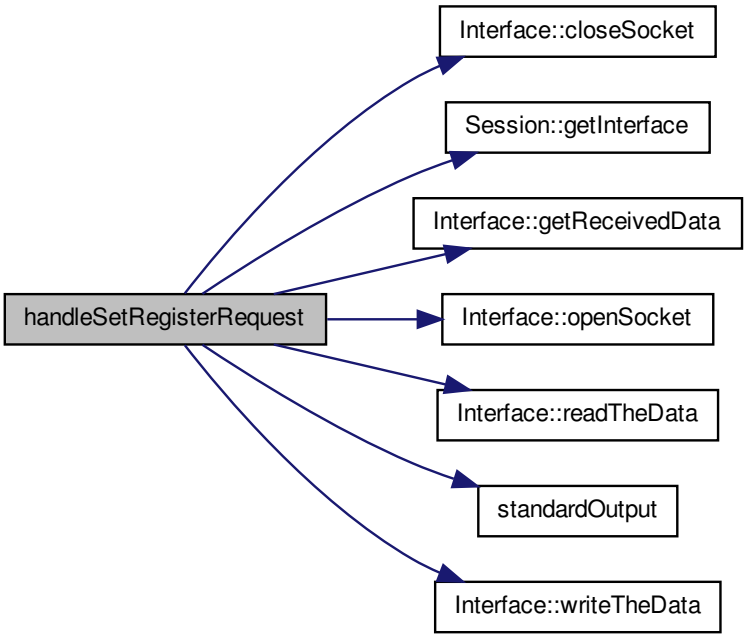
Return values

0	if no error occurs,
1	terminate with an error.

References Interface::closeSocket(), Session::getInterface(), Interface::getReceivedData(), Interface::openSocket(), Interface::readTheData(), RECEIVED_BYTES, standardOutput(), TIME_WAIT, and Interface::writeTheData().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

10.4.2.6 void setCommandLineParser (QCommandLineParser & *parser*)

Set Command Line Parser's parameters and optins.

Todo

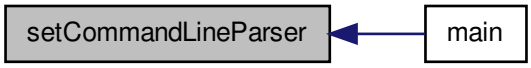
Parameters

in,out	*parser*	of the type QCommandLineParser

References ADJUST, CALIBR, DISCOVERY, INFO, PORT, RESET, and SETREG.

Referenced by main().

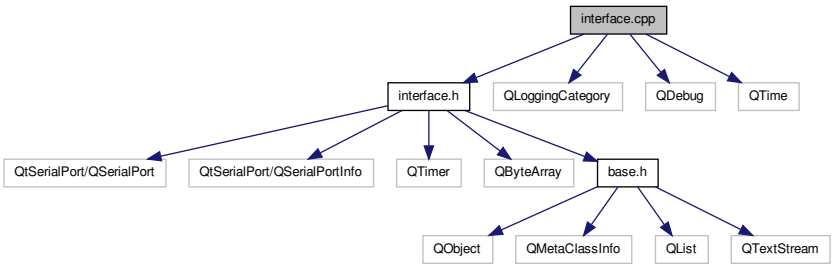
Here is the caller graph for this function:



10.5 interface.cpp File Reference

```
#include "interface.h"
#include <QLoggingCategory>
#include <QDebug>
#include <QTime>
```

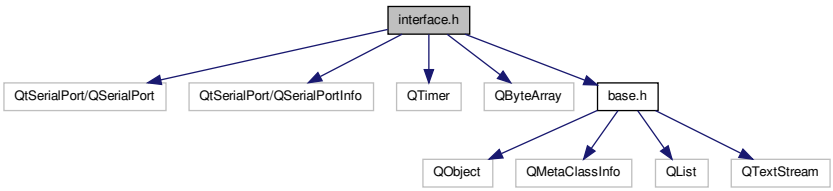
Include dependency graph for interface.cpp:



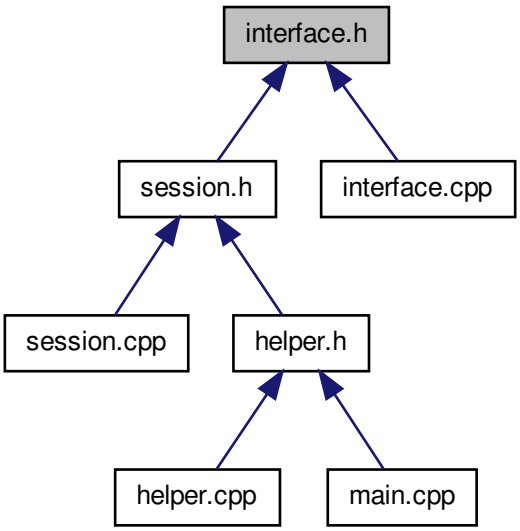
10.6 interface.h File Reference

```
#include <QtSerialPort/QSerialPort>
#include <QtSerialPort/QSerialPortInfo>
#include <QTimer>
#include <QByteArray>
#include "base.h"
```

Include dependency graph for interface.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Interface

  Interface The base class Interface provides for the communication between Bootloader Client and Host Device.

- class InterfaceSP

  The class InterfaceSP provides methods to access serial ports.

Macros

- #define REBOOT_WAIT 2500

10.6.1 Macro Definition Documentation

10.6.1.1 #define REBOOT_WAIT 2500

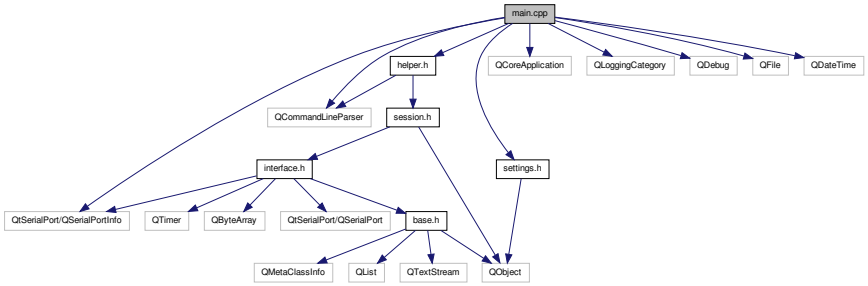
Referenced by InterfaceSP::openSocket().

10.7 main.cpp File Reference

The file contains two important functions, main() and logMessageOutput()

```
#include "helper.h"
#include "settings.h"
#include <QCoreApplication>
#include <QtSerialPort/QSerialPortInfo>
#include <QCommandLineParser>
#include <QLoggingCategory>
#include <QDebug>
#include <QFile>
#include <QDateTime>
```

Include dependency graph for main.cpp:



Functions

- static QT_USE_NAMESPACE QTextStream standardOutput (stdout)
- void logMessageOutput (const QtMsgType type, const QMessageLogContext &context, const QString &msg)

  The function logMessageOutput is a message handler.
- int main (int argc, char ∗argv[])

  main function of SynchroTime

Variables

- static QScopedPointer< QFile > m_logFile

10.7.1 Detailed Description

The file contains two important functions, main() and logMessageOutput() The main function executes an instance of a Qt-console application, sets it up with the specified special parameters, and installs a Qt message handler defined in the logMessageOutput function. In addition, a log journal of the application messages is set up.

10.7.2 Function Documentation

10.7.2.1 void logMessageOutput (const QtMsgType *type,* const QMessageLogContext & *context,* const QString & *msg*)

The function logMessageOutput is a message handler.

This function redirects the messages by their category (QtDebugMsg, QtInfoMsg, QtWarningMsg, QtCriticalMsg, QtFatalMsg) to the log file (m_logFile). The message handler is a function that prints out debug messages, warnings, critical and fatal error messages. The Qt library (debug mode) contains hundreds of warning messages that are printed when internal errors (usually invalid function arguments) occur. Qt built in release mode also contains such warnings unless QT_NO_WARNING_OUTPUT and/ or QT_NO_DEBUG_OUTPUT have been set during compilation. If you implement your own message handler, you get total control of these messages.

Parameters

in	*type*	of the type QtMsgType
in	*context*	of type QMessageLogContext
in	*msg*	of the type QString

Note

- The output of messages is also output to the terminal console. This is for debugging purposes.
- Additional information, such as a line of code, the name of the source file, function names cannot be displayed for the release of the program.

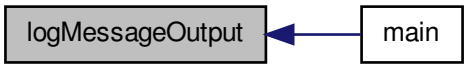
Warning

- The default message handler prints the message to the standard output under X11 or to the debugger under Windows. If it is a fatal message, the application aborts immediately.
- Only one message handler can be defined, since this is usually done on an application-wide basis to control debug output.

References m_logFile.

Referenced by main().

Here is the caller graph for this function:



10.7.2.2 int main (int *argc*, char * *argv[]*)

main function of SynchroTime

In this function, an instance of a Qt-console application app is executed and set up with the parameters entered.

Parameters

in	*argc*	the number of parameter.
in	*argv*	the command line options.

Returns

value of the function QApplication::exec() Enters the main event loop and waits until exit() is called. Returns the value that was set to exit() (which is 0 if exit() is called via quit()).

Note

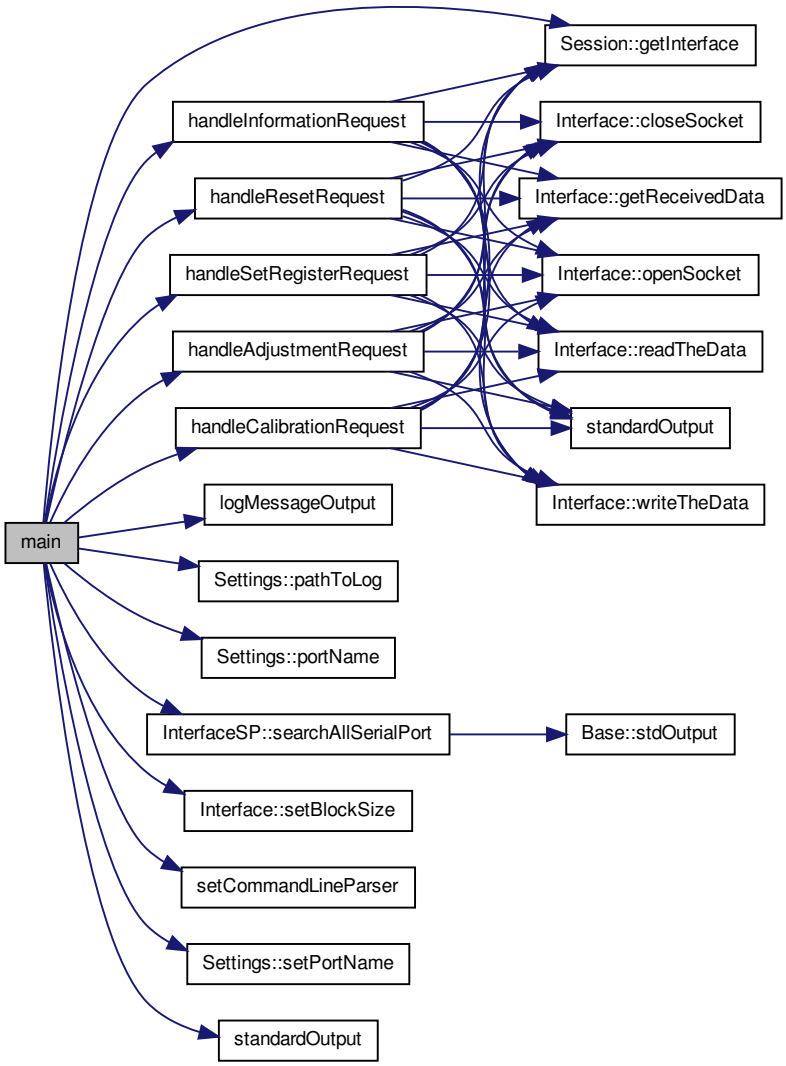
none

Warning

none

References ADJUST, CALIBR, DISCOVERY, Session::getInterface(), handleAdjustmentRequest(), handleCalibrationRequest(), handleInformationRequest(), handleResetRequest(), handleSetRegisterRequest(), INFO, logMessageOutput(), Settings::pathToLog(), PORT, Settings::portName(), RESET, InterfaceSP::searchAllSerialPort(), Interface::setBlockSize(), setCommandLineParser(), Settings::setPortName(), SETREG, and standardOutput().

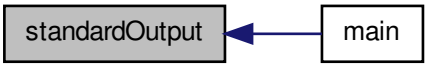
Here is the call graph for this function:



10.7.2.3 static QT_USE_NAMESPACE QTextStream standardOutput (stdout) [static]

Referenced by main().

Here is the caller graph for this function:



10.7.3 Variable Documentation

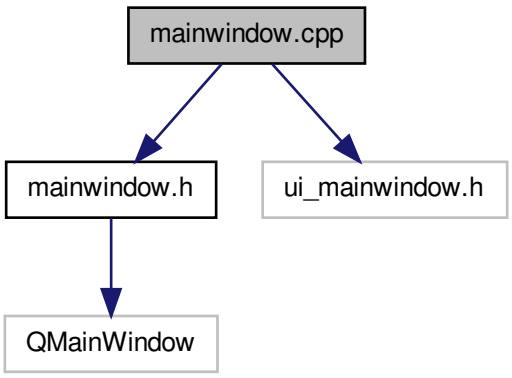
10.7.3.1 QScopedPointer<QFile> m_logFile [static]

Referenced by logMessageOutput().

10.8 mainwindow.cpp File Reference

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
```

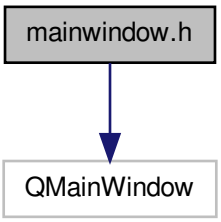
Include dependency graph for mainwindow.cpp:



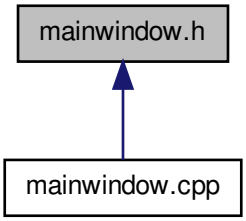
10.9 mainwindow.h File Reference

```
#include <QMainWindow>
```

Include dependency graph for mainwindow.h:



This graph shows which files directly or indirectly include this file:



Classes

- class MainWindow

Namespaces

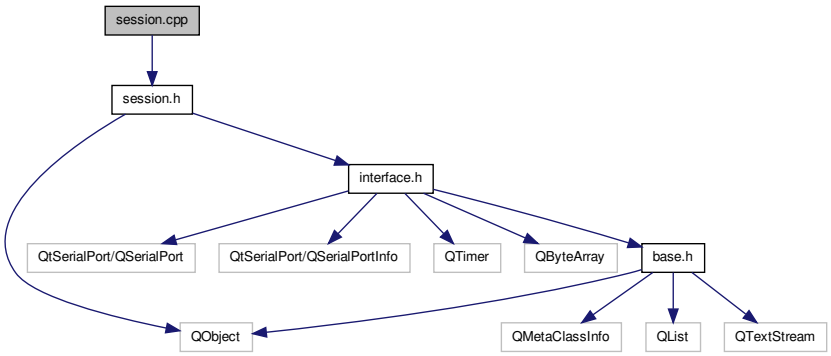
- Ui

10.10 README.md File Reference

10.11 session.cpp File Reference

```
#include "session.h"
```

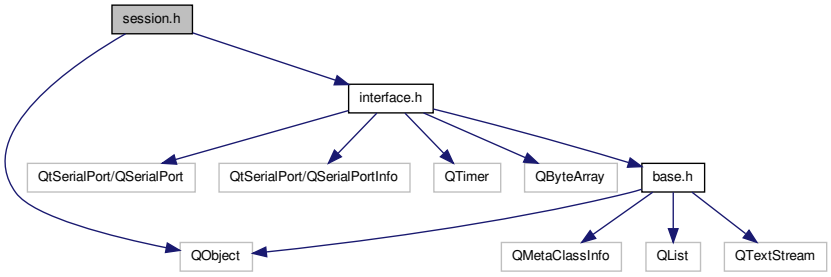
Include dependency graph for session.cpp:



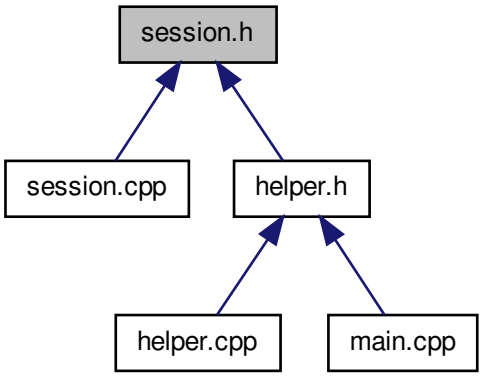
10.12 session.h File Reference

```
#include <QObject>
#include "interface.h"
```

Include dependency graph for session.h:



This graph shows which files directly or indirectly include this file:



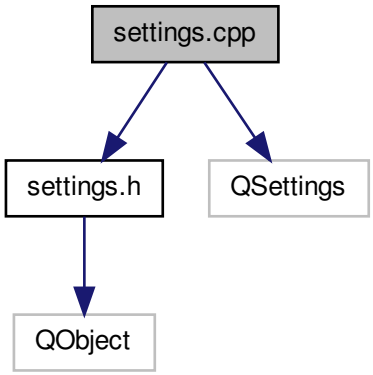
Classes

- class Session

10.13 settings.cpp File Reference

```
#include "settings.h"
#include <QSettings>
```

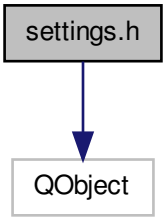
Include dependency graph for settings.cpp:



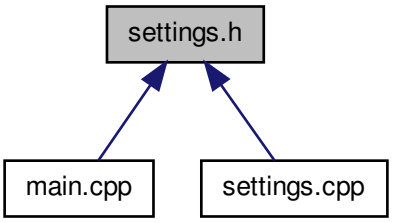
10.14 settings.h File Reference

```
#include <QObject>
```

Include dependency graph for settings.h:



This graph shows which files directly or indirectly include this file:



Classes

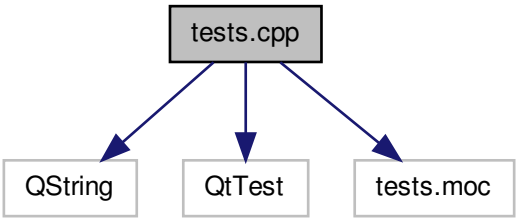
- class Settings

  The Settings class.

10.15 tests.cpp File Reference

```
#include <QString>
#include <QtTest>
#include "tests.moc"
```

Include dependency graph for tests.cpp:



Classes

- class Tests

Index