SSC Arduino Interface

1.0

Generated by Doxygen 1.8.8

Fri Sep 12 2014 00:40:16

Contents

Chapter 1

Interface

Interface code for the SSC Drill project.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

Class Documentation

4.1 CsvWriter Class Reference

The CsvWriter class for simple csv output.

`#include <csvwriter.h>`

Public Member Functions

- CsvWriter (QString target)

  baisc constructor
- void writeNumberVector (QList< double >)

  takes a Qlist of double and writes it into a file in ready to read csv format.

4.1.1 Detailed Description

The CsvWriter class for simple csv output.

A basic class for writing the output as a csv file whenever the user wishes to export recorded data by the software.

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

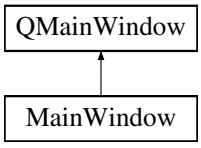
- csvwriter.h
- csvwriter.cpp

4.2 MainWindow Class Reference

The MainWindow class. Actually is the heart of the interface. All the methods which deal with the arduino itself are coded in the source code of this class.

`#include <mainwindow.h>`

Inheritance diagram for MainWindow:



Public Slots

- void serialRecieved ()

  fucntion that is executed everytime a serial message is recieved.
- void sendArduinoData (QString)

  send data to arduino via serial bus.
- void sendArduinoData ()

  similar to sendArduinoData(QString), but sends a null string and act the stream.

Signals

- void sendArduino (QString)

  send Arduino string.
- void sendArduino ()

  equivalent to the string counterpart.
- void serialRecievedProcessed ()

  signal that indicates that the incoming serial stream has been processed.
- void serialSent ()

  signal that indicates that the outgoung serial stream has been processed.
- void arduinoConnection (bool)

  signals that the arduino is connected or not. mainly for GUI purposes.
- void motorEnable (bool)

  signals that motor is enabled or not.
- void motorOutputCW (bool)

  signals that the motor is turning CW or CCW. mainly for GUI purposes.
- void motorReferenceCW (bool)

  same idea as motorOutputCW.
- void motorStalling (bool)

  signals if the motor is in intermittent operation. mainly for GUI purposes.
- void motorOutputRPM (QString)

  used to show a GUI representation of the arduino output RPM.
- void motorReferenceRPM (QString)

  used to show a GUI representation of the arduino reference RPM.
- void motorReferenceReciproFreq (QString)

  simple conversions to frequency. mainly for GUI purposes.
- void motorOutputReciproFreq (QString)

  same idea as motorReferenceReciproFreq(QString).
- void systemHeight (QString)

  signals the system height given by the arduino serial stream.
- void statusMessage (QString, int)

  signal used to show a message into the main windows status bar, with a timer.
- void statusMessage (QString)

  same as statusMessage(QString,int), without timer.

Public Member Functions

- MainWindow (QWidget *parent=0)

  default cosntructor
- void connectArduino ()

  the funtion wich tries to connect to and arduino.

4.2.1 Detailed Description

The MainWindow class. Actually is the heart of the interface. All the methods which deal with the arduino itself are coded in the source code of this class.

4.2.2 Member Function Documentation

4.2.2.1 void MainWindow::connectArduino ( )

the funtion wich tries to connect to and arduino.

This function tries to be as cross-platform possible, by iterating in the lsit given by the QtSerial library and attempting connection with the first possibility it finds. There is a draw back to this approach: The assumption that only on arduino is connected to the machine.

By calling the function the internal variable serial is manipulated in this respect.

4.2.2.2 void MainWindow::motorOutputReciproFreq (QString) [signal]

same idea as motorReferenceReciproFreq(QString).

See also

motorOutputRPM(QString)

4.2.2.3 void MainWindow::motorReferenceCW (bool) [signal]

same idea as motorOutputCW.

See also

motorOutputCW(bool)

4.2.2.4 void MainWindow::motorReferenceReciproFreq (QString) [signal]

simple conversions to frequency. mainly for GUI purposes.

See also

motorReferenceRPM(Qstring)

4.2.2.5 void MainWindow::sendArduino (QString) [signal]

send Arduino string.

The signal to send Arduino any commands and numbers within the string. If the arduino is not connect, however, a message is printed into the status bar telling that the arduino is not connected and no action takes place.

4.2.2.6 void MainWindow::sendArduino ( ) [signal]

equivalent to the string counterpart.

See also

sendArduino(QString);

4.2.2.7 void MainWindow::sendArduinoData (QString *data*) [slot]

send data to arduino via serial bus.

Takes the string given in the slot and sends it in the correct form to the arduino via serial bus. The string can be any command that the arduino recognizes (invalid commands and chars are ignored by the arduino internal parser).

If the serial is still being processes; the string is appended to the end of the outgoing stream.

4.2.2.8 void MainWindow::serialRecieved ( ) [slot]

fucntion that is executed everytime a serial message is recieved.

This function processes all the serial data recieved by the Arduino. Since it does not behave like the arduino processing does, being faster in this respect; a internal buffer is necessary to gather all the data until a linefeed is issued. If this happens, the buffer is taken and processed according to the minimal standard adopted in the arduino:

Enabled; Stalling; reference RPM; output RPM; output Height;

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

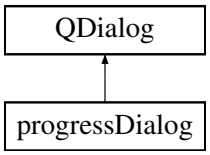
- mainwindow.h
- mainwindow.cpp

4.3 progressDialog Class Reference

The progressDialog class is a simple progress bar for value recording.

`#include <progressdialog.h>`

Inheritance diagram for progressDialog:



Public Slots

- void recordValue (double)

  tkae a double value and add it to the stack.
- void recordValue (QString)

  recordValue placeholder.

Signals

- void setProgress (int)

  simple GUI fucntion.

Public Member Functions

- progressDialog (QWidget ∗parent=0, int dim_=50, QString tracked="")

  progressDialog constructor.

4.3.1 Detailed Description

The progressDialog class is a simple progress bar for value recording.

This class manages both the gui representing the value recording and the recording itself, by storing the samples internally. The sample size is given in the class construction.

The recording itself happens via slots from the mainwindow to one instance of this class.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 progressDialog::progressDialog (QWidget * *parent* = 0, int *dim_* = 50, QString *tracked* = " ") [explicit]

progressDialog constructor.

The constructor must recieve the sample size which it will record so that the sample size can be tracked and managed internally. The name of the title should also be given for aesthetics.

Parameters

parent	generally the main window.
*dim_*	the samplign size that the object will manage.
tracked	the actual title of the GUI.

4.3.3 Member Function Documentation

4.3.3.1 void progressDialog::recordValue (double *value*) [slot]

tkae a double value and add it to the stack.

take the double given in the signal and add it to the internal sampel stack, managing the GUI and internal variables automatically. When the stack reaches the desired sample size, a finished signal is emited and a prompt for save is given.

4.3.3.2 void progressDialog::recordValue (QString *value*) [slot]

recordValue placeholder.

This signal is a placeholder for the recordValue signal, which tries to convert the string to double format and the calls the correct recordValue(double).

4.3.3.3 void progressDialog::setProgress (int) [signal]

simple GUI fucntion.

This takes an int and represent it accordingly into the progress bar, making it show another percentage.

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

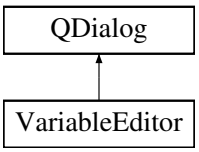
- progressdialog.h
- progressdialog.cpp

4.4 VariableEditor Class Reference

The VariableEditor class is a simple global internal variable editor.

`#include <variableeditor.h>`

Inheritance diagram for VariableEditor:



Signals

- void s_setVars (QList< int >, QList< double >)

  s_setVars signals the new variables to be set in the main window.

Public Member Functions

- VariableEditor (QWidget *parent=0, QList< int > int_vars_=QList< int >(), QList< double > double_vars_=QList< double >())

  VariableEditor contructor, needs a couple of parameters.

4.4.1 Detailed Description

The VariableEditor class is a simple global internal variable editor.

The goal is that the object created changes the global variables determined into the mainwindow such as sample size and height extrema.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 VariableEditor::VariableEditor (QWidget * *parent* = 0, QList< int > *int_vars_* = QList<int>(), QList< double > *double_vars_* = QList<double>()) [explicit]

VariableEditor contructor, needs a couple of parameters.

To make the variables appear in the correct way, they are passed as lists by the mainwindow object and then processed internally by this class.

Although the numbering and rows managning is automatic, the row labels are not. If more global variables are defined and need to be managed, the source code must be modified accordingly, in this class and in the MainWindow class.

Parameters

parent	The usual GUI parent. Genrally the mainwindow.
*int_vars_*	the input list of integer variables.
*double_vars_*	the input list of double/float variables.

4.4.3 Member Function Documentation

4.4.3.1 void VariableEditor::s_setVars (QList< int >, QList< double >) [signal]

s_setVars signals the new variables to be set in the main window.

The variable chaging is not entirelly general in the mainwindow class and if a new global variable is created and needs to be managed, as stated in VariableEditor() description;the correspondent source code of the MainWindow class also needs to be changed. Although it is a straighforward change.

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

- variableeditor.h
- variableeditor.cpp