FlyCAM

Generated by Doxygen 1.8.12

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

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

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 Ui Namespace Reference

CommonInterfaceSelector class Handles the selection of interfaces.

5.1.1 Detailed Description

CommonInterfaceSelector class Handles the selection of interfaces.

Chapter 6

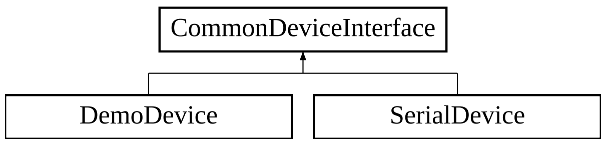
Class Documentation

6.1 CommonDeviceInterface Class Reference

The Abstract Base Class: Common Device Interface A generic template for interfaces.

`#include <commondeviceinterface.h>`

Inheritance diagram for CommonDeviceInterface:



Public Member Functions

- CommonDeviceInterface ()
- virtual ∼CommonDeviceInterface ()
- virtual void syncRX ()=0
- virtual void syncTX ()=0
- virtual bool isReady ()=0
- virtual bool startDevice ()=0
- virtual void stopDevice ()=0
- virtual void setDefaults ()=0
- virtual bool empty ()=0
- virtual void flushRX ()=0
- virtual void flushTX ()=0
- virtual void pushByte (FlyByte)=0
- virtual void pushPacket (FlyPacket)=0
- virtual FlyByte popByte ()=0
- virtual FlyPacket popPacket ()=0
- virtual QString name ()=0

6.1.1 Detailed Description

The Abstract Base Class: Common Device Interface A generic template for interfaces.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 CommonDeviceInterface()

```
CommonDeviceInterface::CommonDeviceInterface ( )  [inline]
```

6.1.2.2 ∼CommonDeviceInterface()

```
virtual CommonDeviceInterface::∼CommonDeviceInterface ( )  [inline], [virtual]
```

6.1.3 Member Function Documentation

6.1.3.1 empty()

```
virtual bool CommonDeviceInterface::empty ( )  [pure virtual]
```

Implemented in SerialDevice, and DemoDevice.

6.1.3.2 flushRX()

```
virtual void CommonDeviceInterface::flushRX ( )  [pure virtual]
```

Implemented in SerialDevice, and DemoDevice.

6.1.3.3 flushTX()

```
virtual void CommonDeviceInterface::flushTX ( )  [pure virtual]
```

Implemented in SerialDevice, and DemoDevice.

6.1.3.4 isReady()

```
virtual bool CommonDeviceInterface::isReady ( )  [pure virtual]
```

Implemented in SerialDevice, and DemoDevice.

6.1.3.5 name()

```
virtual QString CommonDeviceInterface::name ( )  [pure virtual]
```

Implemented in SerialDevice, and DemoDevice.

6.1.3.6 popByte()

```
virtual FlyByte CommonDeviceInterface::popByte ( )  [pure virtual]
```

Implemented in SerialDevice, and DemoDevice.

6.1.3.7 popPacket()

```
virtual FlyPacket CommonDeviceInterface::popPacket ( )  [pure virtual]
```

Implemented in SerialDevice, and DemoDevice.

6.1.3.8 pushByte()

```
virtual void CommonDeviceInterface::pushByte (
             FlyByte  )  [pure virtual]
```

Implemented in SerialDevice, and DemoDevice.

6.1.3.9 pushPacket()

```
virtual void CommonDeviceInterface::pushPacket (
             FlyPacket  )  [pure virtual]
```

Implemented in SerialDevice, and DemoDevice.

6.1.3.10 setDefaults()

```
virtual void CommonDeviceInterface::setDefaults ( )  [pure virtual]
```

Implemented in SerialDevice, and DemoDevice.

6.1.3.11 startDevice()

```
virtual bool CommonDeviceInterface::startDevice ( )  [pure virtual]
```

Implemented in SerialDevice, and DemoDevice.

6.1.3.12 stopDevice()

```
virtual void CommonDeviceInterface::stopDevice ( )  [pure virtual]
```

Implemented in SerialDevice, and DemoDevice.

6.1.3.13 syncRX()

```
virtual void CommonDeviceInterface::syncRX ( )  [pure virtual]
```

Implemented in SerialDevice, and DemoDevice.

6.1.3.14 syncTX()

```
virtual void CommonDeviceInterface::syncTX ( )  [pure virtual]
```

Implemented in SerialDevice, and DemoDevice.

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

- FESS-GUI/commondeviceinterface.h

6.2 CommonInterfaceManager Class Reference

CommonInterfaceManager Class Manages interfaces set in the interface selector menu.

```
#include <commoninterfacemanager.h>
```

Public Member Functions

- CommonInterfaceManager ()

    CommonInterfaceManager::CommonInterfaceManager Set the device to NULL.
- ∼CommonInterfaceManager ()

    CommonInterfaceManager::∼CommonInterfaceManager Deletes the current device.
- CommonDeviceInterface ∗ getCurrentInterface ()

    CommonInterfaceManager::getCurrentInterface Get the current interface.
- void setCurrentInterface (CommonDeviceInterface ∗)

    CommonInterfaceManager::setCurrentInterface Sets the curret device interface.
- bool isADeviceSet ()

    CommonInterfaceManager::isADeviceSet Determines if a device has been set in the menu.
- void closeCurrentInterface ()

    CommonInterfaceManager::closeCurrentInterface Closes the current interface.

6.2.1 Detailed Description

CommonInterfaceManager Class Manages interfaces set in the interface selector menu.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 CommonInterfaceManager()

```
CommonInterfaceManager::CommonInterfaceManager ( )
```

CommonInterfaceManager::CommonInterfaceManager Set the device to NULL.

6.2.2.2 ∼CommonInterfaceManager()

```
CommonInterfaceManager::∼CommonInterfaceManager ( )
```

CommonInterfaceManager::∼CommonInterfaceManager Deletes the current device.

6.2.3 Member Function Documentation

6.2.3.1 closeCurrentInterface()

```
void CommonInterfaceManager::closeCurrentInterface ( )
```

CommonInterfaceManager::closeCurrentInterface Closes the current interface.

6.2.3.2 getCurrentInterface()

```
CommonDeviceInterface * CommonInterfaceManager::getCurrentInterface ( )
```

CommonInterfaceManager::getCurrentInterface Get the current interface.

Returns

CommonDeviceInterface∗ (CommonDeviceInterface Instance Pointer,NULL)

6.2.3.3 isADeviceSet()

```
bool CommonInterfaceManager::isADeviceSet ( )
```

CommonInterfaceManager::isADeviceSet Determines if a device has been set in the menu.

Returns

bool (true=Yes, false=No)

6.2.3.4 setCurrentInterface()

```
void CommonInterfaceManager::setCurrentInterface (
            CommonDeviceInterface * newInterface )
```

CommonInterfaceManager::setCurrentInterface Sets the curret device interface.

Parameters

CommonDeviceInterface∗	

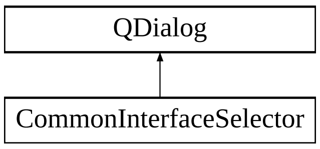
The documentation for this class was generated from the following files:

- FESS-GUI/commoninterfacemanager.h
- FESS-GUI/commoninterfacemanager.cpp

6.3 CommonInterfaceSelector Class Reference

```
#include <commoninterfaceselector.h>
```

Inheritance diagram for CommonInterfaceSelector:



Public Member Functions

- CommonInterfaceSelector (CommonInterfaceManager ∗, QWidget ∗parent=0)

  CommonInterfaceSelector::CommonInterfaceSelector GUI window for configuring devices.
- ∼CommonInterfaceSelector ()

  CommonInterfaceSelector::∼CommonInterfaceSelector Destructor deletes ui and errorHandler objects.

6.3.1 Constructor & Destructor Documentation

6.3.1.1 CommonInterfaceSelector()

```
CommonInterfaceSelector::CommonInterfaceSelector (
            CommonInterfaceManager ∗ commonManager,
            QWidget ∗ parent = 0 )  [explicit]
```

CommonInterfaceSelector::CommonInterfaceSelector GUI window for configuring devices.

Parameters

CommonInterfaceManager∗	(Interface Manager Instance)
QWidget	(Parent Window)

6.3.1.2 ∼CommonInterfaceSelector()

```
CommonInterfaceSelector::∼CommonInterfaceSelector ( )
```

CommonInterfaceSelector::∼CommonInterfaceSelector Destructor deletes ui and errorHandler objects.

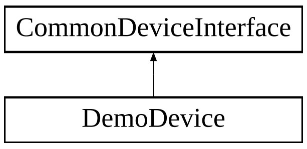
The documentation for this class was generated from the following files:

- FESS-GUI/commoninterfaceselector.h
- FESS-GUI/commoninterfaceselector.cpp

6.4 DemoDevice Class Reference

```
#include <demodevice.h>
```

Inheritance diagram for DemoDevice:



Public Member Functions

- DemoDevice ()

  DemoDevice::DemoDevice Configures a new DemoDevice with its default values.
- ∼DemoDevice ()

  DemoDevice::∼DemoDevice Provided if needed in the future, currently no functionality.
- void syncRX ()

  DemoDevice::syncRX Is the demo interface's implementation of syncRX. Generates values for testing and pushes them into the interface's RX Transmit Buffer.
- void syncTX ()

  DemoDevice::syncRX Is the demo interface's implementation of syncRX. Reacts to the interfaces commands (Only Emergency Stop currently)
- bool isReady ()

  DemoDevice::isReady Is the demo interface's implementation of isReady. Get the status of the demo interface.
- bool startDevice ()

  DemoDevice::startDevice Is the demo interface's implementation of startDevice. Starts the demo interface and updates status.
- void stopDevice ()

  DemoDevice::stopDevice Is the demo interface's implementation of stopDevice. Stops the demo interface and updates status.
- void setDefaults ()

  DemoDevice::stopDevice Is the demo interface's implementation of setDefaults. Sets the serial devices to the default values.
- bool empty ()

  DemoDevice::empty Is the demo interface's implementation of empty.
- void flushRX ()

  DemoDevice::flushRX Is the demo interface's implementation of flushRX. Empties the RX Transmit Buffer.
- void flushTX ()

  DemoDevice::flushTX Is the demo interface's implementation of flushTX. Empties the TX Transmit Buffer.
- void pushByte (FlyByte)

  DemoDevice::popByte Is the demo interface's implementation of pushByte.
- void pushPacket (FlyPacket)

  DemoDevice::popByte Is the demo interface's implementation of pushPacket.
- FlyByte popByte ()

  DemoDevice::popByte Is the demo interface's implementation of popByte.
- FlyPacket popPacket ()

  DemoDevice::popByte Is the demo interface's implementation of popPacket.
- QString name ()

  DemoDevice::name Is the demo interface's implementation of name.

6.4.1 Constructor & Destructor Documentation

6.4.1.1 DemoDevice()

```
DemoDevice::DemoDevice ( )
```

DemoDevice::DemoDevice Configures a new DemoDevice with its default values.

6.4.1.2 ∼DemoDevice()

```
DemoDevice::∼DemoDevice ( )
```

DemoDevice::∼DemoDevice Provided if needed in the future, currently no functionality.

6.4.2 Member Function Documentation

6.4.2.1 empty()

```
bool DemoDevice::empty ( )  [virtual]
```

DemoDevice::empty Is the demo interface's implementation of empty.

Returns

bool (true=Empty, false=Not Empty)

Implements CommonDeviceInterface.

6.4.2.2 flushRX()

```
void DemoDevice::flushRX ( )  [virtual]
```

DemoDevice::flushRX Is the demo interface's implementation of flushRX. Empties the RX Transmit Buffer.

Implements CommonDeviceInterface.

6.4.2.3 flushTX()

```
void DemoDevice::flushTX ( )  [virtual]
```

DemoDevice::flushTX Is the demo interface's implementation of flushTX. Empties the TX Transmit Buffer.

Implements CommonDeviceInterface.

6.4.2.4 isReady()

```
bool DemoDevice::isReady ( )  [virtual]
```

DemoDevice::isReady Is the demo interface's implementation of isReady. Get the status of the demo interface.

Returns

bool (true=Ready, false=Not Ready)

Implements CommonDeviceInterface.

6.4.2.5 name()

```
QString DemoDevice::name ( )  [virtual]
```

DemoDevice::name Is the demo interface's implementation of name.

Returns

Qstring ("Demo Device")

Implements CommonDeviceInterface.

6.4.2.6 popByte()

```
FlyByte DemoDevice::popByte ( )  [virtual]
```

DemoDevice::popByte Is the demo interface's implementation of popByte.

Returns

FlyByte (From the RX Transmit Buffer).

Implements CommonDeviceInterface.

6.4.2.7 popPacket()

```
FlyPacket DemoDevice::popPacket ( )  [virtual]
```

DemoDevice::popByte Is the demo interface's implementation of popPacket.

Returns

FlyPacket (From the RX Transmit Buffer).

Implements CommonDeviceInterface.

6.4.2.8 pushByte()

```
void DemoDevice::pushByte (
             FlyByte dataByte )  [virtual]
```

DemoDevice::popByte Is the demo interface's implementation of pushByte.

Parameters

FlyByte	(Adds it to the TX Transmit Buffer).

Implements CommonDeviceInterface.

6.4.2.9 pushPacket()

```
void DemoDevice::pushPacket (
            FlyPacket dataPacket )  [virtual]
```

DemoDevice::popByte Is the demo interface's implementation of pushPacket.

Parameters

FlyPacket	(Adds to the TX Transmit Buffer).

Implements CommonDeviceInterface.

6.4.2.10 setDefaults()

```
void DemoDevice::setDefaults ( )  [virtual]
```

DemoDevice::stopDevice Is the demo interface's implementation of setDefaults. Sets the serial devices to the default values.

Implements CommonDeviceInterface.

6.4.2.11 startDevice()

```
bool DemoDevice::startDevice ( )  [virtual]
```

DemoDevice::startDevice Is the demo interface's implementation of startDevice. Starts the demo interface and updates status.

Returns

bool (true=Started, false=Not Ready)

Implements CommonDeviceInterface.

6.4.2.12 stopDevice()

```
void DemoDevice::stopDevice ( )  [virtual]
```

DemoDevice::stopDevice Is the demo interface's implementation of stopDevice. Stops the demo interface and updates status.

Implements CommonDeviceInterface.

6.4.2.13 syncRX()

```
void DemoDevice::syncRX ( )  [virtual]
```

DemoDevice::syncRX Is the demo interface's implementation of syncRX. Generates values for testing and pushes them into the interface's RX Transmit Buffer.

Implements CommonDeviceInterface.

6.4.2.14 syncTX()

```
void DemoDevice::syncTX ( )  [virtual]
```

DemoDevice::syncRX Is the demo interface's implementation of syncRX. Reacts to the interfaces commands (Only Emergency Stop currently)

Implements CommonDeviceInterface.

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

- FESS-GUI/demodevice.h
- FESS-GUI/demodevice.cpp

6.5 FlyPacket Class Reference

```
#include <flypacket.h>
```

Public Member Functions

- FlyPacket ()

  FlyPacket::FlyPacket Set the default values.
- FlyPacket (FlyByte, int)

  FlyPacket::FlyPacket Set the default values, header and data.
- FlyPacket (FlyByte, float)

  FlyPacket::FlyPacket Set the default values, header and data.
- ∼FlyPacket ()

  FlyPacket::∼FlyPacket Empty Destructor.
- void setValue (int)

  FlyPacket:setValue Converts a int into a byte array and inserts it into the packet.
- void setValue (float)

  FlyPacket:setValue Converts a float into a byte array and inserts it into the packet.
- void setCommand (FlyByte)

  FlyPacket::setCommand Sets the header of the packet and generates the footer.
- void writeByte (FlyByte)

  FlyPacket::writeByte Writes one byte into the packet until it is full. Sequence: Header, Data, ..., Data, Footer.
- int getInt ()

  FlyPacket::getInt Converts the internal data bytes into a int.
- float getFloat ()

  FlyPacket::getFloat Converts the internal data bytes into a float.

- FlyByte readByte ()

    FlyPacket::readByte Reads one byte from the packet and advances to the next byte.
- FlyByte getCommand ()

    FlyPacket::getCommand Get the header byte.
- void reset ()

    FlyPacket::reset Resets the packet to all zeros: data,header and footer.
- bool isWriteable ()

    FlyPacket::isWriteable Check the packet to see if it has space for more bytes.
- bool isReadable ()

    FlyPacket::isReadable Check the packet to see if all the internal bytes have been read.
- bool isValidPacket ()

    FlyPacket::isValidPacket() Check if the packet is valid.
- bool isValidCommand ()

    FlyPacket::isValidCommand Check if the header is valid.
- FlyByte getMaxSize ()

    FlyPacket::getMaxSize Get the maximum packet size.

6.5.1 Constructor & Destructor Documentation

6.5.1.1 FlyPacket() [1/3]

```
FlyPacket::FlyPacket ( )
```

FlyPacket::FlyPacket Set the default values.

6.5.1.2 FlyPacket() [2/3]

```
FlyPacket::FlyPacket (
            FlyByte commandByte,
            int dataValue )
```

FlyPacket::FlyPacket Set the default values, header and data.

Parameters

FlyByte	(Header Byte)
int	(Data Byte)

6.5.1.3 FlyPacket() [3/3]

```
FlyPacket::FlyPacket (
            FlyByte commandByte,
            float dataValue )
```

FlyPacket::FlyPacket Set the default values, header and data.

Parameters

FlyByte	(Header Byte)
float	(Data Byte)

6.5.1.4 ∼FlyPacket()

```
FlyPacket::∼FlyPacket ( )
```

FlyPacket::∼FlyPacket Empty Destructor.

6.5.2 Member Function Documentation

6.5.2.1 getCommand()

```
FlyByte FlyPacket::getCommand ( )
```

FlyPacket::getCommand Get the header byte.

Returns

FlyByte (Header)

6.5.2.2 getFloat()

```
float FlyPacket::getFloat ( )
```

FlyPacket::getFloat Converts the internal data bytes into a float.

Returns

float (Data)

6.5.2.3 getInt()

```
int FlyPacket::getInt ( )
```

FlyPacket::getInt Converts the internal data bytes into a int.

Returns

int (Data)

6.5.2.4 getMaxSize()

```
FlyByte FlyPacket::getMaxSize ( )
```

FlyPacket::getMaxSize Get the maximum packet size.

Returns

int (PACKET_SIZE)

6.5.2.5 isReadable()

```
bool FlyPacket::isReadable ( )
```

FlyPacket::isReadable Check the packet to see if all the internal bytes have been read.

Returns

bool (true=yes, false=no)

6.5.2.6 isValidCommand()

```
bool FlyPacket::isValidCommand ( )
```

FlyPacket::isValidCommand Check if the header is valid.

Returns

bool (true=valid, false=invalid)

6.5.2.7 isValidPacket()

```
bool FlyPacket::isValidPacket ( )
```

FlyPacket::isValidPacket() Check if the packet is valid.

Returns

bool (true=valid, false=invalid)

6.5.2.8 isWriteable()

```
bool FlyPacket::isWriteable ( )
```

FlyPacket::isWriteable Check the packet to see if it has space for more bytes.

Returns

bool (true=yes, false=no)

6.5.2.9 readByte()

```
FlyByte FlyPacket::readByte ( )
```

FlyPacket::readByte Reads one byte from the packet and advances to the next byte.

Returns

FlyByte (Byte)

6.5.2.10 reset()

```
void FlyPacket::reset ( )
```

FlyPacket::reset Resets the packet to all zeros: data,header and footer.

6.5.2.11 setCommand()

```
void FlyPacket::setCommand (
            FlyByte generalByte )
```

FlyPacket::setCommand Sets the header of the packet and generates the footer.

Parameters

FlyByte	(Header Byte)

6.5.2.12 setValue() [1/2]

```
void FlyPacket::setValue (
            int dataValue )
```

FlyPacket:setValue Converts a int into a byte array and inserts it into the packet.

Parameters

int	(Data)

6.5.2.13 setValue() [2/2]

```
void FlyPacket::setValue (
            float dataValue )
```

FlyPacket:setValue Converts a float into a byte array and inserts it into the packet.

Parameters

float	(Data)

6.5.2.14 writeByte()

```
void FlyPacket::writeByte (
            FlyByte generalByte )
```

FlyPacket::writeByte Writes one byte into the packet until it is full. Sequence: Header, Data, ..., Data, Footer.

Parameters

FlyByte	(Bytes)

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

- FESS-GUI/flypacket.h
- FESS-GUI/flypacket.cpp

6.6 FlyQueue Class Reference

FlyQueue Class A wrapper to allow for reuse on the microcontroller.

```
#include <flyqueue.h>
```

Public Member Functions

- FlyQueue ()

  FlyQueue::FlyQueue Sets the default values for the queue.
- ∼FlyQueue ()

  FlyQueue::∼FlyQueue Empty Destructor.
- void clear ()

  FlyQueue::clear Empties the queue.
- void reset ()

  FlyQueue::reset Restores the default values and empties the queue.
- void setSize (int)

  FlyQueue::setSize Sets the maximum size for the queue.
- bool isEmpty ()

  FlyQueue::isEmpty Check if the queue is empty.
- FlyPacket pop ()

  FlyQueue::pop Gets the first packet from the queue and removes it from the queue.
- void push (FlyPacket)

  FlyQueue::push Puts a packet at the end of the queue.
- FlyPacket operator[ ] (const unsigned int)

  FlyQueue::operator[ ] Allows random access into the queue.

6.6.1 Detailed Description

FlyQueue Class A wrapper to allow for reuse on the microcontroller.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 FlyQueue()

```
FlyQueue::FlyQueue ( )
```

FlyQueue::FlyQueue Sets the default values for the queue.

6.6.2.2 ∼FlyQueue()

```
FlyQueue::∼FlyQueue ( )
```

FlyQueue::∼FlyQueue Empty Destructor.

6.6.3 Member Function Documentation

6.6.3.1 clear()

```
void FlyQueue::clear ( )
```

FlyQueue::clear Empties the queue.

6.6.3.2 isEmpty()

```
bool FlyQueue::isEmpty ( )
```

FlyQueue::isEmpty Check if the queue is empty.

Returns

bool (True=Empty, False=Not Empty)

6.6.3.3 operator[]()

```
FlyPacket FlyQueue::operator[ ] (
             const unsigned int index )
```

FlyQueue::operator[ ] Allows random access into the queue.

Parameters

int	(index in queue)

Returns

FlyPacket

6.6.3.4 pop()

```
FlyPacket FlyQueue::pop ( )
```

FlyQueue::pop Gets the first packet from the queue and removes it from the queue.

Returns

FlyPacket

6.6.3.5 push()

```
void FlyQueue::push (
            FlyPacket incomingPacket )
```

FlyQueue::push Puts a packet at the end of the queue.

6.6.3.6 reset()

```
void FlyQueue::reset ( )
```

FlyQueue::reset Restores the default values and empties the queue.

6.6.3.7 setSize()

```
void FlyQueue::setSize (
            int bufferSize )
```

FlyQueue::setSize Sets the maximum size for the queue.

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

- FESS-GUI/flyqueue.h
- FESS-GUI/flyqueue.cpp

6.7 FlywheelOperation Class Reference

The FlywheelOperation class handles all operations involving the flywheel.

```
#include <flywheeloperation.h>
```

Public Member Functions

- FlywheelOperation ()

  FlywheelOperation::FlywheelOperation Initializes variables to default values.
- FlywheelOperation (CommonDeviceInterface ∗)

  FlywheelOperation::FlywheelOperation Sets the deviceInterface and initializes variables to default values.
- ∼FlywheelOperation ()

  FlywheelOperation::∼FlywheelOperation Deletes pointers.
- void sync ()
- void setDefaults ()

  FlywheelOperation::setDefaults Sets default values for member variables.
- void setVelocity (float)

  FlywheelOperation::setVelocity Sets the velocity of the flywheel.
- void setAcceleration (float)

  FlywheelOperation::setAcceleration Sets the acceleration of the flywheel.
- void setJerk (float)

  FlywheelOperation::setJerk Sets the jerk of the flywheel.
- void setMotion (float, float, float)

  FlywheelOperation::setMotion Sets all motion parameters for the flywheel.
- float getVelocity ()

  FlywheelOperation::getVelocity Gets the current velocity of the flywheel.
- float getAcceleration ()

  FlywheelOperation::getAcceleration Gets the current acceleration of the flywheel.
- float getJerk ()

  FlywheelOperation::getJerk Gets the current jerk of the flywheel.
- void emergencyStop ()

  FlywheelOperation::emergencyStop Sends a command to stop the flywheel immediately.
- void setInterface (CommonDeviceInterface ∗)

  FlywheelOperation::setInterface Sets the device interface.
- QPointF getUpperDisplacement ()

  FlywheelOperation::getUpperDisplacement Gets the upper displacement of the flywheel.
- QPointF getLowerDisplacement ()

  FlywheelOperation::getLowerDisplacement Gets the lower displacement of the flywheel.
- QPointF getRotationalPosition ()

  FlywheelOperation::getRotationalPosition Gets the rotational position of the flywheel.

6.7.1 Detailed Description

The FlywheelOperation class handles all operations involving the flywheel.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 FlywheelOperation() [1/2]

```
FlywheelOperation::FlywheelOperation ( )
```

FlywheelOperation::FlywheelOperation Initializes variables to default values.

6.7.2.2 FlywheelOperation() [2/2]

```
FlywheelOperation::FlywheelOperation (
            CommonDeviceInterface ∗ deviceInterface )
```

FlywheelOperation::FlywheelOperation Sets the deviceInterface and initializes variables to default values.

Parameters

deviceInterface	

6.7.2.3 ∼FlywheelOperation()

```
FlywheelOperation::∼FlywheelOperation ( )
```

FlywheelOperation::∼FlywheelOperation Deletes pointers.

6.7.3 Member Function Documentation

6.7.3.1 emergencyStop()

```
void FlywheelOperation::emergencyStop ( )
```

FlywheelOperation::emergencyStop Sends a command to stop the flywheel immediately.

6.7.3.2 getAcceleration()

```
float FlywheelOperation::getAcceleration ( )
```

FlywheelOperation::getAcceleration Gets the current acceleration of the flywheel.

Returns

The current acceleration of the flywheel.

6.7.3.3 getJerk()

```
float FlywheelOperation::getJerk ( )
```

FlywheelOperation::getJerk Gets the current jerk of the flywheel.

Returns

The current jerk of the flywheel.

6.7.3.4 getLowerDisplacement()

```
QPointF FlywheelOperation::getLowerDisplacement ( )
```

FlywheelOperation::getLowerDisplacement Gets the lower displacement of the flywheel.

Returns

The lower displacement of the flywheel.

6.7.3.5 getRotationalPosition()

```
QPointF FlywheelOperation::getRotationalPosition ( )
```

FlywheelOperation::getRotationalPosition Gets the rotational position of the flywheel.

Returns

The rotational position of the flywheel.

6.7.3.6 getUpperDisplacement()

```
QPointF FlywheelOperation::getUpperDisplacement ( )
```

FlywheelOperation::getUpperDisplacement Gets the upper displacement of the flywheel.

Returns

The upper displacement of the flywheel.

6.7.3.7 getVelocity()

```
float FlywheelOperation::getVelocity ( )
```

FlywheelOperation::getVelocity Gets the current velocity of the flywheel.

Returns

The current velocity of the flywheel.

6.7.3.8 setAcceleration()

```
void FlywheelOperation::setAcceleration (
             float acceleration )
```

FlywheelOperation::setAcceleration Sets the acceleration of the flywheel.

Parameters

acceleration	The acceleration to set the flywheel to.

6.7.3.9 setDefaults()

```
void FlywheelOperation::setDefaults ( )
```

FlywheelOperation::setDefaults Sets default values for member variables.

6.7.3.10 setInterface()

```
void FlywheelOperation::setInterface (
             CommonDeviceInterface * deviceInterface )
```

FlywheelOperation::setInterface Sets the device interface.

Parameters

CommonDeviceInterface∗	(Pointer to the interface to set member variable to)

6.7.3.11 setJerk()

```
void FlywheelOperation::setJerk (
             float jerkValue )
```

FlywheelOperation::setJerk Sets the jerk of the flywheel.

Parameters

jerk	The jerk to set the flywheel to.

6.7.3.12 setMotion()

```
void FlywheelOperation::setMotion (
             float velocity,
             float acceleration,
             float jerk )
```

FlywheelOperation::setMotion Sets all motion parameters for the flywheel.

Parameters

velocity	The velocity to set the flywheel to.
acceleration	The acceleration to set the flywheel to.
jerk	The jerk to set the flywheel to.

6.7.3.13 setVelocity()

```
void FlywheelOperation::setVelocity (
             float velocity )
```

FlywheelOperation::setVelocity Sets the velocity of the flywheel.

Parameters

velocity	The velocity to se tthe flywheel to.

6.7.3.14 sync()

```
void FlywheelOperation::sync ( )
```

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

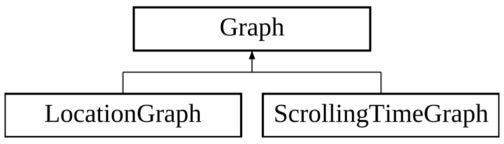
- FESS-GUI/flywheeloperation.h
- FESS-GUI/flywheeloperation.cpp

6.8 Graph Class Reference

The Graph class, base class for all graphs. Each graph contains a main plot (the large graph that is seen when a graph is selected) and a auxiliary plot (the smaller graph that is always visible on the right side).

```
#include <graph.h>
```

Inheritance diagram for Graph:



Public Member Functions

- Graph ()

    Graph::Graph Empty constructor for the base class. This should not be used.
- virtual QString maxDisplay ()
- virtual QString currentDisplay ()

Public Attributes

- QColor primaryColor
- QColor secondaryColor

Protected Attributes

- QCustomPlot ∗ mainPlot
- QCustomPlot ∗ auxPlot
- QString displayUnit

6.8.1 Detailed Description

The Graph class, base class for all graphs. Each graph contains a main plot (the large graph that is seen when a graph is selected) and a auxiliary plot (the smaller graph that is always visible on the right side).

6.8.2 Constructor & Destructor Documentation

6.8.2.1 Graph()

```
Graph::Graph ( )
```

Graph::Graph Empty constructor for the base class. This should not be used.

6.8.3 Member Function Documentation

6.8.3.1 currentDisplay()

```
virtual QString Graph::currentDisplay ( )  [inline], [virtual]
```

Reimplemented in LocationGraph, and ScrollingTimeGraph.

6.8.3.2 maxDisplay()

```
virtual QString Graph::maxDisplay ( )  [inline], [virtual]
```

Reimplemented in LocationGraph, and ScrollingTimeGraph.

6.8.4 Member Data Documentation

6.8.4.1 auxPlot

```
QCustomPlot * Graph::auxPlot  [protected]
```

6.8.4.2 displayUnit

```
QString Graph::displayUnit  [protected]
```

6.8.4.3 mainPlot

```
QCustomPlot* Graph::mainPlot  [protected]
```

6.8.4.4 primaryColor

```
QColor Graph::primaryColor
```

6.8.4.5 secondaryColor

`QColor Graph::secondaryColor`

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

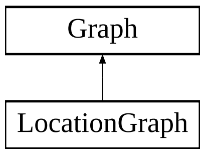
- FESS-GUI/graph.h
- FESS-GUI/graph.cpp

6.9 LocationGraph Class Reference

The LocationGraph class, which inherits from the Graph class. This represents a graph where the x and y axes represent location. This is a real-time graph, with no view of what happened in the past.

```
#include <graph.h>
```

Inheritance diagram for LocationGraph:



Public Member Functions

- LocationGraph (QCustomPlot ∗mainPlot, QCustomPlot ∗auxPlot, std::vector< QColor > colors, QString displayUnit, int numPoints)

    LocationGraph::LocationGraph Constructs a LocationGraph.
- void addData (std::vector< QPointF > points)

    LocationGraph::addData Adds a vector of points to the graph.
- QString maxDisplay () override

    LocationGraph::maxDisplay Returns a display of the maximum value seen by this graph so far, with units.
- QString currentDisplay () override

    LocationGraph::currentDisplay Returns a display of the most recent values seen by this graph, with units.

Public Attributes

- std::vector< QColor > colors

Additional Inherited Members

6.9.1 Detailed Description

The LocationGraph class, which inherits from the Graph class. This represents a graph where the x and y axes represent location. This is a real-time graph, with no view of what happened in the past.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 LocationGraph()

```
LocationGraph::LocationGraph (
             QCustomPlot * mainPlot,
             QCustomPlot * auxPlot,
             std::vector< QColor > colors,
             QString displayUnit,
             int numPoints )
```

LocationGraph::LocationGraph Constructs a LocationGraph.

Parameters

mainPlot	The main plot for this graph.
auxPlot	The auxiliary plot for this graph, which lives in the sidebar.
colors	A vector of the colors to be given to the points in this graph.
displayUnit	A string containing the name of the units to use in this graph's displays.
numPoints	The number of points to display on this graph.

6.9.3 Member Function Documentation

6.9.3.1 addData()

```
void LocationGraph::addData (
             std::vector< QPointF > points )
```

LocationGraph::addData Adds a vector of points to the graph.

Parameters

points	The points to add to this graph.

6.9.3.2 currentDisplay()

```
QString LocationGraph::currentDisplay ( )  [override], [virtual]
```

LocationGraph::currentDisplay Returns a display of the most recent values seen by this graph, with units.

Returns

A string containing the most recent values seen by this graph.

Reimplemented from Graph.

6.9.3.3 maxDisplay()

```
QString LocationGraph::maxDisplay ( )  [override], [virtual]
```

LocationGraph::maxDisplay Returns a display of the maximum value seen by this graph so far, with units.

Returns

A string containing the maximum values seen by this graph so far.

Reimplemented from Graph.

6.9.4 Member Data Documentation

6.9.4.1 colors

```
std::vector<QColor> LocationGraph::colors
```

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

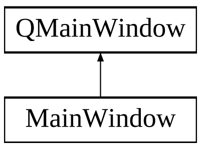
- FESS-GUI/graph.h
- FESS-GUI/graph.cpp

6.10 MainWindow Class Reference

The MainWindow class Comprises the bulk of the GUI.

```
#include <mainwindow.h>
```

Inheritance diagram for MainWindow:



Public Member Functions

- MainWindow (QWidget ∗parent=0)

  MainWindow::MainWindow Constructs the MainWindow object. Initializes all variables it needs to. By convention, we initialize values in the constructor rather than in the header/declaraton.
- ∼MainWindow ()

  MainWindow::∼MainWindow The destructor.

Public Attributes

- QMediaPlayer ∗ goplayer
- QMediaPlayer ∗ stopplayer
- RecordingOperation ∗ recording
- QTimer ∗ flywheelRefreshTimer
- QTimer ∗ graphRefreshTimer
- QTimer ∗ velocitySlopeTimer
- QTimer ∗ accelerationSlopeTimer
- bool playSounds
- bool isRecording
- bool isScaleLocked
- double graphRefreshRate
- double flywheelRefreshRate
- double targetVelocity
- double currentExpectedVelocity
- double targetAcceleration
- double currentExpectedAcceleration
- double currentExpectedJerk
- double yAxisDisplayBuffer
- int maximumVelocity
- int maximumAcceleration
- int sliderTickInterval
- QKeySequence eStopKey
- QElapsedTimer uptime
- QAction ∗ eStopShortcut
- Ui::MainWindow ∗ ui

6.10.1 Detailed Description

The MainWindow class Comprises the bulk of the GUI.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 MainWindow()

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

MainWindow::MainWindow Constructs the MainWindow object. Initializes all variables it needs to. By convention, we initialize values in the constructor rather than in the header/declaraton.

Parameters

parent	

6.10.2.2 ∼MainWindow()

```
MainWindow::∼MainWindow ( )
```

MainWindow::∼MainWindow The destructor.

6.10.3 Member Data Documentation

6.10.3.1 accelerationSlopeTimer

```
QTimer* MainWindow::accelerationSlopeTimer
```

6.10.3.2 currentExpectedAcceleration

```
double MainWindow::currentExpectedAcceleration
```

6.10.3.3 currentExpectedJerk

```
double MainWindow::currentExpectedJerk
```

6.10.3.4 currentExpectedVelocity

```
double MainWindow::currentExpectedVelocity
```

6.10.3.5 eStopKey

```
QKeySequence MainWindow::eStopKey
```

6.10.3.6 eStopShortcut

```
QAction* MainWindow::eStopShortcut
```

6.10.3.7 flywheelRefreshRate

```
double MainWindow::flywheelRefreshRate
```

6.10.3.8 flywheelRefreshTimer

```
QTimer* MainWindow::flywheelRefreshTimer
```

6.10.3.9 goplayer

```
QMediaPlayer* MainWindow::goplayer
```

6.10.3.10 graphRefreshRate

```
double MainWindow::graphRefreshRate
```

6.10.3.11 graphRefreshTimer

```
QTimer* MainWindow::graphRefreshTimer
```

6.10.3.12 isRecording

```
bool MainWindow::isRecording
```

6.10.3.13 isScaleLocked

```
bool MainWindow::isScaleLocked
```

6.10.3.14 maximumAcceleration

```
int MainWindow::maximumAcceleration
```

6.10.3.15 maximumVelocity

```
int MainWindow::maximumVelocity
```

6.10.3.16 playSounds

```
bool MainWindow::playSounds
```

6.10.3.17 recording

```
RecordingOperation* MainWindow::recording
```

6.10.3.18 sliderTickInterval

```
int MainWindow::sliderTickInterval
```

6.10.3.19 stopplayer

```
QMediaPlayer* MainWindow::stopplayer
```

6.10.3.20 targetAcceleration

```
double MainWindow::targetAcceleration
```

6.10.3.21 targetVelocity

```
double MainWindow::targetVelocity
```

6.10.3.22 ui

```
Ui::MainWindow* MainWindow::ui
```

6.10.3.23 uptime

```
QElapsedTimer MainWindow::uptime
```

6.10.3.24 velocitySlopeTimer

```
QTimer* MainWindow::velocitySlopeTimer
```

6.10.3.25 yAxisDisplayBuffer

```
double MainWindow::yAxisDisplayBuffer
```

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

- FESS-GUI/mainwindow.h
- FESS-GUI/mainwindow.cpp

6.11 RecordingOperation Class Reference

The RecordingOperation class Handles operations involving recording values to csv.

```
#include <recordingoperation.h>
```

Public Member Functions

- RecordingOperation ()

  RecordingOperation::RecordingOperation.
- void Start ()

  RecordingOperation::Start Starts the recording process. Creates a file "FlywheelOutput_{time}.csv", opens a filestream for it, and prints the heading row.
- void Stop ()

  RecordingOperation::Stop Stops the current recording process. Flushes the uffer and closes the filestream.
- void Record (double time, double velocity, double acceleration, double upperDispX, double upperDispY, double lowerDispX, double lowerDispY, double rotationalPosX, double rotationalPosY)

  RecordingOperation::Record Records the given values in a row.

6.11.1 Detailed Description

The RecordingOperation class Handles operations involving recording values to csv.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 RecordingOperation()

```
RecordingOperation::RecordingOperation ( )
```

RecordingOperation::RecordingOperation.

6.11.3 Member Function Documentation

6.11.3.1 Record()

```
void RecordingOperation::Record (
             double time,
             double velocity,
             double acceleration,
             double upperDispX,
             double upperDispY,
             double lowerDispX,
             double lowerDispY,
             double rotationalPosX,
             double rotationalPosY )
```

RecordingOperation::Record Records the given values in a row.

Parameters

time	Value to record in the time column.
velocity	Value to record in the velocity column.
acceleration	Value to record in the acceleration column.
upperDispX	Value to record in the upper displacement x column.
upperDispY	Value to record in the upper displacement y column.
lowerDispX	Value to record in the lower displacement x column.
lowerDispY	Value to record in the lower displacement y column.
rotationalPosX	Value to record in the rotational position x column.
rotationalPosY	Value to record in the rotational position y column.

6.11.3.2 Start()

```
void RecordingOperation::Start ( )
```

RecordingOperation::Start Starts the recording process. Creates a file "FlywheelOutput_{time}.csv", opens a filestream for it, and prints the heading row.

6.11.3.3 Stop()

```
void RecordingOperation::Stop ( )
```

RecordingOperation::Stop Stops the current recording process. Flushes the uffer and closes the filestream.

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

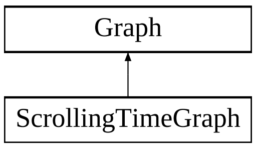
- FESS-GUI/recordingoperation.h
- FESS-GUI/recordingoperation.cpp

6.12 ScrollingTimeGraph Class Reference

The ScrollingTimeGraph class, which inherits from the Graph class. This represents a graph with time as the x axis, that "scrolls" with time, showing a sliding window of values. The y axis represents whatever value this graph displays.

```
#include <graph.h>
```

Inheritance diagram for ScrollingTimeGraph:



Public Member Functions

- ScrollingTimeGraph (QMainWindow ∗mainWindow, QCustomPlot ∗mainPlot, QCustomPlot ∗auxPlot, QColor primaryColor, QColor secondaryColor, QString displayUnit, int numDisplayValues)

  ScrollingTimeGraph::ScrollingTimeGraph Constructs the ScrollingTimeGraph.
- void addData (double time, double primaryData, double secondDaryData, int maxValue=-1)

  ScrollingTimeGraph::addData Adds two data points to both plots at the given time.
- void setFill (QColor fillColor)

  ScrollingTimeGraph::setFill Sets the fill between two lines.
- QString maxDisplay () override

  ScrollingTimeGraph::maxDisplay Returns a string showing the maximum values seen by this graph so far, with units.
- QString currentDisplay () override

  ScrollingTimeGraph::currentDisplay Returns a string showing the most recent values seen by this graph. with units.

Additional Inherited Members

6.12.1 Detailed Description

The ScrollingTimeGraph class, which inherits from the Graph class. This represents a graph with time as the x axis, that "scrolls" with time, showing a sliding window of values. The y axis represents whatever value this graph displays.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 ScrollingTimeGraph()

```
ScrollingTimeGraph::ScrollingTimeGraph (
            QMainWindow * mainWindow,
            QCustomPlot * mainPlot,
            QCustomPlot * auxPlot,
            QColor primaryColor,
            QColor secondaryColor,
            QString displayUnit,
            int numDisplayValues )
```

ScrollingTimeGraph::ScrollingTimeGraph Constructs the ScrollingTimeGraph.

Parameters

mainWindow	A pointer to the mainWindow, used for connecting ranges of the graph.
mainPlot	The main plot for this graph.
auxPlot	The auxiliary plot for this graph, which lives in the sidebar.
primaryColor	The color of the primary line of this graph.
secondaryColor	The color of the secondary line of this graph.
displayUnit	The units to use in displays.
numDisplayValues	The number of values to display in maxDisplay and currentDisplay.

6.12.3 Member Function Documentation

6.12.3.1 addData()

```
void ScrollingTimeGraph::addData (
            double time,
            double primaryData,
            double secondaryData,
            int maxValue = -1 )
```

ScrollingTimeGraph::addData Adds two data points to both plots at the given time.

Parameters

time	The x-axis value for the new points.
primaryData	The y-value of the point to add to the primary line.
secondaryData	The y-value of the point to add to the secondary line.
maxValue	The maximum expected y-value on the graph. If this is set to a negative number, this is disregarded, and the plots resize dynamically to accomodate real values.

6.12.3.2 currentDisplay()

```
QString ScrollingTimeGraph::currentDisplay ( )  [override], [virtual]
```

ScrollingTimeGraph::currentDisplay Returns a string showing the most recent values seen by this graph. with units.

Returns

The string which contains the most recent values seen by this graph.

Reimplemented from Graph.

6.12.3.3 maxDisplay()

```
QString ScrollingTimeGraph::maxDisplay ( )  [override], [virtual]
```

ScrollingTimeGraph::maxDisplay Returns a string showing the maximum values seen by this graph so far, with units.

Returns

The string which contains the maximum values seen by this graph.

Reimplemented from Graph.

6.12.3.4 setFill()

```
void ScrollingTimeGraph::setFill (
             QColor fillColor )
```

ScrollingTimeGraph::setFill Sets the fill between two lines.

Parameters

fillColor	The color to set the fill to.

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

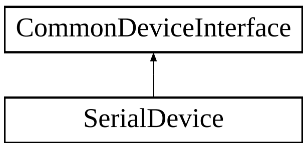
- FESS-GUI/graph.h
- FESS-GUI/graph.cpp

6.13 SerialDevice Class Reference

The Serial Class Provides access to serial interfaces.

`#include <serialdevice.h>`

Inheritance diagram for SerialDevice:



Public Member Functions

- SerialDevice ()

  SerialDevice::SerialDevice Creates a new serial instance and configures it with the default values.
- SerialDevice (QSerialPortInfo)

  SerialDevice::SerialDevice Creates a new serial instance and configures it with the default values.
- SerialDevice (QSerialPortInfo, int, int, int, int, int)
- SerialDevice (QString)

  SerialDevice::SerialDevice Creates a new serial instance and configures it with the default values.
- SerialDevice (QString, int, int, int, int, int)
- ∼SerialDevice ()

  SerialDevice::∼SerialDevice Stops transmissions and deletes the serial instance.
- void syncRX ()

  SerialDevice::syncRX Is the serial interface's implementation of syncRX. Same functionality as readRX.
- void syncTX ()

  SerialDevice::syncRX Is the serial interface's implementation of syncRX. Same functionality as sendTX.
- bool isReady ()

  SerialDevice::isReady Is the serial interface's implementation of isReady. Get the status of the serial interface.
- bool startDevice ()

  SerialDevice::startDevice Is the serial interface's implementation of startDevice. Starts the serial interface, clears its buffers and updates status.
- void stopDevice ()

  SerialDevice::stopDevice Is the serial interface's implementation of stopDevice. Stops the serial interface, empties the serial devices buffers, empties the Transmit Buffers and updates status.
- void setDefaults ()

  SerialDevice::stopDevice Is the serial interface's implementation of setDefaults. Sets the serial devices to the default values. (Rate=9600, Parity=None, Flow=None, Data=8, Stop=1)
- bool empty ()

  SerialDevice::empty Is the serial interface's implementation of empty.
- void flushRX ()

  SerialDevice::flushRX Is the serial interface's implementation of flushRX. Empties the RX Transmit Buffer.
- void flushTX ()

  SerialDevice::flushTX Is the serial interface's implementation of flushTX. Empties the TX Transmit Buffer.
- void pushByte (FlyByte)

  SerialDevice::popByte Is the serial interface's implementation of pushByte.
- void pushPacket (FlyPacket)

  SerialDevice::popByte Is the serial interface's implementation of pushPacket.
- FlyByte popByte ()

  SerialDevice::popByte Is the serial interface's implementation of popByte.
- FlyPacket popPacket ()

  SerialDevice::popByte Is the serial interface's implementation of popPacket.
- QString name ()

  SerialDevice::name Is the serial interface's implementation of name.
- void setDevice (int)
- void setBaudRate (int)

  SerialDevice::setBaudRate Changes the serial device's baud rate.
- void setParity (int)

  SerialDevice::setParity Changes the serial device's parity.
- void setFlowControl (int)

  SerialDevice::setFlowControl Changes the serial device's flow control.
- void setDataBits (int)

  SerialDevice::setDataBits Changes the serial device's number of data bits.

- void setStopBits (int)

  SerialDevice::setStopBits Changes the serial device's number of stop bits.
- void setPort (QSerialPortInfo)

  SerialDevice::setPort Sets the serial port.

6.13.1 Detailed Description

The Serial Class Provides access to serial interfaces.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 SerialDevice() [1/5]

```
SerialDevice::SerialDevice ( )
```

SerialDevice::SerialDevice Creates a new serial instance and configures it with the default values.

6.13.2.2 SerialDevice() [2/5]

```
SerialDevice::SerialDevice (
             QSerialPortInfo port )
```

SerialDevice::SerialDevice Creates a new serial instance and configures it with the default values.

Parameters

QSerialPortInfo	(Serial Port Instance)

6.13.2.3 SerialDevice() [3/5]

```
SerialDevice::SerialDevice (
             QSerialPortInfo ,
             int ,
             int ,
             int ,
             int ,
             int  )
```

6.13.2.4 SerialDevice() [4/5]

```
SerialDevice::SerialDevice (
             QString path )
```

SerialDevice::SerialDevice Creates a new serial instance and configures it with the default values.

Parameters

QString	(Windows Example: COMM1, ∗INX Example: /dev/ttyUSB0)

6.13.2.5 SerialDevice() [5/5]

```
SerialDevice::SerialDevice (
            QString ,
            int ,
            int ,
            int ,
            int ,
            int  )
```

6.13.2.6 ∼SerialDevice()

```
SerialDevice::∼SerialDevice ( )
```

SerialDevice::∼SerialDevice Stops transmissions and deletes the serial instance.

6.13.3 Member Function Documentation

6.13.3.1 empty()

```
bool SerialDevice::empty ( )  [virtual]
```

SerialDevice::empty Is the serial interface's implementation of empty.

Returns

bool (true=Empty, false=Not Empty)

Implements CommonDeviceInterface.

6.13.3.2 flushRX()

```
void SerialDevice::flushRX ( )  [virtual]
```

SerialDevice::flushRX Is the serial interface's implementation of flushRX. Empties the RX Transmit Buffer.

Implements CommonDeviceInterface.

6.13.3.3 flushTX()

```
void SerialDevice::flushTX ( )  [virtual]
```

SerialDevice::flushTX Is the serial interface's implementation of flushTX. Empties the TX Transmit Buffer.

Implements CommonDeviceInterface.

6.13.3.4 isReady()

```
bool SerialDevice::isReady ( )  [virtual]
```

SerialDevice::isReady Is the serial interface's implementation of isReady. Get the status of the serial interface.

Returns

bool (true=Ready, false=Not Ready)

Implements CommonDeviceInterface.

6.13.3.5 name()

```
QString SerialDevice::name ( )  [virtual]
```

SerialDevice::name Is the serial interface's implementation of name.

Returns

Qstring (Serial Device Port)

Implements CommonDeviceInterface.

6.13.3.6 popByte()

```
FlyByte SerialDevice::popByte ( )  [virtual]
```

SerialDevice::popByte Is the serial interface's implementation of popByte.

Returns

FlyByte (From the RX Transmit Buffer).

Implements CommonDeviceInterface.

6.13.3.7 popPacket()

```
FlyPacket SerialDevice::popPacket ( )  [virtual]
```

SerialDevice::popByte Is the serial interface's implementation of popPacket.

Returns

FlyPacket (From the RX Transmit Buffer).

Implements CommonDeviceInterface.

6.13.3.8 pushByte()

```
void SerialDevice::pushByte (
            FlyByte dataByte )  [virtual]
```

SerialDevice::popByte Is the serial interface's implementation of pushByte.

Parameters

FlyByte	(Adds it to the TX Transmit Buffer).

Implements CommonDeviceInterface.

6.13.3.9 pushPacket()

```
void SerialDevice::pushPacket (
             FlyPacket dataPacket )  [virtual]
```

SerialDevice::popByte Is the serial interface's implementation of pushPacket.

Parameters

FlyPacket	(Adds to the TX Transmit Buffer).

Implements CommonDeviceInterface.

6.13.3.10 setBaudRate()

```
void SerialDevice::setBaudRate (
             int rate )
```

SerialDevice::setBaudRate Changes the serial device's baud rate.

Parameters

int	(0-9999999999)

6.13.3.11 setDataBits()

```
void SerialDevice::setDataBits (
             int bits )
```

SerialDevice::setDataBits Changes the serial device's number of data bits.

Parameters

int	(5-8)

6.13.3.12 setDefaults()

```
void SerialDevice::setDefaults ( )  [virtual]
```

SerialDevice::stopDevice Is the serial interface's implementation of setDefaults. Sets the serial devices to the default values. (Rate=9600, Parity=None, Flow=None, Data=8, Stop=1)

Implements CommonDeviceInterface.

6.13.3.13 setDevice()

```
void SerialDevice::setDevice (
             int  )
```

6.13.3.14 setFlowControl()

```
void SerialDevice::setFlowControl (
             int flow )
```

SerialDevice::setFlowControl Changes the serial device's flow control.

Parameters

int	(0=NONE, 1=HW, 2=SW)

6.13.3.15 setParity()

```
void SerialDevice::setParity (
             int pari )
```

SerialDevice::setParity Changes the serial device's parity.

Parameters

int	(0=NO, 1=ODD, 2=EVEN)

6.13.3.16 setPort()

```
void SerialDevice::setPort (
             QSerialPortInfo port )
```

SerialDevice::setPort Sets the serial port.

Parameters

QSerialPortInfo	Instance

6.13.3.17 setStopBits()

```
void SerialDevice::setStopBits (
            int bits )
```

SerialDevice::setStopBits Changes the serial device's number of stop bits.

Parameters

int	(1=1, 2=1.5, 3=2)

6.13.3.18 startDevice()

```
bool SerialDevice::startDevice ( )  [virtual]
```

SerialDevice::startDevice Is the serial interface's implementation of startDevice. Starts the serial interface, clears its buffers and updates status.

Returns

bool (true=Started, false=Not Ready)

Implements CommonDeviceInterface.

6.13.3.19 stopDevice()

```
void SerialDevice::stopDevice ( )  [virtual]
```

SerialDevice::stopDevice Is the serial interface's implementation of stopDevice. Stops the serial interface, empties the serial devices buffers, empties the Transmit Buffers and updates status.

Implements CommonDeviceInterface.

6.13.3.20 syncRX()

```
void SerialDevice::syncRX ( )  [virtual]
```

SerialDevice::syncRX Is the serial interface's implementation of syncRX. Same functionality as readRX.

Implements CommonDeviceInterface.

6.13.3.21 syncTX()

```
void SerialDevice::syncTX ( )  [virtual]
```

SerialDevice::syncRX Is the serial interface's implementation of syncRX. Same functionality as sendTX.

Implements CommonDeviceInterface.

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

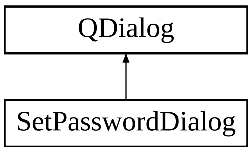
- FESS-GUI/serialdevice.h
- FESS-GUI/serialdevice.cpp

6.14 SetPasswordDialog Class Reference

The SetPasswordDialog class Represents a dialog for setting and resetting passwords.

```
#include <setpassworddialog.h>
```

Inheritance diagram for SetPasswordDialog:



Public Member Functions

- SetPasswordDialog (QWidget ∗parent=0)

  SetPasswordDialog::SetPasswordDialog Constructs the object.
- ∼SetPasswordDialog ()

  SetPasswordDialog::∼SetPasswordDialog Destructs the object.

6.14.1 Detailed Description

The SetPasswordDialog class Represents a dialog for setting and resetting passwords.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 SetPasswordDialog()

```
SetPasswordDialog::SetPasswordDialog (
            QWidget * parent = 0 )   [explicit]
```

SetPasswordDialog::SetPasswordDialog Constructs the object.

Parameters

parent	The parent widget (should be the mainWindow).

6.14.2.2 ∼SetPasswordDialog()

```
SetPasswordDialog::∼SetPasswordDialog ( )
```

SetPasswordDialog::∼SetPasswordDialog Destructs the object.

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

- FESS-GUI/setpassworddialog.h
- FESS-GUI/setpassworddialog.cpp

6.15 TransmitBuffer Class Reference

The Transmit Buffer Class Provides queueing for the communication packets and raw output of packets for interfaces.

`#include <transmitbuffer.h>`

Public Member Functions

- TransmitBuffer ()

    TransmitBuffer::TransmitBuffer Provided if needed in the future, currently no functionality.
- ∼TransmitBuffer ()

    TransmitBuffer::∼TransmitBuffer Provided if needed in the future, currently no functionality.
- void pushByte (FlyByte)

    TransmitBuffer::pushByte Takes bytes until there are enough to build a packet. It converts the bytes to a packet and adds the packet to the packet queue.
- void pushPacket (FlyPacket)

    TransmitBuffer::pushPacket Adds the packets to the packet queue.
- FlyByte popByte ()

    TransmitBuffer::popByte Get the Byte at the front of the output byte queue created from a packet.
- FlyPacket popPacket ()

    TransmitBuffer::popPacket Adds the packets to the packet queue.
- void flush ()

    TransmitBuffer::flush Empties all internal queues.
- bool bytesAvailable ()

    TransmitBuffer::bytesAvailable Checks if the byte and packet queues are empty.
- bool packetsAvailable ()

    TransmitBuffer::packetsAvailable Checks if the packet queue is empty.

6.15.1 Detailed Description

The Transmit Buffer Class Provides queueing for the communication packets and raw output of packets for interfaces.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 TransmitBuffer()

```
TransmitBuffer::TransmitBuffer ( )
```

TransmitBuffer::TransmitBuffer Provided if needed in the future, currently no functionality.

6.15.2.2 ∼TransmitBuffer()

```
TransmitBuffer::∼TransmitBuffer ( )
```

TransmitBuffer::∼TransmitBuffer Provided if needed in the future, currently no functionality.

6.15.3 Member Function Documentation

6.15.3.1 bytesAvailable()

```
bool TransmitBuffer::bytesAvailable ( )
```

TransmitBuffer::bytesAvailable Checks if the byte and packet queues are empty.

Returns

bool (true=yes | false=no)

6.15.3.2 flush()

```
void TransmitBuffer::flush ( )
```

TransmitBuffer::flush Empties all internal queues.

6.15.3.3 packetsAvailable()

```
bool TransmitBuffer::packetsAvailable ( )
```

TransmitBuffer::packetsAvailable Checks if the packet queue is empty.

Returns

bool (true=yes | false=no)

6.15.3.4 popByte()

```
FlyByte TransmitBuffer::popByte ( )
```

TransmitBuffer::popByte Get the Byte at the front of the output byte queue created from a packet.

Returns

FlyByte (First byte in the output queue)

6.15.3.5 popPacket()

```
FlyPacket TransmitBuffer::popPacket ( )
```

TransmitBuffer::popPacket Adds the packets to the packet queue.

Returns

FlyPacket (Gets a packet from the front of the packet queue)

6.15.3.6 pushByte()

```
void TransmitBuffer::pushByte (
            FlyByte incomingByte )
```

TransmitBuffer::pushByte Takes bytes until there are enough to build a packet. It converts the bytes to a packet and adds the packet to the packet queue.

Parameters

FlyByte	(Added to the incoming queue)

6.15.3.7 pushPacket()

```
void TransmitBuffer::pushPacket (
            FlyPacket incomingPacket )
```

TransmitBuffer::pushPacket Adds the packets to the packet queue.

Parameters

FlyPacket	(Added to the Packet Queue)

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

- FESS-GUI/transmitbuffer.h
- FESS-GUI/transmitbuffer.cpp

Chapter 7

File Documentation

7.1 FESS-GUI/commondeviceinterface.h File Reference

```
#include <QString>
#include "flypacket.h"
```

Classes

- class CommonDeviceInterface

  The Abstract Base Class: Common Device Interface A generic template for interfaces.

7.2 FESS-GUI/commoninterfacemanager.cpp File Reference

```
#include "commoninterfacemanager.h"
```

7.3 FESS-GUI/commoninterfacemanager.h File Reference

```
#include "commondeviceinterface.h"
```

Classes

- class CommonInterfaceManager

  CommonInterfaceManager Class Manages interfaces set in the interface selector menu.

7.4 FESS-GUI/commoninterfaceselector.cpp File Reference

```
#include "commoninterfaceselector.h"
```

7.5 FESS-GUI/commoninterfaceselector.h File Reference

```
#include <QDialog>
#include <QErrorMessage>
#include <QSerialPortInfo>
#include "demodevice.h"
#include "serialdevice.h"
#include "commoninterfacemanager.h"
#include "ui_commoninterfaceselector.h"
```

Classes

- class CommonInterfaceSelector

Namespaces

- Ui

  CommonInterfaceSelector class Handles the selection of interfaces.

7.6 FESS-GUI/conversions.cpp File Reference

```
#include "conversions.h"
```

Functions

- float byteArrayToFloat (FlyByte ∗buffer)

  byteArrayToFloat Interprets and copies a byte a array into a float.
- void floatToByteArray (FlyByte ∗buffer, float ∗val)

  floatToByteArray Interprets and copies a float into a byte array.
- int byteArrayToInt (FlyByte ∗buffer)

  byteArrayToInt Interprets and copies a byte array into an int.
- void intToByteArray (FlyByte ∗buffer, int ∗val)

  intToByteArray Interprets and copies an int into a byte array.
- void zeroArray (void ∗target, size_t size)

  zeroArray fills an array with zeros.
- double radsPerSecondToRPM (double rads)

  radsPerSecondToRPM Convertes a value in radians per second to rotations per minute.
- double RPMtoRadsPerSecond (double RPM)

  RPMtoRadsPerSecond Converts a value in rotations per miute to radians per second.
- float derivative (float value, float prev)

  derivative Returns the difference between two values.
- float refreshRateToMS (int rate)

  refreshRateToMS Converts a rate in Hz to its corresponding interval in milliseconds.

7.6.1 Function Documentation

7.6.1.1 byteArrayToFloat()

```
float byteArrayToFloat (
            FlyByte ∗ buffer )
```

byteArrayToFloat Interprets and copies a byte a array into a float.

Parameters

buffer	The source byte array.

Returns

The array converted to a float.

7.6.1.2 byteArrayToInt()

```
int byteArrayToInt (
            FlyByte * buffer )
```

byteArrayToInt Interprets and copies a byte array into an int.

Parameters

buffer	The source byte array.

Returns

The array converted to an int.

7.6.1.3 derivative()

```
float derivative (
            float value,
            float prev )
```

derivative Returns the difference between two values.

Parameters

value	The given value.
prev	The previous value.

Returns

The difference between value and prev.

7.6.1.4 floatToByteArray()

```
void floatToByteArray (
            FlyByte * buffer,
            float * val )
```

floatToByteArray Interprets and copies a float into a byte array.

Parameters

buffer	The destination byte array.
val	The source float.

7.6.1.5 intToByteArray()

```
void intToByteArray (
            FlyByte * buffer,
            int * val )
```

intToByteArray Interprets and copies an int into a byte array.

Parameters

buffer	The destination byte array.
val	The source int.

7.6.1.6 radsPerSecondToRPM()

```
double radsPerSecondToRPM (
            double rads )
```

radsPerSecondToRPM Convertes a value in radians per second to rotations per minute.

Parameters

rads	The source value in radians per second.

Returns

The value converted to rotations per minute.

7.6.1.7 refreshRateToMS()

```
float refreshRateToMS (
            int rate )
```

refreshRateToMS Converts a rate in Hz to its corresponding interval in milliseconds.

Parameters

rate	The source value in Hz.

Returns

The corresponding interval in milliseconds.

7.6.1.8 RPMtoRadsPerSecond()

```
double RPMtoRadsPerSecond (
             double RPM )
```

RPMtoRadsPerSecond Converts a value in rotations per miute to radians per second.

Parameters

RPM	The source value in rotations per minute.

Returns

The value converted to radians per second.

7.6.1.9 zeroArray()

```
void zeroArray (
             void * target,
             size_t size )
```

zeroArray fills an array with zeros.

Parameters

target	The destination array.
size	The size of the array.

7.7 FESS-GUI/conversions.h File Reference

```
#include <cstring>
```

Macros

- #define TAU 6.283185307179586476925286766559005768394

  The conversions library, containing all conversions our program uses.

Typedefs

- typedef unsigned char FlyByte

Functions

- float byteArrayToFloat (FlyByte ∗)

  byteArrayToFloat Interprets and copies a byte a array into a float.
- void floatToByteArray (FlyByte ∗, float ∗)

  floatToByteArray Interprets and copies a float into a byte array.
- int byteArrayToInt (FlyByte ∗)

  byteArrayToInt Interprets and copies a byte array into an int.
- void intToByteArray (FlyByte ∗, int ∗)

  intToByteArray Interprets and copies an int into a byte array.
- void zeroArray (void ∗, size_t)

  zeroArray fills an array with zeros.
- double radsPerSecondToRPM (double)

  radsPerSecondToRPM Convertes a value in radians per second to rotations per minute.
- double RPMtoRadsPerSecond (double)

  RPMtoRadsPerSecond Converts a value in rotations per miute to radians per second.
- float derivative (float, float)

  derivative Returns the difference between two values.
- float refreshRateToMS (int)

  refreshRateToMS Converts a rate in Hz to its corresponding interval in milliseconds.

7.7.1 Macro Definition Documentation

7.7.1.1 TAU

```
#define TAU 6.283185307179586476925286766559005768394
```

The conversions library, containing all conversions our program uses.

7.7.2 Typedef Documentation

7.7.2.1 FlyByte

```
typedef unsigned char FlyByte
```

7.7.3 Function Documentation

7.7.3.1 byteArrayToFloat()

```
float byteArrayToFloat (
            FlyByte ∗ buffer )
```

byteArrayToFloat Interprets and copies a byte a array into a float.

Parameters

buffer	The source byte array.

Returns

The array converted to a float.

7.7.3.2 byteArrayToInt()

```
int byteArrayToInt (
            FlyByte * buffer )
```

byteArrayToInt Interprets and copies a byte array into an int.

Parameters

buffer	The source byte array.

Returns

The array converted to an int.

7.7.3.3 derivative()

```
float derivative (
            float value,
            float prev )
```

derivative Returns the difference between two values.

Parameters

value	The given value.
prev	The previous value.

Returns

The difference between value and prev.

7.7.3.4 floatToByteArray()

```
void floatToByteArray (
            FlyByte * buffer,
            float * val )
```

floatToByteArray Interprets and copies a float into a byte array.

Parameters

buffer	The destination byte array.
val	The source float.

7.7.3.5 intToByteArray()

```
void intToByteArray (
            FlyByte * buffer,
            int * val )
```

intToByteArray Interprets and copies an int into a byte array.

Parameters

buffer	The destination byte array.
val	The source int.

7.7.3.6 radsPerSecondToRPM()

```
double radsPerSecondToRPM (
            double rads )
```

radsPerSecondToRPM Convertes a value in radians per second to rotations per minute.

Parameters

rads	The source value in radians per second.

Returns

The value converted to rotations per minute.

7.7.3.7 refreshRateToMS()

```
float refreshRateToMS (
            int rate )
```

refreshRateToMS Converts a rate in Hz to its corresponding interval in milliseconds.

Parameters

rate	The source value in Hz.

Returns

The corresponding interval in milliseconds.

7.7.3.8 RPMtoRadsPerSecond()

```
double RPMtoRadsPerSecond (
            double RPM )
```

RPMtoRadsPerSecond Converts a value in rotations per miute to radians per second.

Parameters

RPM	The source value in rotations per minute.

Returns

The value converted to radians per second.

7.7.3.9 zeroArray()

```
void zeroArray (
             void * target,
             size_t size )
```

zeroArray fills an array with zeros.

Parameters

target	The destination array.
size	The size of the array.

7.8 FESS-GUI/demodevice.cpp File Reference

```
#include <QtGui>
#include <QString>
#include <cmath>
#include "conversions.h"
#include "demodevice.h"
```

7.9 FESS-GUI/demodevice.h File Reference

```
#include <string>
#include "flypacket.h"
#include "transmitbuffer.h"
#include "commondeviceinterface.h"
```

Classes

- class DemoDevice

Macros

- #define RANDOM 0

    The Demo Class Simulates flywheel activity.
- #define STOP 1
- #define COMMAND 2

7.9.1 Macro Definition Documentation

7.9.1.1 COMMAND

```
#define COMMAND 2
```

7.9.1.2 RANDOM

```
#define RANDOM 0
```

The Demo Class Simulates flywheel activity.

7.9.1.3 STOP

```
#define STOP 1
```

7.10 FESS-GUI/flypacket.cpp File Reference

```
#include "flypacket.h"
```

7.11 FESS-GUI/flypacket.h File Reference

```
#include "conversions.h"
```

Classes

- class FlyPacket

Macros

- #define ICM_START 0b00000001

  FlyPacket Class Provide a packet structure to transport data with approirate headers and footers.
- #define ICM_STOP 0b00000010
- #define ICM_EMERGENCY_STOP 0b00000011
- #define ICM_SET_VELOCITY 0b00000100
- #define ICM_SET_ACCELERATION 0b00000101
- #define ICM_SET_JERK 0b00000110
- #define CCM_START 0b10000001
- #define CCM_STOP 0b10000010
- #define CCM_EMERGENCY_STOP 0b10000011
- #define CCM_SET_VELOCITY 0b10000100
- #define CCM_SET_ACCELERATION 0b10000101
- #define CCM_SET_JERK 0b10000110

- #define ICC_ERROR 0b00100001
- #define CCC_ERROR 0b10100001
- #define IDM_SEND_NULL 0b00000000
- #define IDM_SEND_VELOCITY 0b01000001
- #define IDM_SEND_ACCELERATION 0b01000010
- #define IDM_SEND_JERK 0b01000011
- #define IDM_SEND_LOWER_DISPLACEMENT_X 0b01000100
- #define IDM_SEND_LOWER_DISPLACEMENT_Y 0b01000101
- #define IDM_SEND_UPPER_DISPLACEMENT_X 0b01000110
- #define IDM_SEND_UPPER_DISPLACEMENT_Y 0b01000111
- #define IDM_SEND_ROTATIONAL_POSITION_X 0b01001000
- #define IDM_SEND_ROTATIONAL_POSITION_Y 0b01001001
- #define CDM_SEND_NULL 0b10000000
- #define CDM_SEND_VELOCITY 0b11000001
- #define CDM_SEND_ACCELERATION 0b11000010
- #define CDM_SEND_JERK 0b11000011
- #define CDM_SEND_LOWER_DISPLACEMENT_X 0b11000100
- #define CDM_SEND_LOWER_DISPLACEMENT_Y 0b11000101
- #define CDM_SEND_UPPER_DISPLACEMENT_X 0b11000110
- #define CDM_SEND_UPPER_DISPLACEMENT_Y 0b11000111
- #define CDM_SEND_ROTATIONAL_POSITION_X 0b11001000
- #define CDM_SEND_ROTATIONAL_POSITION_Y 0b11001001
- #define IDM_CMD_DIFFERENCE 0b10000000
- #define HEADER_SIZE 1
- #define FOOTER_SIZE 1
- #define MAX_PAYLOAD 4
- #define PACKET_BEGINNING 0
- #define PACKET_END HEADER_SIZE + MAX_PAYLOAD
- #define DATA_BEGINNING HEADER_SIZE
- #define DATA_END PACKET_END - FOOTER_SIZE
- #define PACKET_SIZE PACKET_END + FOOTER_SIZE

7.11.1 Macro Definition Documentation

7.11.1.1 CCC_ERROR

```
#define CCC_ERROR 0b10100001
```

7.11.1.2 CCM_EMERGENCY_STOP

```
#define CCM_EMERGENCY_STOP 0b10000011
```

7.11.1.3 CCM_SET_ACCELERATION

```
#define CCM_SET_ACCELERATION 0b10000101
```

7.11.1.4 CCM_SET_JERK

```
#define CCM_SET_JERK 0b10000110
```

7.11.1.5 CCM_SET_VELOCITY

```
#define CCM_SET_VELOCITY 0b10000100
```

7.11.1.6 CCM_START

```
#define CCM_START 0b10000001
```

7.11.1.7 CCM_STOP

```
#define CCM_STOP 0b10000010
```

7.11.1.8 CDM_SEND_ACCELERATION

```
#define CDM_SEND_ACCELERATION 0b11000010
```

7.11.1.9 CDM_SEND_JERK

```
#define CDM_SEND_JERK 0b11000011
```

7.11.1.10 CDM_SEND_LOWER_DISPLACEMENT_X

```
#define CDM_SEND_LOWER_DISPLACEMENT_X 0b11000100
```

7.11.1.11 CDM_SEND_LOWER_DISPLACEMENT_Y

```
#define CDM_SEND_LOWER_DISPLACEMENT_Y 0b11000101
```

7.11.1.12 CDM_SEND_NULL

```
#define CDM_SEND_NULL 0b10000000
```

7.11.1.13 CDM_SEND_ROTATIONAL_POSITION_X

```
#define CDM_SEND_ROTATIONAL_POSITION_X 0b11001000
```

7.11.1.14 CDM_SEND_ROTATIONAL_POSITION_Y

```
#define CDM_SEND_ROTATIONAL_POSITION_Y 0b11001001
```

7.11.1.15 CDM_SEND_UPPER_DISPLACEMENT_X

```
#define CDM_SEND_UPPER_DISPLACEMENT_X 0b11000110
```

7.11.1.16 CDM_SEND_UPPER_DISPLACEMENT_Y

```
#define CDM_SEND_UPPER_DISPLACEMENT_Y 0b11000111
```

7.11.1.17 CDM_SEND_VELOCITY

```
#define CDM_SEND_VELOCITY 0b11000001
```

7.11.1.18 DATA_BEGINNING

```
#define DATA_BEGINNING HEADER_SIZE
```

7.11.1.19 DATA_END

```
#define DATA_END PACKET_END - FOOTER_SIZE
```

7.11.1.20 FOOTER_SIZE

```
#define FOOTER_SIZE 1
```

7.11.1.21 HEADER_SIZE

```
#define HEADER_SIZE 1
```

7.11.1.22 ICC_ERROR

```
#define ICC_ERROR 0b00100001
```

7.11.1.23 ICM_EMERGENCY_STOP

```
#define ICM_EMERGENCY_STOP 0b00000011
```

7.11.1.24 ICM_SET_ACCELERATION

```
#define ICM_SET_ACCELERATION 0b00000101
```

7.11.1.25 ICM_SET_JERK

```
#define ICM_SET_JERK 0b00000110
```

7.11.1.26 ICM_SET_VELOCITY

```
#define ICM_SET_VELOCITY 0b00000100
```

7.11.1.27 ICM_START

```
#define ICM_START 0b00000001
```

FlyPacket Class Provide a packet structure to transport data with approirate headers and footers.

7.11.1.28 ICM_STOP

```
#define ICM_STOP 0b00000010
```

7.11.1.29 IDM_CMD_DIFFERENCE

```
#define IDM_CMD_DIFFERENCE 0b10000000
```

7.11.1.30 IDM_SEND_ACCELERATION

```
#define IDM_SEND_ACCELERATION 0b01000010
```

7.11.1.31 IDM_SEND_JERK

```
#define IDM_SEND_JERK 0b01000011
```

7.11.1.32 IDM_SEND_LOWER_DISPLACEMENT_X

```
#define IDM_SEND_LOWER_DISPLACEMENT_X 0b01000100
```

7.11.1.33 IDM_SEND_LOWER_DISPLACEMENT_Y

```
#define IDM_SEND_LOWER_DISPLACEMENT_Y 0b01000101
```

7.11.1.34 IDM_SEND_NULL

```
#define IDM_SEND_NULL 0b00000000
```

7.11.1.35 IDM_SEND_ROTATIONAL_POSITION_X

```
#define IDM_SEND_ROTATIONAL_POSITION_X 0b01001000
```

7.11.1.36 IDM_SEND_ROTATIONAL_POSITION_Y

```
#define IDM_SEND_ROTATIONAL_POSITION_Y 0b01001001
```

7.11.1.37 IDM_SEND_UPPER_DISPLACEMENT_X

```
#define IDM_SEND_UPPER_DISPLACEMENT_X 0b01000110
```

7.11.1.38 IDM_SEND_UPPER_DISPLACEMENT_Y

```
#define IDM_SEND_UPPER_DISPLACEMENT_Y 0b01000111
```

7.11.1.39 IDM_SEND_VELOCITY

```
#define IDM_SEND_VELOCITY 0b01000001
```

7.11.1.40 MAX_PAYLOAD

```
#define MAX_PAYLOAD 4
```

7.11.1.41 PACKET_BEGINNING

```
#define PACKET_BEGINNING 0
```

7.11.1.42 PACKET_END

```
#define PACKET_END HEADER_SIZE + MAX_PAYLOAD
```

7.11.1.43 PACKET_SIZE

```
#define PACKET_SIZE PACKET_END + FOOTER_SIZE
```

7.12 FESS-GUI/flyqueue.cpp File Reference

```
#include "flyqueue.h"
```

7.13 FESS-GUI/flyqueue.h File Reference

```
#include <deque>
#include <flypacket.h>
```

Classes

- class FlyQueue

  FlyQueue Class A wrapper to allow for reuse on the microcontroller.

7.14 FESS-GUI/flywheeloperation.cpp File Reference

```
#include "flypacket.h"
#include "flywheeloperation.h"
```

7.15 FESS-GUI/flywheeloperation.h File Reference

```
#include <QPointF>
#include <queue>
#include "commondeviceinterface.h"
```

Classes

- class FlywheelOperation

  The FlywheelOperation class handles all operations involving the flywheel.

7.16 FESS-GUI/graph.cpp File Reference

```
#include "graph.h"
```

7.17 FESS-GUI/graph.h File Reference

```
#include "qcustomplot.h"
#include <vector>
#include <QString>
#include <QPointF>
```

Classes

- class Graph

  The Graph class, base class for all graphs. Each graph contains a main plot (the large graph that is seen when a graph is selected) and a auxiliary plot (the smaller graph that is always visible on the right side).
- class ScrollingTimeGraph

  The ScrollingTimeGraph class, which inherits from the Graph class. This represents a graph with time as the x axis, that "scrolls" with time, showing a sliding window of values. The y axis represents whatever value this graph displays.
- class LocationGraph

  The LocationGraph class, which inherits from the Graph class. This represents a graph where the x and y axes represent location. This is a real-time graph, with no view of what happened in the past.

7.18 FESS-GUI/main.cpp File Reference

```
#include <QApplication>
#include <QPushButton>
#include <QSlider>
#include <QHBoxLayout>
#include <QSpinBox>
#include "mainwindow.h"
```

Functions

- int main (int argc, char ∗argv[ ])

7.18.1 Function Documentation

7.18.1.1 main()

```
int main (
          int argc,
          char ∗ argv[ ] )
```

7.19 FESS-GUI/mainwindow.cpp File Reference

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "conversions.h"
#include "setpassworddialog.h"
#include "flywheeloperation.h"
#include "commoninterfacemanager.h"
#include "commoninterfaceselector.h"
#include <ctime>
#include <QTime>
#include <QKeyEvent>
#include "qmath.h"
#include <vector>
```

7.20 FESS-GUI/mainwindow.h File Reference

```
#include <QMainWindow>
#include <QMediaPlayer>
#include <QTimer>
#include <qcustomplot.h>
#include "graph.h"
#include "flywheeloperation.h"
#include "recordingoperation.h"
#include "commoninterfacemanager.h"
```

Classes

- class MainWindow

  The MainWindow class Comprises the bulk of the GUI.

Namespaces

- Ui

  CommonInterfaceSelector class Handles the selection of interfaces.

7.21 FESS-GUI/recordingoperation.cpp File Reference

```
#include <ctime>
#include <iomanip>
#include <sstream>
#include "recordingoperation.h"
```

7.22 FESS-GUI/recordingoperation.h File Reference

```
#include <fstream>
```

Classes

- class RecordingOperation

  The RecordingOperation class Handles operations involving recording values to csv.

7.23 FESS-GUI/serialdevice.cpp File Reference

```
#include "serialdevice.h"
```

7.24 FESS-GUI/serialdevice.h File Reference

```
#include <QString>
#include <QSerialPort>
#include <QSerialPortInfo>
#include "transmitbuffer.h"
#include "commondeviceinterface.h"
```

Classes

- class SerialDevice

  The Serial Class Provides access to serial interfaces.

7.25 FESS-GUI/setpassworddialog.cpp File Reference

```
#include <QSettings>
#include <QString>
#include <QDebug>
#include <QCryptographicHash>
#include <QTime>
#include "setpassworddialog.h"
#include "ui_setpassworddialog.h"
```

Functions

- bool passwordMatches (QString prov)

  passwordMatches Checks if a provided password matches the set password.
- QString GetRandomString ()

  GetRandomString Generates a random string of letters and numbers of size 16 - 32.

7.25.1 Function Documentation

7.25.1.1 GetRandomString()

```
QString GetRandomString ( )
```

GetRandomString Generates a random string of letters and numbers of size 16 - 32.

Returns

The generated random string.

7.25.1.2 passwordMatches()

```
bool passwordMatches (
            QString prov )
```

passwordMatches Checks if a provided password matches the set password.

Parameters

prov	A provided password that will be checked.

Returns

True if the provided password matches the set password. False otherwise.

7.26 FESS-GUI/setpassworddialog.h File Reference

```
#include <QDialog>
```

Classes

- class SetPasswordDialog

  The SetPasswordDialog class Represents a dialog for setting and resetting passwords.

Namespaces

- Ui

  CommonInterfaceSelector class Handles the selection of interfaces.

Functions

- bool passwordMatches (QString)

  passwordMatches Checks if a provided password matches the set password.
- QString GetRandomString ()

  GetRandomString Generates a random string of letters and numbers of size 16 - 32.

7.26.1 Function Documentation

7.26.1.1 GetRandomString()

```
QString GetRandomString ( )
```

GetRandomString Generates a random string of letters and numbers of size 16 - 32.

Returns

The generated random string.

7.26.1.2 passwordMatches()

```
bool passwordMatches (
            QString prov )
```

passwordMatches Checks if a provided password matches the set password.

Parameters

prov	A provided password that will be checked.

Returns

True if the provided password matches the set password. False otherwise.

7.27 FESS-GUI/transmitbuffer.cpp File Reference

```
#include "transmitbuffer.h"
```

7.28 FESS-GUI/transmitbuffer.h File Reference

```
#include <deque>
#include "flyqueue.h"
#include "flypacket.h"
```

Classes

- class TransmitBuffer

  The Transmit Buffer Class Provides queueing for the communication packets and raw output of packets for interfaces.