RTNETDEMO Example

Table of Contents

RTNETDEMO Example

1 Symbol Reference

1.1 Alignment

Class Hierarchy

Command
  CommandRTNet
    Alignment

```
class Alignment : public CommandRTNet;
```

File

Alignment.h

Description

Demonstrate alignment

Members

Methods

Method	Description
Alignment (see page 1)	Constructor used by factory
RunRTNet (see page 1)	Implementation

Legend

	Method
	virtual

1.1.1 Alignment::Alignment

```
Alignment();
```

Description

Constructor used by factory

1.1.2 Alignment::RunRTNet

```
virtual void RunRTNet() throw(TracedException, codaRTNet::NetworkException,
codaRTNet::DeviceStatusArray);
```

Description

Implementation

1.2 AlignmentFactory

Class Hierarchy

CommandFactory
  CommandFactoryRTNet
    AlignmentFactory

```
class AlignmentFactory : public CommandFactoryRTNet;
```

File

Alignment.h

Description

Factory for alignment commands

Members

Methods

Method	Description
AlignmentFactory (see page 2)	This is AlignmentFactory, a member of class AlignmentFactory.

Legend

	Method

1.2.1 AlignmentFactory::AlignmentFactory

```
AlignmentFactory(ResultLog& _results, codaRTNet::RTNetClient& _client);
```

Description

This is AlignmentFactory, a member of class AlignmentFactory.

1.3 AcqContinuous

Class Hierarchy

Command
  CommandRTNet
    CommandRTNetData
      AcqContinuous

```
class AcqContinuous : public CommandRTNetData;
```

File

AcqContinuous.h

Description

Demonstrate use of continuous acquisition

Members

Methods

Method	Description
AcqContinuous (see page 3)	Constructor used by factory
RunRTNet (see page 3)	Implementation

Legend

	Method
V	virtual

1.3.1 AcqContinuous::AcqContinuous

```
AcqContinuous();
```

Description

Constructor used by factory

1.3.2 AcqContinuous::RunRTNet

```
virtual void RunRTNet() throw(TracedException, codaRTNet::NetworkException,
codaRTNet::DeviceStatusArray);
```

Description

Implementation

1.4 AcqContinuousFactory

Class Hierarchy

CommandFactory
  CommandFactoryRTNet
    CommandFactoryRTNetData
      AcqContinuousFactory

```
class AcqContinuousFactory : public CommandFactoryRTNetData;
```

File

AcqContinuous.h

Description

Factory for continuous acquisition commands

Members

Methods

Method	Description
AcqContinuousFactory (see page 3)	This is AcqContinuousFactory, a member of class AcqContinuousFactory.

Legend

	Method

1.4.1 AcqContinuousFactory::AcqContinuousFactory

```
AcqContinuousFactory(ResultLog& _results, codaRTNet::RTNetClient& _client, DataWriter&
_dataoutput);
```

Description

This is AcqContinuousFactory, a member of class AcqContinuousFactory.

1.5 AcqContinuousBuffered

Class Hierarchy

Command
  CommandRTNet
    CommandRTNetData
      AcqContinuousBuffered

```
class AcqContinuousBuffered : public CommandRTNetData;
```

File

AcqContinuousBuffered.h

Description

Demonstrate use of continuous buffered acquisition

Members

Methods

Method	Description
AcqContinuousBuffered (see page 4)	Constructor used by factory
V RunRTNet (see page 4)	Implementation

Legend

	Method
V	virtual

1.5.1 AcqContinuousBuffered::AcqContinuousBuffered

```
AcqContinuousBuffered();
```

Description

Constructor used by factory

1.5.2 AcqContinuousBuffered::RunRTNet

```
virtual void RunRTNet() throw(TracedException, codaRTNet::NetworkException,
codaRTNet::DeviceStatusArray);
```

Description

Implementation

1.6 AcqContinuousBufferedFactory

Class Hierarchy

CommandFactory
  CommandFactoryRTNet
    CommandFactoryRTNetData
      AcqContinuousBufferedFactory

```
class AcqContinuousBufferedFactory : public CommandFactoryRTNetData;
```

File

AcqContinuousBuffered.h

Description

Factory for continuous buffered acquisition commands

Members

Methods

Method	Description
AcqContinuousBufferedFactory (see page 5)	This is AcqContinuousBufferedFactory, a member of class AcqContinuousBufferedFactory.

Legend

	Method

1.6.1 AcqContinuousBufferedFactory::AcqContinuousBufferedFactory

```
AcqContinuousBufferedFactory(ResultLog& _results, codaRTNet::RTNetClient& _client,
DataWriter& _dataoutput);
```

Description

This is AcqContinuousBufferedFactory, a member of class AcqContinuousBufferedFactory.

1.7 AcqSingleShot

Class Hierarchy

Command
  CommandRTNet
    CommandRTNetData
      AcqSingleShot

```
class AcqSingleShot : public CommandRTNetData;
```

File

AcqSingleShot.h

Description

Demonstrate single-shot acquisition command

Members

Methods

Method	Description
AcqSingleShot (see page 6)	Constructor used by factory
RunRTNet (see page 6)	Implementation

Legend

	Method
	virtual

1.7.1 AcqSingleShot::AcqSingleShot

```
AcqSingleShot();
```

Description

Constructor used by factory

1.7.2 AcqSingleShot::RunRTNet

```
virtual void RunRTNet() throw(TracedException, codaRTNet::NetworkException,
codaRTNet::DeviceStatusArray);
```

Description

Implementation

1.8 AcqSingleShotFactory

Class Hierarchy

CommandFactory
  CommandFactoryRTNet
    CommandFactoryRTNetData
      AcqSingleShotFactory

```
class AcqSingleShotFactory : public CommandFactoryRTNetData;
```

File

AcqSingleShot.h

Description

Factory for single-shot acquisition commands

Members

Methods

Method	Description
AcqSingleShotFactory (see page 7)	This is AcqSingleShotFactory, a member of class AcqSingleShotFactory.

Legend

	Method

1.8.1 AcqSingleShotFactory::AcqSingleShotFactory

```
AcqSingleShotFactory(ResultLog& _results, codaRTNet::RTNetClient& _client, DataWriter&
_dataoutput);
```

Description

This is AcqSingleShotFactory, a member of class AcqSingleShotFactory.

1.9 CodaMode

Class Hierarchy

Command
  CommandRTNet
    CodaMode

```
class CodaMode : public CommandRTNet;
```

File

CodaMode.h

Description

Demonstrate setting of CODA mode

Members

Methods

Method	Description
CodaMode (see page 7)	Constructor used by factory
RunRTNet (see page 7)	Implementation

Legend

	Method
	virtual

1.9.1 CodaMode::CodaMode

```
CodaMode();
```

Description

Constructor used by factory

1.9.2 CodaMode::RunRTNet

```
virtual void RunRTNet() throw(TracedException, codaRTNet::NetworkException,
codaRTNet::DeviceStatusArray);
```

Description

Implementation

1.10 CodaModeFactory

Class Hierarchy

CommandFactory
  CommandFactoryRTNet
    CodaModeFactory

```
class CodaModeFactory : public CommandFactoryRTNet;
```

File

CodaMode.h

Description

Factory for code mode commands

Members

Methods

Method	Description
CodaModeFactory (see page 8)	This is CodaModeFactory, a member of class CodaModeFactory.

Legend

	Method

1.10.1 CodaModeFactory::CodaModeFactory

```
CodaModeFactory(ResultLog& _results, codaRTNet::RTNetClient& _client);
```

Description

This is CodaModeFactory, a member of class CodaModeFactory.

1.11 StartSystem

Class Hierarchy

Command
  CommandRTNet
    StartSystem

```
class StartSystem : public CommandRTNet;
```

File

StartSystem.h

Description

Demonstrate system start-up

Members

Methods

Method	Description
RunRTNet (see page 9)	Implementation
StartSystem (see page 9)	Constructor used by factory

Legend

	Method
	virtual

1.11.1 StartSystem::RunRTNet

```
virtual void RunRTNet() throw(TracedException, codaRTNet::NetworkException,
codaRTNet::DeviceStatusArray);
```

Description

Implementation

1.11.2 StartSystem::StartSystem

```
StartSystem();
```

Description

Constructor used by factory

1.12 StartSystemFactory

Class Hierarchy

CommandFactory
  CommandFactoryRTNet
    StartSystemFactory

```
class StartSystemFactory : public CommandFactoryRTNet;
```

File

StartSystem.h

Description

Factory to make start-system commands

Members

Methods

Method	Description
StartSystemFactory (see page 9)	This is StartSystemFactory, a member of class StartSystemFactory.

Legend

	Method

1.12.1 StartSystemFactory::StartSystemFactory

```
StartSystemFactory(ResultLog& _results, codaRTNet::RTNetClient& _client);
```

Description

This is StartSystemFactory, a member of class StartSystemFactory.

1.13 RTNetDemoOptions

Class Hierarchy

CommandLineOptions
  RTNetDemoOptions

```
class RTNetDemoOptions : protected CommandLineOptions;
```

File

RTNetDemoOptions.h

Description

Command (see page 20) line options class specific to our RTNetDemo program

Members

Methods

Method	Description
CommandFile (see page 10)	Structured text file containing commands
DataFile (see page 10)	File to write data output
Parse (see page 10)	Call CommandLineOptions::Parse (see page 16) and log the resulting set of options
RTNetDemoOptions (see page 11)	This is RTNetDemoOptions, a member of class RTNetDemoOptions.
Server (see page 11)	RT Net server ip address as string

Legend

	Method

1.13.1 RTNetDemoOptions::CommandFile

```
const std::string& CommandFile() const;
```

Description

Structured text file containing commands

1.13.2 RTNetDemoOptions::DataFile

```
const std::string& DataFile() const;
```

Description

File to write data output

1.13.3 RTNetDemoOptions::Parse

```
void Parse(ResultLog& results, int argc, char* argv[]) throw(TracedException);
```

Parameters

Parameters	Description
`ResultLog& results`	Log to write options to
`int argc`	The argc from main

`char* argv[]`	The argv from main

Description

Call CommandLineOptions::Parse (see page 16) and log the resulting set of options

Exceptions

TracedException (see page 18) if there was a problem parsing options in base class

1.13.4 RTNetDemoOptions::RTNetDemoOptions

```
RTNetDemoOptions();
```

Description

This is RTNetDemoOptions, a member of class RTNetDemoOptions.

1.13.5 RTNetDemoOptions::Server

```
const std::string& Server() const;
```

Description

RT Net server ip address as string

1.14 CommandRTNet

Class Hierarchy

Command
  CommandRTNet

```
class CommandRTNet : public Command;
```

File

CommandRTNet.h

Description

Extension of abstract Command (see page 20) class to allow global RT Net SDK client to be set from factory and RT Net SDK exceptions to be caught

Members

Methods

Method	Description
Client (see page 12)	Get client which was set by factory just after construction
Run (see page 12)	Implementation of abstract Run method to use new abstract RunRTNet (see page 12) method and catch RTNet exceptions
RunRTNet (see page 12)	Abstract method to be used by derived classes

Friends

Friend	Description
class CommandFactoryRTNet (see page 12)	This is friend friend class CommandFactoryRTNet.

Legend

	Method
V	virtual
A	abstract

1.14.1 CommandRTNet::Client

```
codaRTNet::RTNetClient& Client();
```

Description

Get client which was set by factory just after construction

1.14.2 CommandRTNet::Run

```
virtual void Run() throw(TracedException);
```

Description

Implementation of abstract Run method to use new abstract RunRTNet (see page 12) method and catch RTNet exceptions

1.14.3 CommandRTNet::RunRTNet

```
virtual void RunRTNet() = 0 throw(codaRTNet::NetworkException,
codaRTNet::DeviceStatusArray);
```

Description

Abstract method to be used by derived classes

1.14.4 friend class CommandFactoryRTNet

```
friend class CommandFactoryRTNet;
```

Description

This is friend friend class CommandFactoryRTNet.

1.15 CommandFactoryRTNet

Class Hierarchy

CommandFactory
  CommandFactoryRTNet

```
class CommandFactoryRTNet : public CommandFactory;
```

File

CommandRTNet.h

Description

Extension of command factory to allow RT Net SDK client to be set in the commands we build

Members

Methods

Method	Description
Build (see page 13)	Build new command and set RT Net SDK client
CommandFactoryRTNet (see page 13)	Construct knowing the log and client which we will give to the commands we build

Legend

	Method
	virtual

1.15.1 CommandFactoryRTNet::Build

```
virtual CommandRTNet* Build(const std::string& name);
```

Parameters

Parameters	Description
`const std::string& name`	Name to use for command module

Description

Build new command and set RT Net SDK client

1.15.2 CommandFactoryRTNet::CommandFactoryRTNet

```
CommandFactoryRTNet(ResultLog& _results, codaRTNet::RTNetClient& _client);
```

Parameters

Parameters	Description
`ResultLog& _results`	Results log to give to commands
`codaRTNet::RTNetClient& _client`	RT Net SDK client to give to commands

Description

Construct knowing the log and client which we will give to the commands we build

1.16 CommandRTNetData

Class Hierarchy

Command
  CommandRTNet
    CommandRTNetData

```
class CommandRTNetData : public CommandRTNet;
```

File

CommandRTNetData.h

Description

Extend abstract RTNet command class also to allow a data output writer to be set by factory

Members

Methods

Method	Description
DataOutput (see page 14)	Get data writer as set by factory just after construction

Friends

Friend	Description
class CommandFactoryRTNetData (see page 14)	This is friend friend class CommandFactoryRTNetData.

Legend

	Method

1.16.1 CommandRTNetData::DataOutput

```
DataWriter& DataOutput();
```

Description

Get data writer as set by factory just after construction

1.16.2 friend class CommandFactoryRTNetData

```
friend class CommandFactoryRTNetData;
```

Description

This is friend friend class CommandFactoryRTNetData.

1.17 CommandFactoryRTNetData

Class Hierarchy

CommandFactory
  CommandFactoryRTNet
    CommandFactoryRTNetData

```
class CommandFactoryRTNetData : public CommandFactoryRTNet;
```

File

CommandRTNetData.h

Description

Extend general RT Net command factory to allow data output writer to be set in the commands we build

Members

Methods

Method	Description
Build (see page 15)	Build new command and set data writer
CommandFactoryRTNetData (see page 15)	Construct knowing the log, client, and data output which we will give to the commands we build

Legend

	Method
V	virtual

1.17.1 CommandFactoryRTNetData::Build

```
virtual CommandRTNetData* Build(const std::string& name);
```

Parameters

Parameters	Description
`const std::string& name`	Name to use for command module

Description

Build new command and set data writer

1.17.2 CommandFactoryRTNetData::CommandFactoryRTNetData

```
CommandFactoryRTNetData(ResultLog& _results, codaRTNet::RTNetClient& _client, DataWriter&
_dataoutput);
```

Parameters

Parameters	Description
`ResultLog& _results`	Results log to give to commands
`codaRTNet::RTNetClient& _client`	RT Net SDK client to give to commands

Description

Construct knowing the log, client, and data output which we will give to the commands we build

1.18 Framework

Framework classes used by the RTNetDemo example

Classes

Class	Description
CommandLineOptions (see page 15)	Register command line options to look for and parse the command line accordingly.
ResultLog (see page 17)	Produce a formatted log
TracedException (see page 18)	An exception class which can log the source file and source code line at which the exception was thrown, plus module name and error message.

Macros

Macro	Description
STOP (see page 19)	Throw a TracedException (see page 18) tagged with this line of source code and source file name.
STOP2 (see page 19)	Throw a TracedException (see page 18) tagged with this line of source code and source file name, and two message strings which are concatenated and separated by a colon.

1.18.1 CommandLineOptions

Class Hierarchy

CommandLineOptions

```
class CommandLineOptions;
```

File

CommandLineOptions.h

Description

Register command line options to look for and parse the command line accordingly.

Members

Methods

Method	Description
CommandLineOptions (see page 16)	Constructor (constructs with no options)
OptionRegister (see page 16)	Get the array of all options registered
Parse (see page 16)	Parse a command line
RegisterOption (see page 16)	Register a possible option with the default value to use if not found

Legend

	Method

1.18.1.1 CommandLineOptions::CommandLineOptions

```
CommandLineOptions();
```

Description

Constructor (constructs with no options)

1.18.1.2 CommandLineOptions::OptionRegister

```
const std::map<std::string, std::string*>& OptionRegister() const;
```

Returns

Array of options

Description

Get the array of all options registered

1.18.1.3 CommandLineOptions::Parse

```
void Parse(int argc, char* argv[]) throw(TracedException);
```

Parameters

Parameters	Description
`int argc`	The argc from main
`char* argv[]`	The argv from main

Description

Parse a command line

Exceptions

TracedException (see page 18)

1.18.1.4 CommandLineOptions::RegisterOption

```
void RegisterOption(const char* name, std::string& option, const char* defaultvalue);
```

Parameters

Parameters	Description
`const char* name`	Name to find on command line (the name on the command line will be prefixed with --)
`std::string& option`	A string object to fill with the value found for this option
`const char* defaultvalue`	The default value to put in option if no value found

Description

Register a possible option with the default value to use if not found

1.18.2 ResultLog

Class Hierarchy

ResultLog

```
class ResultLog;
```

File

ResultLog.h

Description

Produce a formatted log

Members

Methods

Method	Description
Log1 (see page 17)	Log a message
Log2 (see page 17)	Concatenate two messages and log them
LogException (see page 18)	Log the given exception which includes source file name and line of code
ResultLog (see page 18)	Construct log to specified stream

Legend

	Method

1.18.2.1 ResultLog::Log1

```
void Log1(const std::string& module, const std::string& message);
```

Parameters

Parameters	Description
`const std::string& module`	Module name to use in prefix to message
`const std::string& message`	Message to write

Description

Log a message

1.18.2.2 ResultLog::Log2

```
void Log2(const std::string& module, const std::string& message1, const std::string&
message2);
```

Parameters

Parameters	Description
`const std::string& module`	Module name to use in prefix to message
`const std::string& message1`	First message to write, will be followed by colon and space

`const std::string& message2`	Second message to write

Description

Concatenate two messages and log them

1.18.2.3 ResultLog::LogException

```
void LogException(const std::string& module, const TracedException& e);
```

Parameters

Parameters	Description
`const std::string& module`	Module to use in prefix (should be the logging module, not necessarily the module which threw the exception)
`const TracedException& e`	Exception to log

Description

Log the given exception which includes source file name and line of code

1.18.2.4 ResultLog::ResultLog

```
ResultLog(std::ostream& _diag);
```

Parameters

Parameters	Description
`std::ostream& _diag`	Stream to log to

Description

Construct log to specified stream

1.18.3 TracedException

An exception class which can log the source file and source code line at which the exception was thrown, plus module name and error message.

Class Hierarchy

TracedException

```
class TracedException;
```

File

TracedException.h

Description

This class is not to be thrown directly but rather via the STOP (see page 19) and STOP2 (see page 19) macros which use the C preprocessor to log the source file and source line at which they are used.

Members

Methods

Method	Description
Report (see page 19)	Print this exception to the given stream
TracedException (see page 19)	Copy constructor
TracedException (see page 19)	Constructor to be called from STOP (see page 19) and STOP2 (see page 19)

Legend

	Method

1.18.3.1 TracedException::Report

```
void Report(std::ostream& report) const;
```

Parameters

Parameters	Description
std::ostream& report	The stream to print to

Description

Print this exception to the given stream

1.18.3.2 TracedException::TracedException

```
TracedException(const TracedException& src);
```

Parameters

Parameters	Description
const TracedException& src	Instance to copy

Description

Copy constructor

1.18.3.3 TracedException::TracedException

```
TracedException(const std::string& _module, const char* _sourcefile, unsigned long
_sourceline, const std::string& _message);
```

Description

Constructor to be called from STOP (see page 19) and STOP2 (see page 19)

1.18.4 STOP

```
#define STOP(_module, _message) throw TracedException(_module, __FILE__, __LINE__, _message)
```

File

TracedException.h

Parameters

Parameters	Description
_module	Module name to put in the exception
_message	Error message to put in the exception

Description

Throw a TracedException (see page 18) tagged with this line of source code and source file name.

1.18.5 STOP2

```
#define STOP2(_module, _message1, _value) STOP(_module, std::string(_message1) +
std::string(": ") + _value)
```

File

TracedException.h

Parameters

Parameters	Description
`_module`	Module name to put in the exception
`_message1`	Error message to put in the exception
`_value`	Value to append to error message following a colon

Description

Throw a TracedException (see page 18) tagged with this line of source code and source file name, and two message strings which are concatenated and separated by a colon.

1.18.6 Commands

Classes to allow a list of commands to be created and run.

Classes

Class	Description
Command (see page 20)	The purpose of this class is to have an abstract way to describe commands such that a list of them can be read from a file and executed. Derive from this class to add scriptable commands to the program. Parameters can be added to the command by having member variables derived from Parameter (see page 25) which are registered using the RegisterParameter (see page 22) method.
CommandFactory (see page 23)	A factory object for building commands of a specified type Derive from this to implement factories for specific commands
CommandList (see page 23)	A list of command factories and the ability to manufacture commands by name and keep them in a local list
Parameter (see page 25)	Base class for parameters which we can register with our command objects
ParameterInteger (see page 26)	Integer command parameter
ParameterString (see page 27)	Text string command parameter

Macros

Macro	Description
COMMAND_STOP (see page 28)	Call the STOP (see page 19) macro to throw a TracedException (see page 18) to log this source file and line plus this command's name
COMMAND_STOP2 (see page 28)	Call the STOP2 (see page 19) macro to throw a TracedException (see page 18) to log this source file and line plus this command's name. Also like STOP2 (see page 19) this will concatenate message and value.

1.18.6.1 Command

The purpose of this class is to have an abstract way to describe commands such that a list of them can be read from a file and executed. Derive from this class to add scriptable commands to the program. Parameters can be added to the command by having member variables derived from Parameter (see page 25) which are registered using the RegisterParameter (see page 22) method.

Class Hierarchy

Command

```
class Command;
```

File

Command.h

Description

An abstract command which can be executed by our program.

Members

Methods

Method	Description
~Command (see page 21)	Virtual destructor
Command (see page 21)	Protected empty constructor to be used by the CommandFactory (see page 23)
Comment (see page 21)	Add message to log using this command's name as module name
Comment (see page 21)	Add contents of string stream to log using this command's name as module name
Comment1 (see page 21)	Add message and string to log using this command's name as module name
Comment1 (see page 22)	Add message and number to log using this command's name as module name
GetParameter (see page 22)	Get the wrapper object for a parameter
HaveRequiredParameters (see page 22)	See if all required parameters are set
Name (see page 22)	Name of this command
RegisterParameter (see page 22)	Register a parameter name used by this command - useful for looking up parameters in script file
Run (see page 22)	Abstract command method to be implemented by derived classes

Friends

Friend	Description
class CommandFactory (see page 23)	This is friend friend class CommandFactory.

Legend

	Method
	virtual
	protected
	abstract

1.18.6.1.1 Command::~Command

```
virtual ~Command();
```

Description

Virtual destructor

1.18.6.1.2 Command::Command

```
Command();
```

Description

Protected empty constructor to be used by the CommandFactory (see page 23)

1.18.6.1.3 Command::Comment

```
void Comment(const std::string& message);
```

Description

Add message to log using this command's name as module name

1.18.6.1.4 Command::Comment

```
void Comment(std::ostrstream& formattedmessage);
```

Description

Add contents of string stream to log using this command's name as module name

1.18.6.1.5 Command::Comment1

```
void Comment1(const std::string& message, const std::string& value);
```

Description

Add message and string to log using this command's name as module name

1.18.6.1.6 Command::Comment1

```
void Comment1(const std::string& message, long value);
```

Description

Add message and number to log using this command's name as module name

1.18.6.1.7 Command::GetParameter

```
Parameter* GetParameter(const std::string& str);
```

Parameters

Parameters	Description
`const std::string& str`	name of parameter to find

Returns

The parameter wrapper if found, NULL otherwise

Description

Get the wrapper object for a parameter

1.18.6.1.8 Command::HaveRequiredParameters

```
bool HaveRequiredParameters() const;
```

Returns

true if all parameters filled, false otherwise

Description

See if all required parameters are set

1.18.6.1.9 Command::Name

```
const std::string& Name() const;
```

Description

Name of this command

1.18.6.1.10 Command::RegisterParameter

```
void RegisterParameter(const std::string& parametername, Parameter& parameter);
```

Description

Register a parameter name used by this command - useful for looking up parameters in script file

1.18.6.1.11 Command::Run

```
virtual void Run() = 0 throw(TracedException);
```

Description

Abstract command method to be implemented by derived classes

1.18.6.1.12 friend class CommandFactory

```
friend class CommandFactory;
```

Description

This is friend friend class CommandFactory.

1.18.6.2 CommandFactory

Class Hierarchy

CommandFactory

```
class CommandFactory;
```

File

CommandFactory.h

Description

A factory object for building commands of a specified type Derive from this to implement factories for specific commands

Members

Methods

Method	Description
Build (see page 23)	Build a new command object
CommandFactory (see page 23)	Construct

Legend

	Method
	virtual

1.18.6.2.1 CommandFactory::Build

```
virtual Command* Build(const std::string& name);
```

Description

Build a new command object

1.18.6.2.2 CommandFactory::CommandFactory

```
CommandFactory(ResultLog& _results);
```

Parameters

Parameters	Description
`ResultLog& _results`	Results log to give to the command objects we make

Description

Construct

1.18.6.3 CommandList

Class Hierarchy

CommandList

```
class CommandList;
```

File

CommandList.h

Description

A list of command factories and the ability to manufacture commands by name and keep them in a local list

Members

Methods

Method	Description
~CommandList (see page 24)	Destructor
BuildCommand (see page 24)	Look up factory with specified command name, make new command and store in list
CommandList (see page 24)	Construct empty
RegisterCommand (see page 24)	Register a command factory by name
RunAll (see page 25)	Run all commands in local list (will have been made using BuildCommand (see page 24))

Legend

	Method
	virtual

1.18.6.3.1 CommandList::~CommandList

```
virtual ~CommandList();
```

Description

Destructor

1.18.6.3.2 CommandList::BuildCommand

```
Command* BuildCommand(const std::string& commandname);
```

Parameters

Parameters	Description
`const std::string& commandname`	Name to find

Description

Look up factory with specified command name, make new command and store in list

1.18.6.3.3 CommandList::CommandList

```
CommandList();
```

Description

Construct empty

1.18.6.3.4 CommandList::RegisterCommand

```
void RegisterCommand(const std::string& name, CommandFactory* factory);
```

Parameters

Parameters	Description
`const std::string& name`	Name to register
`CommandFactory* factory`	Corresponding factory

Description

Register a command factory by name

1.18.6.3.5 CommandList::RunAll

```
void RunAll() const throw(TracedException);
```

Description

Run all commands in local list (will have been made using BuildCommand (see page 24))

Exceptions

TracedException (see page 18) if a command fails

1.18.6.4 Parameter

Class Hierarchy

Parameter

```
class Parameter;
```

File

Parameter.h

Description

Base class for parameters which we can register with our command objects

Members

Methods

Method	Description
V ~Parameter (see page 25)	Destructor
IsSet (see page 25)	See if successfully parsed
Parameter (see page 25)	Construct empty / default value
A Parse (see page 26)	Parse from string read from structured text

Legend

	Method
V	virtual
A	abstract

1.18.6.4.1 Parameter::~Parameter

```
virtual ~Parameter();
```

Description

Destructor

1.18.6.4.2 Parameter::IsSet

```
bool IsSet() const;
```

Returns

true if successfully parsed

Description

See if successfully parsed

1.18.6.4.3 Parameter::Parameter

```
Parameter();
```

Description

Construct empty / default value

1.18.6.4.4 Parameter::Parse

```
virtual void Parse(const std::string& str) = 0;
```

Parameters

Parameters	Description
`const std::string& str`	Text to parse

Description

Parse from string read from structured text

1.18.6.5 ParameterInteger

Class Hierarchy

Parameter
  ParameterInteger

```
class ParameterInteger : public Parameter;
```

File

ParameterInteger.h

Description

Integer command parameter

Members

Methods

Method	Description
~ParameterInteger (see page 26)	Destructor
ParameterInteger (see page 26)	Construct and initialise to zero
Parse (see page 27)	Parse string to integer
Value (see page 27)	Get value of integer

Legend

	Method
	virtual

1.18.6.5.1 ParameterInteger::~ParameterInteger

```
virtual ~ParameterInteger();
```

Description

Destructor

1.18.6.5.2 ParameterInteger::ParameterInteger

```
ParameterInteger();
```

Description

Construct and initialise to zero

1.18.6.5.3 ParameterInteger::Parse

```
virtual void Parse(const std::string& str);
```

Description

Parse string to integer

1.18.6.5.4 ParameterInteger::Value

```
const long Value() const;
```

Returns

value

Description

Get value of integer

1.18.6.6 ParameterString

Class Hierarchy

Parameter
  ParameterString

```
class ParameterString : public Parameter;
```

File

ParameterString.h

Description

Text string command parameter

Members

Methods

Method	Description
~ParameterString (see page 27)	Destructor
ParameterString (see page 27)	Construct empty
Parse (see page 28)	Parse from string (just copies)
Value (see page 28)	Get value

Legend

	Method
	virtual

1.18.6.6.1 ParameterString::~ParameterString

```
virtual ~ParameterString();
```

Description

Destructor

1.18.6.6.2 ParameterString::ParameterString

```
ParameterString();
```

Description

Construct empty

1.18.6.6.3 ParameterString::Parse

```
virtual void Parse(const std::string& str);
```

Parameters

Parameters	Description
`const std::string& str`	String to copy

Description

Parse from string (just copies)

1.18.6.6.4 ParameterString::Value

```
const std::string& Value() const;
```

Returns

text value

Description

Get value

1.18.6.7 COMMAND_STOP

```
#define COMMAND_STOP(_message) STOP(Name(), _message)
```

File

Command.h

Parameters

Parameters	Description
`_message`	Error message to log

Description

Call the STOP (see page 19) macro to throw a TracedException (see page 18) to log this source file and line plus this command's name

Exceptions

TracedException (see page 18)

1.18.6.8 COMMAND_STOP2

```
#define COMMAND_STOP2(_message, _value) STOP2(Name(), _message, _value)
```

File

Command.h

Parameters

Parameters	Description
`_message`	Error message to log

Description

Call the STOP2 (see page 19) macro to throw a TracedException (see page 18) to log this source file and line plus this command's name. Also like STOP2 (see page 19) this will concatenate message and value.

Exceptions

TracedException (see page 18)

1.18.7 IO

Input and output to text files

Classes

Class	Description
DataAcquisition (see page 29)	Data acquisition as read from structured text
DataPacket (see page 30)	Data packet as read from structured text corresponding to RTNetworkPacket
DataPoint (see page 32)	Data point (element of packet) as read from structured text
DataReader (see page 34)	Read data from structured text as output by DataWriter (see page 36)
DataWriter (see page 36)	Write data to a structured text file
TextReader (see page 38)	Read structured text files
CommandListReader (see page 40)	Read (see page 41) list of commands from structured text file

1.18.7.1 DataAcquisition

Class Hierarchy

DataAcquisition

```
class DataAcquisition;
```

File

DataReader.h

Description

Data acquisition as read from structured text

Members

Methods

Method	Description
AddPacket (see page 29)	Append a packet corresponding to RTNetworkPacket information
DataAcquisition (see page 30)	Construct empty
GetDeviceSampleRate (see page 30)	Retrieve sample rate info
Packets (see page 30)	Get array of packets
SetDeviceSampleRate (see page 30)	Append device sample-rate settings

Legend

	Method

1.18.7.1.1 DataAcquisition::AddPacket

```
void AddPacket(DataPacket& packet);
```

Parameters

Parameters	Description
`DataPacket& packet`	Packet to append

Description

Append a packet corresponding to RTNetworkPacket information

1.18.7.1.2 DataAcquisition::DataAcquisition

```
DataAcquisition();
```

Description

Construct empty

1.18.7.1.3 DataAcquisition::GetDeviceSampleRate

```
bool GetDeviceSampleRate(long device, float& rate);
```

Parameters

Parameters	Description
`long device`	Device whose rate to look up
`float& rate`	Filled with rate if found

Returns

true if found, false otherwise

Description

Retrieve sample rate info

1.18.7.1.4 DataAcquisition::Packets

```
const std::vector<DataPacket>& Packets() const;
```

Description

Get array of packets

1.18.7.1.5 DataAcquisition::SetDeviceSampleRate

```
void SetDeviceSampleRate(long device, float rate);
```

Parameters

Parameters	Description
`long device`	Device to set
`float rate`	Rate in Hz

Description

Append device sample-rate settings

1.18.7.2 DataPacket

Class Hierarchy

DataPacket

```
class DataPacket;
```

File

DataReader.h

Description

Data packet as read from structured text corresponding to RTNetworkPacket

Members

Data Members

Data Member	Description
device (see page 31)	Device ID from RTNetworkPacket
page (see page 31)	Device page from RTNetworkPacket
tick (see page 31)	Device tick from RTNetworkPacket

Methods

Method	Description
= (see page 31)	Copy operator
AddPoint (see page 31)	Append a point
DataPacket (see page 32)	Construct empty
DataPacket (see page 32)	Copy constructor - calls copy operator
Points (see page 32)	Get points array

Legend

	Data Member
	Method

1.18.7.2.1 DataPacket::device

```
long device;
```

Description

Device ID from RTNetworkPacket

1.18.7.2.2 DataPacket::page

```
long page;
```

Description

Device page from RTNetworkPacket

1.18.7.2.3 DataPacket::tick

```
long tick;
```

Description

Device tick from RTNetworkPacket

1.18.7.2.4 DataPacket::=

```
const DataPacket& operator =(const DataPacket& src);
```

Description

Copy operator

1.18.7.2.5 DataPacket::AddPoint

```
void AddPoint(const DataPoint& p);
```

Parameters

Parameters	Description
`const DataPoint& p`	Point to append

Description

Append a point

1.18.7.2.6 DataPacket::DataPacket

```
DataPacket();
```

Description

Construct empty

1.18.7.2.7 DataPacket::DataPacket

```
DataPacket(const DataPacket& src);
```

Description

Copy constructor - calls copy operator

1.18.7.2.8 DataPacket::Points

```
const std::vector<DataPoint>& Points() const;
```

Description

Get points array

1.18.7.3 DataPoint

Class Hierarchy

DataPoint

```
class DataPoint;
```

File

DataReader.h

Description

Data point (element of packet) as read from structured text

Members

Methods

Method	Description
= (see page 33)	Copy operator
AddIntensity (see page 33)	Append intensity value
AddResidual (see page 33)	Append residual value
AddValue (see page 33)	Append coordinate value
DataPoint (see page 33)	Construct empty
DataPoint (see page 33)	Copy constructor - calls copy operator=()
Intensity (see page 34)	Get intensity array
Occluded (see page 34)	Get occlusion flag
Residual (see page 34)	Get residual array
SetOccluded (see page 34)	Set occlusion flag
Value (see page 34)	Get value array

Legend

	Method

1.18.7.3.1 DataPoint::=

```
const DataPoint& operator =(const DataPoint& src);
```

Description

Copy operator

1.18.7.3.2 DataPoint::AddIntensity

```
void AddIntensity(long i);
```

Parameters

Parameters	Description
long i	intensity

Description

Append intensity value

1.18.7.3.3 DataPoint::AddResidual

```
void AddResidual(long r);
```

Parameters

Parameters	Description
long r	residual

Description

Append residual value

1.18.7.3.4 DataPoint::AddValue

```
void AddValue(float x);
```

Parameters

Parameters	Description
float x	Value (see page 34)

Description

Append coordinate value

1.18.7.3.5 DataPoint::DataPoint

```
DataPoint();
```

Description

Construct empty

1.18.7.3.6 DataPoint::DataPoint

```
DataPoint(const DataPoint& src);
```

Description

Copy constructor - calls copy operator=()

1.18.7.3.7 DataPoint::Intensity

```
const std::vector<long>& Intensity() const;
```

Description

Get intensity array

1.18.7.3.8 DataPoint::Occluded

```
bool Occluded() const;
```

Description

Get occlusion flag

1.18.7.3.9 DataPoint::Residual

```
const std::vector<long>& Residual() const;
```

Description

Get residual array

1.18.7.3.10 DataPoint::SetOccluded

```
void SetOccluded(bool o);
```

Parameters

Parameters	Description
`bool o`	flag value

Description

Set occlusion flag

1.18.7.3.11 DataPoint::Value

```
const std::vector<float>& Value() const;
```

Description

Get value array

1.18.7.4 DataReader

Class Hierarchy

TextReader
  DataReader

```
class DataReader : protected TextReader;
```

File

DataReader.h

Description

Read data from structured text as output by DataWriter (see page 36)

Members

Methods

Method	Description
DataReader (see page 35)	Construct for specified stream
LoadAcquisition (see page 35)	Load structed acquisition object as writted using DataWriter (see page 36)
LocateObject (see page 35)	Find start of object with specified type and title
ParseFloat32 (see page 36)	Helper to parse string to 32-bit floating point
ParseInt32 (see page 36)	Helper to parse string to 32-bit integer

Legend

	Method

1.18.7.4.1 DataReader::DataReader

```
DataReader(std::istream& _stream);
```

Parameters

Parameters	Description
`std::istream& _stream`	Stream to read from

Description

Construct for specified stream

1.18.7.4.2 DataReader::LoadAcquisition

```
void LoadAcquisition(DataAcquisition& acquisition, const std::string& title)
throw(TracedException);
```

Parameters

Parameters	Description
`DataAcquisition& acquisition`	To fill with acquisition read
`const std::string& title`	The title of the acquisition object to read

Description

Load structed acquisition object as writted using DataWriter (see page 36)

Exceptions

TracedException (see page 18) if problems parsing acquisition

1.18.7.4.3 DataReader::LocateObject

```
bool LocateObject(const std::string& objtype, const std::string& title)
throw(TracedException);
```

Parameters

Parameters	Description
`const std::string& objtype`	type to look for
`const std::string& title`	title to look for

Returns

true if found, false if not

Description

Find start of object with specified type and title

Exceptions

TracedException (see page 18) if there was a problem parsing type or title

1.18.7.4.4 DataReader::ParseFloat32

```
static float ParseFloat32(const std::string& value) throw(TracedException);
```

Parameters

Parameters	Description
`const std::string& value`	String to parse

Returns

parsed value

Description

Helper to parse string to 32-bit floating point

Exceptions

TracedException (see page 18) if could not parse

1.18.7.4.5 DataReader::ParseInt32

```
static long ParseInt32(const std::string& value) throw(TracedException);
```

Parameters

Parameters	Description
`const std::string& value`	String to parse

Returns

parsed value

Description

Helper to parse string to 32-bit integer

Exceptions

TracedException (see page 18) if could not parse

1.18.7.5 DataWriter

Write data to a structured text file

Class Hierarchy

DataWriter

```
class DataWriter;
```

File

DataWriter.h

Description

Writes data in structured text format. This is intended to be a subset of the JSON standard JSON format (http://www.json.org/) though some aspects such as escaped character sequences are not supported.

Members

Methods

Method	Description
AddAcquisitionPacket (see page 37)	Write acquisition packet in structured format - supports 3DResultExt and ADC16 packet layouts
AddVersion (see page 37)	Write version information
BeginAcquisition (see page 37)	Begin an acquisition object.

BeginFile (see page 37)	Write file opening section, just an open bracket
DataWriter (see page 37)	Construct to write to specified stream
EndAcquisition (see page 38)	Write end-of-acquisition characters
EndFile (see page 38)	Complete file to form JSON we can parse

Legend

	Method

1.18.7.5.1 DataWriter::AddAcquisitionPacket

```
void AddAcquisitionPacket(const codaRTNet::RTNetworkPacket& packet, bool sync_stop = false);
```

Parameters

Parameters	Description
`const codaRTNet::RTNetworkPacket& packet`	Packet in know format @parma sync_stop Useful for external sync diagnosics - sets an external sync flag in packet

Description

Write acquisition packet in structured format - supports 3DResultExt and ADC16 packet layouts

1.18.7.5.2 DataWriter::AddVersion

```
void AddVersion(const std::string& title, const codaRTNet::Version& v);
```

Parameters

Parameters	Description
`const std::string& title`	The title to associate with this version
`const codaRTNet::Version& v`	Version using RTNet SDK version class

Description

Write version information

1.18.7.5.3 DataWriter::BeginAcquisition

Begin an acquisition object.

```
void BeginAcquisition(const std::string& title, const std::map<WORD, float>&
device_sample_rate);
```

Parameters

Parameters	Description
`const std::string& title`	The title to write
`Sample`	rate list to write

Description

Will write a type and title, followed by sample rates for each device, followed by start of packet list.

1.18.7.5.4 DataWriter::BeginFile

```
void BeginFile();
```

Description

Write file opening section, just an open bracket

1.18.7.5.5 DataWriter::DataWriter

```
DataWriter(std::ostream& _stream);
```

Parameters

Parameters	Description
`std::ostream& _stream`	Stream to write to

Description

Construct to write to specified stream

1.18.7.5.6 DataWriter::EndAcquisition

```
void EndAcquisition();
```

Description

Write end-of-acquisition characters

1.18.7.5.7 DataWriter::EndFile

```
void EndFile();
```

Description

Complete file to form JSON we can parse

1.18.7.6 TextReader

Read structured text files

Class Hierarchy

TextReader

```
class TextReader;
```

File

TextReader.h

Description

This is designed to read parts of the structured text files which are output from DataWriter (see page 36). They use a subset of the JSON format (http://www.json.org/).

Members

Methods

Method	Description
NextNonSpace (see page 38)	Skip to next non-space and read it
ParseKeyValue (see page 39)	Parse a key and value separated by colon and arbitary numbers of space characters
ParseQuotedString (see page 39)	Read from quote character until and including the next quote character and extract the string in between.
ParseValue (see page 39)	Parse a value.
SkipSpace (see page 40)	Skip space
TextReader (see page 40)	Construct to read from given stream

Legend

	Method

1.18.7.6.1 TextReader::NextNonSpace

```
int NextNonSpace();
```

Returns

The value of next non-space character

Description

Skip to next non-space and read it

1.18.7.6.2 TextReader::ParseKeyValue

Parse a key and value separated by colon and arbitary numbers of space characters

```
void ParseKeyValue(std::string& key, std::string& value) throw(TracedException);
```

Parameters

Parameters	Description
`std::string& key`	Filled with key read
`std::string& value`	Filled with value

Description

Parse a key-value pair. Supports quoted and non-quoted values. For example:

"mykey" : "myvalue"

Or:

"some number" : 3

Exceptions

TracedException (see page 18) if stream is not positioned before such a pair

1.18.7.6.3 TextReader::ParseQuotedString

```
void ParseQuotedString(std::string& str) throw(TracedException);
```

Parameters

Parameters	Description
`std::string& str`	Filled with the string between the quotes

Description

Read from quote character until and including the next quote character and extract the string in between.

Exceptions

TracedException (see page 18) if the stream is not positioned just before a quote

1.18.7.6.4 TextReader::ParseValue

Parse a value.

```
void ParseValue(std::string& value) throw(TracedException);
```

Parameters

Parameters	Description
`str`	Filled with the value read

Description

If the next nons-space character is a quote, this will read up to and including the next quote and return the string in between (like ParseQuotedString (see page 39)). But if the next nons-space character is not a quote then it will read up until any space or control character is found, and so can read numbers which are not generally enclosed in quotes

Exceptions

TracedException (see page 18)

1.18.7.6.5 TextReader::SkipSpace

```
void SkipSpace();
```

Description

Skip space

1.18.7.6.6 TextReader::TextReader

```
TextReader(std::istream& _stream);
```

Parameters

Parameters	Description
`std::istream& _stream`	Stream to read

Description

Construct to read from given stream

1.18.7.7 CommandListReader

Class Hierarchy

TextReader
  CommandListReader

```
class CommandListReader : protected TextReader;
```

File

CommandListReader.h

Description

Read (see page 41) list of commands from structured text file

Members

Methods

Method	Description
CommandListReader (see page 40)	Construct to read from specified stream
Read (see page 41)	Parse according to registered command factories and build command instances

Legend

	Method

1.18.7.7.1 CommandListReader::CommandListReader

```
CommandListReader(std::istream& _stream);
```

Parameters

Parameters	Description
`std::istream& _stream`	Stream to read

Description

Construct to read from specified stream

1.18.7.7.2 CommandListReader::Read

```
void Read(CommandList& commandlist) throw(TracedException);
```

Parameters

Parameters	Description
`CommandList& commandlist`	Command (see page 20) list to populate

Description

Parse according to registered command factories and build command instances

Exceptions

TracedException (see page 18) if structured text format was not as expected

1.18.8 Tests

These are automated self-tests for certain sections of code.

Functions

Function	Description
SelfTest (see page 41)	Run all self-tests
SelfTestCommandLine (see page 42)	Test command line parsing
SelfTestDataReader (see page 42)	Test data reader
SelfTestDataWriter (see page 42)	Test data writer
SelfTestScripts (see page 42)	Test command script parsing and execution

Macros

Macro	Description
SELF_TEST_ASSERT (see page 43)	Self-test failure if condition is false. The code given for the condition will be logged as the failure message
SELF_TEST_FAIL (see page 43)	Indicate self-test failure with specified message

Variables

Variable	Description
SelfTestModuleName (see page 43)	Identifier to use for module name if an exception is thrown during self-test

1.18.8.1 SelfTest

```
void SelfTest(ResultLog& results) throw(TracedException);
```

File

SelfTest.h

Parameters

Parameters	Description
`ResultLog& results`	Log list of tests to this log

Description

Run all self-tests

Exceptions

TracedException (see page 18)

1.18.8.2 SelfTestCommandLine

```
void SelfTestCommandLine() throw(TracedException);
```

File

SelfTestCommandLine.h

Description

Test command line parsing

Exceptions

TracedException (see page 18)

1.18.8.3 SelfTestDataReader

```
void SelfTestDataReader() throw(TracedException);
```

File

SelfTestDataReader.h

Description

Test data reader

Exceptions

TracedException (see page 18)

1.18.8.4 SelfTestDataWriter

```
void SelfTestDataWriter() throw(TracedException);
```

File

SelfTestDataWriter.h

Description

Test data writer

Exceptions

TracedException (see page 18)

1.18.8.5 SelfTestScripts

```
void SelfTestScripts() throw(TracedException);
```

File

SelfTestScripts.h

Description

Test command script parsing and execution

Exceptions

TracedException (see page 18)

1.18.8.6 SelfTestModuleName

```
const char* SelfTestModuleName;
```

File

SelfTestMacros.h

Description

Identifier to use for module name if an exception is thrown during self-test

1.18.8.7 SELF_TEST_ASSERT

```
#define SELF_TEST_ASSERT(condition) { if (!(condition)) { SELF_TEST_FAIL(#condition); } }
```

File

SelfTestMacros.h

Parameters

Parameters	Description
condition	Condition to test

Description

Self-test failure if condition is false. The code given for the condition will be logged as the failure message

1.18.8.8 SELF_TEST_FAIL

```
#define SELF_TEST_FAIL(_message) STOP(SelfTestModuleName, _message)
```

File

SelfTestMacros.h

Parameters

Parameters	Description
_message	The message to log

Description

Indicate self-test failure with specified message

Index

F

I

P

R

S

T