BMeasure-lib

0.2.8

Generated by Doxygen 1.8.14

Contents

Chapter 1

Main Page

Author

Dr Terry Barnaby

Version

0.2.8

Date

2019-10-14

1.1 Introduction

The Beam BMeasure-125i unit is a flexible and powerful IoT system for data capture, data logging and control in the laboratory, industrial and remote sensing arenas. It is based around an 8 channel, fully differential, synchronous sampling, 24 bit ADC that can sample at speeds up to 128 ksps. Multiple units can be connected together to provide more synchronously sampled channels.

This reference information describes the data types and functions provided by the host API library allowing programs to be written to control the operation of a BMeasure unit and acquire the data from it. The API operates over a number of different physical interfaces including: USB 2.0, Ethernet, Wifi and RS485.

In addition there is a software manual providing an overview of using this API which should be read first. This document is available at:: `https://portal.beam.ltd.uk/files/products/bmeasure-125i/doc/↩ BMeasure-api.pdf`

1.2 Overview

The BMeasure API library, bmeasure-lib, is implemented in the C++ computer language. It has bindings layered on top of this for Python, with Matlab due to be supported soon. The API has an object orientated architecture. It has been designed as a general purpose API library for the Beam BMeasure-125i and future BMeasure products. Currently it has ports to Linux (Redhat7, Fedora29, Debian) and Microsoft Windows 7, 8 and 10.

The API provides the following functionality:

- Find BMeasure units on the USB bus or local Ethernet and Wifi networks.
- Connect to one or more BMeasure units.
- Fetch information and configure the BMeasure units.
- Start the BMeasure unit capturing and processing the sensor inputs.
- Capture the data from all of the analogue and digital channels from one or a combined set of BMeasure units running in sync.
- Access the data log files on the unit and download them to the host.
- Configure the AWG to produce waveforms or set voltages on the analogue output channels.
- Operate relays, read switches and other auxiliary operations.

The BMeasure API is implemented using the Beam BOAP (Beam Object Access protocol) communications system. It offers an BMeasureUnit API class to access an individual BMeasure unit in a relatively low level manner and an BMeasureUnits API class to access a set of BMeasure units synchronised together to operate as a single unit and with a queued data reception system..

The API supports threaded and non-threaded operation.

The referenve information provided describes the API from a C++ programming perspective. The Python and other language bindings are very similar the differences being noted under the particular language bindings section in the software manual..

1.3 API Usage

To use the API the core procedure is:

1. Either find the available BMeasure units using: BMeasureApi::BMeasureUnit::findDevices() or use a BMeasure URL string..
2. Choose to use the simple single unit interface BMeasureApi::BMeasureUnit or the BMeasureApi::BMeasureUnits classes.
3. If using the simple single unit interface, connect to the unit using the BMeasureApi::BMeasureUnit::connect() function.
4. If using the multiple unit interface, add the units using the BMeasureApi::BMeasureUnits::unitAdd() function and connect using the BMeasureApi::BMeasureUnits::unitsConnect() function.
5. Use the interface to communicate to the unit.

See the examples below and the software manual for more details.

1.4 API Usage

There are some examples of client applications using the BMeasure API in the **examples** directory of the source code. Some simple client examples are listed below:

Simple example to access and read single sets of data samples in C++

```
/*******************************************************************************
 *      Example005-dataClient-single.cpp
 *      T.Barnaby,      BEAM Ltd,       2019-10-09
 *******************************************************************************
 */
#include <BMeasureUnit.h>
#include <unistd.h>

using namespace BMeasureApi;

// Function to read some data
BError test1(){
        BError                          err;
        BList<BMeasureUnitDevice>       devices;
        BString                         device;
        BMeasureUnit                        bmeasure;
        Information                  info;
        Configuration                      config;
        MeasurementConfig             mc;
        DataBlock                     data;
        BUInt                           c;

        printf("Start Processing Task\n");
        bmeasure.start();

        printf("Find BMeasure units\n");
        if(err = BMeasureUnit::findDevicesUsb(devices)){
                return err;
        }
        if(devices.number() == 0){
                return err.set(1, "No USB BMeasure units found\n");
        }
        device = devices[0].device;

        printf("Connect\n");
        if(err = bmeasure.connect(device))
                return err;

        //printf("Exit\n"); return err;

        printf("Get Info\n");
        if(err = bmeasure.getInformation(info))
                return err;

        printf("NumChannels: %d\n", info.numChannels);

        //printf("Exit\n"); return err;

        printf("Get Config\n");
        if(err = bmeasure.getConfig(config))
                return err;

        printf("Configure measurement\n");
        mc.measureMode = MeasureModeOneShot;
        mc.triggerMode = TriggerModeOff;
        mc.triggerConfig = TriggerConfigNone;
        mc.triggerChannel = 0;
        mc.triggerLevel = 0;
        mc.triggerDelay = 0;
        mc.sampleRate = 8000.0;
        mc.measurePeriod = 0;
        mc.numSamples0 = 1;
        mc.numSamples1 = 0;
        if(err = bmeasure.setMeasurementConfig(mc))
                return err;

        printf("Run single measurement\n");
        if(err = bmeasure.measure(data))
                return err;

        printf("DataBlock: from: %d numChannels: %d numSamples: %d\n", data.source,
      data.numChannels, data.numSamples);
        for(c = 0; c < data.numChannels; c++){
                printf("%f ", data.data[c]);
        }
        printf("\n");

        return err;
}

int main(){
        BError  err;

        if(err = test1()){
                printf("Error: %d %s\n", err.getErrorNo(), err.str());
                return 1;
        }

        printf("Complete\n");

        return 0;
}
```

Simple example to access and read single sets of data samples in Python

```
#!/usr/bin/python3

import sys
import time
import getopt
from threading import Thread
from bmeasure import *

# Function to read some data
def test1():
        bmeasure = BMeasureUnit(True);

        print("Find BMeasure units");
        (err, devices) = BMeasureUnit.findDevicesUsb();
        if(err):
                return err;

        if(devices.number() == 0):
                return err.set(1, "No USB BMeasure units found\n");

        print("Found", len(devices));
        device = devices[0].device;

        print("Start Processing Task");
        bmeasure.start();

        print("Connect to BMeasure");
        err = bmeasure.connect(device);
        if(err):
                return err;

        print("Get Info");
        (err, info) = bmeasure.getInformation();
        if(err):
                return err;

        print("NumChannels: ", info.numChannels);

        print("Get Config");
        (err, config) = bmeasure.getConfig();
        if(err):
                return err;

        # Set configuration
        print("Set Config");
        config.captureData = 1;
        config.source = 10;
        err = bmeasure.setConfig(config);
        if(err):
                return err;

        print("Get measurement config");
        (err, mc) = bmeasure.getMeasurementConfig();
        if(err):
                return err;

        print("Configure measurement");
        #mc = MeasurementConfig();
        mc.measureMode = MeasureModeOneShot;
        mc.triggerMode = TriggerModeOff;
        mc.triggerConfig = TriggerConfigNone;
        mc.triggerChannel = 0;
        mc.triggerLevel = 0;
        mc.triggerDelay = 0;
        mc.sampleRate = 4000;
        mc.numSamples0 = 1;
        mc.numSamples1 = 0;
        mc.measurePeriod = 0;
        err = bmeasure.setMeasurementConfig(mc);
        if(err):
                return err;

        print("Run single measurement");
        (err, data) = bmeasure.measure();
        if(err):
                return err;

        print("DataBlock: from: %d numChannels: %d numSamples: %d" % (data.source, data.numChannels,
      data.numSamples));
        for c in range(0, data.numChannels):
                print("Chan:", c, data.data[c]);

        return err;


def main():
        err = test1();
        if(err):
                print("Error:", err.getErrorNo(), err.getString());
                return 1;

        print("Complete");

        return 0;

if __name__ == "__main__":
        main();
```

Simple example to show operating the relays in Python

```
#!/usr/bin/python3

import sys
import time
import getopt
from threading import Thread
from bmeasure import *

# Function to set the relays on/off
def test1():
        bmeasure = BMeasureUnit(True);

        print("Find BMeasure units");
        (err, devices) = BMeasureUnit.findDevicesUsb();
        if(err):
                return err;

        if(devices.number() == 0):
                return err.set(1, "No USB BMeasure units found\n");

        print("Found", len(devices));
        device = devices[0].device;

        print("Start Communications Task");
        bmeasure.start();

        print("Connect");
        err = bmeasure.connect(device);
        if(err):
                return err;

        print("Get Info");
        (err, info) = bmeasure.getInformation();
        if(err):
                return err;

        print("NumChannels: ", info.numChannels);

        # Toggle relay1
        state = 0;
        for i in range(0, 6):
                if(state):
                        state = 0;
                else:
                        state = 1;
                print("Set relay 0: %d" % (state));
                err = bmeasure.setRelay(0, state);
                if(err):
                        return err;

                time.sleep(1);

        return err;


def main():
        if(0):
                err = find();
                if(err):
                        print("Error:", err.getErrorNo(), err.getString());
                        return 1;

        err = test1();
        if(err):
                print("Error:", err.getErrorNo(), err.getString());
                return 1;

        print("Complete");

        return 0;

if __name__ == "__main__":
        main();
```

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 BMeasureApi Namespace Reference

Classes

- class AwgConfig
- class BMeasure
- class BMeasureUnit
- class BMeasureUnit1
- class BMeasureUnitDevice
- class BMeasureUnits
- class BMeasureUnitsDataBlock
- class BoardConfig
- class ChannelConfig
- class CommsNet
- class CommsSerial
- class CommsUsb
- class ConfigItem
- class Configuration
- class DataBlock
- class DataFile
- class FileData
- class FileInfo
- class FilesysInfo
- class InfoBlock
- class Information
- class MeasurementConfig
- class NodeInfo
- class NodeStatus
- class Version

Typedefs

- typedef **BArray**< ChannelConfig > ChannelConfigs

Enumerations

- enum ErrorNum { ErrorSystem = 64, ErrorDataOverrun = 65 }
- enum NodeType { NodeTypeNone = 0, NodeTypeBMeasure1 = 1 }
- enum SecureMode { SecureModeOpen, SecureMoteRemote, SecureModeFull }
- enum Status {
  StatusNone = 0x00, StatusError = 0x01, StatusWarning = 0x02, StatusRun = 0x04,
  StatusDataOverrun = 0x08, StatusEnd0 = 0x10, StatusEnd1 = 0x20, StatusFpgaOverrun = 0x40 }
- enum Mode {
  ModeSleep = 0, ModeIdle = 1, ModeRun = 2, ModeRunProgram = 3,
  ModeInternal = 4, ModeDemo1 = 5 }
- enum BlockTypes { BlockTypeInfo = 0x424E4531, BlockTypeData = 0x424E4532 }
- enum ChannelType {
  ChannelTypeNone = 0, ChannelTypeAnalogueIn = 1, ChannelTypeAnalogueOut = 0x81, ChannelTypeDigitalIn = 2,
  ChannelTypeDigitalOut = 0x82 }
- enum SampleType {
  SampleTypeNone = 0, SampleTypeBool = 1, SampleTypeInt8 = 2, SampleTypeInt16 = 3,
  SampleTypeInt32 = 4, SampleTypeFloat32 = 5, SampleTypeFloat64 = 6 }
- enum SyncMode { SyncModeOff = 0, SyncModeMaster = 1, SyncModeSlave = 2 }
- enum MeasureMode { MeasureModeOff = 0, MeasureModeOneShot = 1, MeasureModeRepeat = 2, MeasureModeContinuous = 3 }
- enum TriggerMode { TriggerModeOff = 0, TriggerModePositive = 1, TriggerModeNegative = 2 }
- enum TriggerConfig { TriggerConfigNone = 0 }
- enum DigitalMode {
  DigitalModeInput = 0, DigitalModeOutput = 1, DigitalInOut = 2, DigitalModeSyncMaster = 3,
  DigitalModeSyncSlave = 4 }
- enum Waveform {
  WaveformNone, WaveformSine, WaveformSquare, WaveformTriangle,
  WaveformNoise, WaveformArbitrary }
- enum CalibrateMode { CalibrateModeNone, CalibrateModeOffsets }
- enum FileType { FileTypeNone, FileTypeFile, FileTypeDir }
- enum FilesysDeleteType { FilesysDeleteTypeNone, FilesysDeleteTypeData, FilesysDeleteTypeFormat }
- enum LogDataMode { LogDataModeNormal, LogDataModeDeleteOld }
- enum TdsDataType {
  TdsTypeVoid, TdsTypeI8, TdsTypeI16, TdsTypeI32,
  TdsTypeI64, TdsTypeU8, TdsTypeU16, TdsTypeU32,
  TdsTypeU64, TdsTypeSingleFloat, TdsTypeDoubleFloat, TdsTypeExtendedFloat,
  TdsTypeSingleFloatWithUnit =0x19, TdsTypeDoubleFloatWithUnit, TdsTypeExtendedFloatWithUnit,
  TdsTypeString =0x20,
  TdsTypeBoolean =0x21, TdsTypeTimeStamp =0x44, TdsTypeFixedPoint =0x4F, TdsTypeComplexSingleFloat =0x08000c,
  TdsTypeComplexDoubleFloat =0x10000d, TdsTypeDAQmxRawData =0xFFFFFFFF }

Functions

- const char ∗ channelTypeString (ChannelType type)
- const char ∗ sampleTypeString (SampleType type)
- **BFloat32** toFloat (**BUInt32** v)
- static int unitSort (BMeasureUnit1 ∗&u1, BMeasureUnit1 ∗&u2)
- const **BUInt32** TocMetaData (1<< 1)
- const **BUInt32** TocNewObjList (1<< 2)
- const **BUInt32** TocRawData (1<< 3)
- const **BUInt32** TocInterleavedData (1<< 5)
- const **BUInt32** TocBigEndian (1<< 6)
- const **BUInt32** TocDaqRawData (1<< 7)
- **BUInt32** round512 (**BUInt32** s)

Variables

- const **BUInt32** apiVersion = 0

6.1.1 Typedef Documentation

6.1.1.1 ChannelConfigs

```
typedef BArray<ChannelConfig> BMeasureApi::ChannelConfigs
```

6.1.2 Enumeration Type Documentation

6.1.2.1 BlockTypes

```
enum BMeasureApi::BlockTypes
```

Enumerator

BlockTypeInfo	
BlockTypeData	

6.1.2.2 CalibrateMode

```
enum BMeasureApi::CalibrateMode
```

Enumerator

CalibrateModeNone	
CalibrateModeOffsets	

6.1.2.3 ChannelType

```
enum BMeasureApi::ChannelType
```

Enumerator

ChannelTypeNone	
ChannelTypeAnalogueIn	
ChannelTypeAnalogueOut	
ChannelTypeDigitalIn	
ChannelTypeDigitalOut	

6.1.2.4 DigitalMode

```
enum BMeasureApi::DigitalMode
```

Enumerator

DigitalModeInput	
DigitalModeOutput	
DigitalInOut	
DigitalModeSyncMaster	
DigitalModeSyncSlave	

6.1.2.5 ErrorNum

```
enum BMeasureApi::ErrorNum
```

Enumerator

ErrorSystem	
ErrorDataOverrun	

6.1.2.6 FilesysDeleteType

```
enum BMeasureApi::FilesysDeleteType
```

Enumerator

FilesysDeleteTypeNone	
FilesysDeleteTypeData	
FilesysDeleteTypeFormat	

6.1.2.7 FileType

```
enum BMeasureApi::FileType
```

Enumerator

FileTypeNone	
FileTypeFile	
FileTypeDir	

6.1.2.8 LogDataMode

```
enum BMeasureApi::LogDataMode
```

Enumerator

LogDataModeNormal	
LogDataModeDeleteOld	

6.1.2.9 MeasureMode

```
enum BMeasureApi::MeasureMode
```

Enumerator

MeasureModeOff	
MeasureModeOneShot	
MeasureModeRepeat	
MeasureModeContinuous	

6.1.2.10 Mode

```
enum BMeasureApi::Mode
```

Enumerator

ModeSleep	
ModeIdle	

Enumerator

ModeRun	
ModeRunProgram	
ModeInternal	
ModeDemo1	

6.1.2.11 NodeType

```
enum BMeasureApi::NodeType
```

Enumerator

NodeTypeNone	
NodeTypeBMeasure1	

6.1.2.12 SampleType

```
enum BMeasureApi::SampleType
```

Enumerator

SampleTypeNone	
SampleTypeBool	
SampleTypeInt8	
SampleTypeInt16	
SampleTypeInt32	
SampleTypeFloat32	
SampleTypeFloat64	

6.1.2.13 SecureMode

```
enum BMeasureApi::SecureMode
```

Enumerator

SecureModeOpen	
SecureMoteRemote	
SecureModeFull	

6.1.2.14 Status

```
enum BMeasureApi::Status
```

Enumerator

StatusNone	
StatusError	
StatusWarning	
StatusRun	
StatusDataOverrun	
StatusEnd0	
StatusEnd1	
StatusFpgaOverrun	

6.1.2.15 SyncMode

```
enum BMeasureApi::SyncMode
```

Enumerator

SyncModeOff	
SyncModeMaster	
SyncModeSlave	

6.1.2.16 TdsDataType

```
enum BMeasureApi::TdsDataType
```

Enumerator

TdsTypeVoid	
TdsTypeI8	
TdsTypeI16	
TdsTypeI32	
TdsTypeI64	
TdsTypeU8	
TdsTypeU16	
TdsTypeU32	
TdsTypeU64	
TdsTypeSingleFloat	

Enumerator

TdsTypeDoubleFloat	
TdsTypeExtendedFloat	
TdsTypeSingleFloatWithUnit	
TdsTypeDoubleFloatWithUnit	
TdsTypeExtendedFloatWithUnit	
TdsTypeString	
TdsTypeBoolean	
TdsTypeTimeStamp	
TdsTypeFixedPoint	
TdsTypeComplexSingleFloat	
TdsTypeComplexDoubleFloat	
TdsTypeDAQmxRawData	

6.1.2.17 TriggerConfig

```
enum BMeasureApi::TriggerConfig
```

Enumerator

TriggerConfigNone	

6.1.2.18 TriggerMode

```
enum BMeasureApi::TriggerMode
```

Enumerator

TriggerModeOff	
TriggerModePositive	
TriggerModeNegative	

6.1.2.19 Waveform

```
enum BMeasureApi::Waveform
```

Enumerator

WaveformNone	

Enumerator

WaveformSine	
WaveformSquare	
WaveformTriangle	
WaveformNoise	
WaveformArbitrary	

6.1.3 Function Documentation

6.1.3.1 channelTypeString()

```
const char * BMeasureApi::channelTypeString (
            ChannelType type )
```

6.1.3.2 round512()

```
 BUInt32 BMeasureApi::round512 (
             BUInt32 s )
```

6.1.3.3 sampleTypeString()

```
const char * BMeasureApi::sampleTypeString (
            SampleType type )
```

6.1.3.4 TocBigEndian()

```
const  BUInt32 BMeasureApi::TocBigEndian (
            1<< 6 )
```

6.1.3.5 TocDaqRawData()

```
const  BUInt32 BMeasureApi::TocDaqRawData (
            1<< 7 )
```

6.1.3.6 TocInterleavedData()

```
const  BUInt32 BMeasureApi::TocInterleavedData (
             1<< 5 )
```

6.1.3.7 TocMetaData()

```
const  BUInt32 BMeasureApi::TocMetaData (
             1<< 1 )
```

6.1.3.8 TocNewObjList()

```
const  BUInt32 BMeasureApi::TocNewObjList (
             1<< 2 )
```

6.1.3.9 TocRawData()

```
const  BUInt32 BMeasureApi::TocRawData (
             1<< 3 )
```

6.1.3.10 toFloat()

```
 BFloat32 BMeasureApi::toFloat (
              BUInt32 v )  [inline]
```

6.1.3.11 unitSort()

```
static int BMeasureApi::unitSort (
             BMeasureUnit1 *& u1,
             BMeasureUnit1 *& u2 )  [static]
```

6.1.4 Variable Documentation

6.1.4.1 apiVersion

```
const  BUInt32 BMeasureApi::apiVersion = 0
```

Chapter 7

Class Documentation

7.1 BMeasureApi::AwgConfig Class Reference

```
#include <BMeasureD.h>
```

Static Public Member Functions

- static const **BObjMember** ∗ getMembers ()

Public Attributes

- Waveform waveform

  The waveform.
- **BFloat32** frequency

  The frequency.
- **BFloat32** amplitude

  The peak amplitude in Volts.
- **BFloat32** offset

  The DC offset in volts.
- **BFloat32** duty

  The Duty cycle in %.

7.1.1 Member Function Documentation

7.1.1.1 getMembers()

```
const BObjMember * BMeasureApi::AwgConfig::getMembers ( ) [static]
```

7.1.2 Member Data Documentation

7.1.2.1 amplitude

`BFloat32 BMeasureApi::AwgConfig::amplitude`

The peak amplitude in Volts.

7.1.2.2 duty

`BFloat32 BMeasureApi::AwgConfig::duty`

The Duty cycle in %.

7.1.2.3 frequency

`BFloat32 BMeasureApi::AwgConfig::frequency`

The frequency.

7.1.2.4 offset

`BFloat32 BMeasureApi::AwgConfig::offset`

The DC offset in volts.

7.1.2.5 waveform

`Waveform BMeasureApi::AwgConfig::waveform`

The waveform.

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

- BMeasureD.h
- BMeasureD.cpp

7.2 BMdns Class Reference

```
#include <BMdns.h>
```

Public Member Functions

- BMdns ()
- ∼BMdns ()
- **BError** init ()
- **BError** findServices (**BString** service, **BUInt32** timeoutMs, **BList**< BMdnsService > &services)

Private Attributes

- **BSocket** osocket
- **BUInt32** otransactionId

7.2.1 Constructor & Destructor Documentation

7.2.1.1 BMdns()

```
BMdns::BMdns ( )
```

7.2.1.2 ∼BMdns()

```
BMdns::∼BMdns ( )
```

7.2.2 Member Function Documentation

7.2.2.1 findServices()

```
 BError BMdns::findServices (
             BString service,
             BUInt32 timeoutMs,
             BList< BMdnsService > & services )
```

Unicast response, class IN

7.2.2.2 init()

```
BError BMdns::init ( )
```

7.2.3 Member Data Documentation

7.2.3.1 osocket

```
BSocket BMdns::osocket  [private]
```

7.2.3.2 otransactionId

```
BUInt32 BMdns::otransactionId  [private]
```

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

- BMdns.h
- BMdns.cpp

7.3 BMdnsService Class Reference

```
#include <BMdns.h>
```

Public Attributes

- **BString** name
- **BSocketAddressINET** address
- **BString** hostname
- BStringList extra

7.3.1 Member Data Documentation

7.3.1.1 address

```
BSocketAddressINET BMdnsService::address
```

7.3.1.2 extra

```
BStringList BMdnsService::extra
```

7.3.1.3 hostname

```
BString BMdnsService::hostname
```

7.3.1.4 name

```
BString BMdnsService::name
```

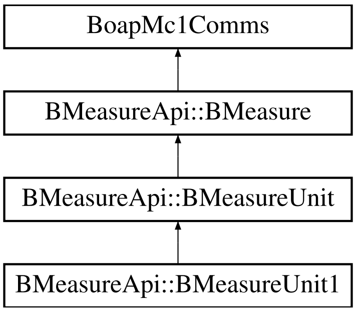
The documentation for this class was generated from the following file:

- BMdns.h

7.4 BMeasureApi::BMeasure Class Reference

```
#include <BMeasureB.h>
```

Inheritance diagram for BMeasureApi::BMeasure:



Public Member Functions

- BMeasure (**Bool** threaded=0, **BUInt** reqSize=512)
- **BError** getNodeInfo (NodeInfo &nodeInfo)

  Get node information.
- void factoryReset (const **BInt32** &bootLoader, const **BInt32** &resetConfig)

  Factory reset.
- **BError** getStatus (NodeStatus &nodeStatus)

  Get the node status.
- void sendStatus (const NodeStatus &nodeStatus)

  Sends the current status.
- void sendTime (const **BTimeUs** &time)

  Sends the current time.
- **BError** setSecureMode (const **BUInt64** &key, const SecureMode &secureMode)

  Set the security mode.
- **BError** login (const **BUInt64** &key, const **BString** &user, const **BString** &password)

  Provides user/password information for secure connection.
- **BError** setMode (const Mode &mode)

  Set the current operational mode.
- **BError** getInformation (Information &info)
- **BError** getInfoBlock (InfoBlock &infoBlock)
- **BError** getChannelConfig (const **BUInt32** &channelNumber, ChannelConfig &channelConfig)
- **BError** setChannelConfig (const **BUInt32** &channelNumber, const ChannelConfig &channelConfig)
- **BError** setChannelConfigFull (const **BUInt64** &key, const **BUInt32** &channelNumber, const ChannelConfig &channelConfig)
- **BError** getConfig (Configuration &config)

  Should we have this, not generic for different instruments ?
- **BError** setConfig (const Configuration &config)

  Should we have this, not generic for different instruments ?
- **BError** getMeasurementConfig (MeasurementConfig &measurementConfig)

  Get measurement config.
- **BError** setMeasurementConfig (const MeasurementConfig &measurementConfig)

  Set measurement config.
- void sendInfo (const InfoBlock &infoBlock)

  Sends an info block.
- void sendData (const DataBlock &dataBlock)

  Sends a data block.
- **BError** measure (DataBlock &dataBlock)

  Performs a single measurement.
- **BError** getAwgConfig (AwgConfig &awgConfig)

  Get AWG Configuration.
- **BError** setAwgConfig (const AwgConfig &awgConfig)

  Configure AWG.
- **BError** setAwgWaveform (const DataBlock &dataBlock)

  Configure AWG Arbitrary waveform.
- **BError** setDigital (const **BUInt32** &bits)

  Set digital bits.
- **BError** getDigital (**BUInt32** &bits)

  Get digital bits.
- **BError** setRelay (const **BUInt32** &relayNum, const **BInt32** &state)

  Set relay.

- **BError** getSwitch (const **BUInt32** &switchNum, **BInt32** &state)

  Get digital bits.
- **BError** filesysInfo (const **BString** &path, FilesysInfo &filesysInfo)
- **BError** filesysDelete (const **BString** &path, const FilesysDeleteType &deleteType)
- **BError** fileList (const **BString** &path, const **BUInt32** &pos, FileInfo &fileInfo)
- **BError** fileOpen (const **BString** &name, const **BString** &mode, **BUInt32** &handle)
- **BError** fileRead (const **BUInt32** &handle, const **BUInt32** &pos, const **BUInt32** &len, FileData & **data**)
- **BError** fileWrite (const **BUInt32** &handle, const **BUInt32** &pos, const FileData & **data**)
- **BError** fileClose (const **BUInt32** &handle)
- **BError** fileDelete (const **BString** &name)
- **BError** getBoardConfig (BoardConfig &config)

  Get the boards configuration.
- **BError** setBoardConfig (const BoardConfig &config)

  Sets the boards configuration.
- **BError** runBoardTest (const **BString** &test, **BString** &result)

  Runs the given board test.
- **BError** calibrate (const CalibrateMode &calibMode)

  Calibrate items.
- void sendDebugMessage (const **BUInt32** &source, const **BString** &message)
- **BError** processRequest (**BTimeout** timeoutUs= **BTimeoutForever**)
- virtual **BError** getNodeInfoServe (NodeInfo &nodeInfo)
- virtual void factoryResetServe (const **BInt32** &bootLoader, const **BInt32** &resetConfig)
- virtual **BError** getStatusServe (NodeStatus &nodeStatus)
- virtual void sendStatusServe (const NodeStatus &nodeStatus)
- virtual void sendTimeServe (const **BTimeUs** &time)
- virtual **BError** setSecureModeServe (const **BUInt64** &key, const SecureMode &secureMode)
- virtual **BError** loginServe (const **BUInt64** &key, const **BString** &user, const **BString** &password)
- virtual **BError** setModeServe (const Mode &mode)
- virtual **BError** getInformationServe (Information &info)
- virtual **BError** getInfoBlockServe (InfoBlock &infoBlock)
- virtual **BError** getChannelConfigServe (const **BUInt32** &channelNumber, ChannelConfig &channelConfig)
- virtual **BError** setChannelConfigServe (const **BUInt32** &channelNumber, const ChannelConfig &channel←Config)
- virtual **BError** setChannelConfigFullServe (const **BUInt64** &key, const **BUInt32** &channelNumber, const ChannelConfig &channelConfig)
- virtual **BError** getConfigServe (Configuration &config)
- virtual **BError** setConfigServe (const Configuration &config)
- virtual **BError** getMeasurementConfigServe (MeasurementConfig &measurementConfig)
- virtual **BError** setMeasurementConfigServe (const MeasurementConfig &measurementConfig)
- virtual void sendInfoServe (const InfoBlock &infoBlock)
- virtual void sendDataServe (const DataBlock &dataBlock)
- virtual **BError** measureServe (DataBlock &dataBlock)
- virtual **BError** getAwgConfigServe (AwgConfig &awgConfig)
- virtual **BError** setAwgConfigServe (const AwgConfig &awgConfig)
- virtual **BError** setAwgWaveformServe (const DataBlock &dataBlock)
- virtual **BError** setDigitalServe (const **BUInt32** &bits)
- virtual **BError** getDigitalServe (**BUInt32** &bits)
- virtual **BError** setRelayServe (const **BUInt32** &relayNum, const **BInt32** &state)
- virtual **BError** getSwitchServe (const **BUInt32** &switchNum, **BInt32** &state)
- virtual **BError** filesysInfoServe (const **BString** &path, FilesysInfo &filesysInfo)
- virtual **BError** filesysDeleteServe (const **BString** &path, const FilesysDeleteType &deleteType)
- virtual **BError** fileListServe (const **BString** &path, const **BUInt32** &pos, FileInfo &fileInfo)
- virtual **BError** fileOpenServe (const **BString** &name, const **BString** &mode, **BUInt32** &handle)

- virtual **BError** fileReadServe (const **BUInt32** &handle, const **BUInt32** &pos, const **BUInt32** &len, FileData & **data**)
- virtual **BError** fileWriteServe (const **BUInt32** &handle, const **BUInt32** &pos, const FileData & **data**)
- virtual **BError** fileCloseServe (const **BUInt32** &handle)
- virtual **BError** fileDeleteServe (const **BString** &name)
- virtual **BError** getBoardConfigServe (BoardConfig &config)
- virtual **BError** setBoardConfigServe (const BoardConfig &config)
- virtual **BError** runBoardTestServe (const **BString** &test, **BString** &result)
- virtual **BError** calibrateServe (const CalibrateMode &calibMode)
- virtual void sendDebugMessageServe (const **BUInt32** &source, const **BString** &message)

Additional Inherited Members

7.4.1 Constructor & Destructor Documentation

7.4.1.1 BMeasure()

```
BMeasureApi::BMeasure::BMeasure (
            Bool threaded = 0,
            BUInt reqSize = 512 )
```

7.4.2 Member Function Documentation

7.4.2.1 calibrate()

```
BError BMeasureApi::BMeasure::calibrate (
            const CalibrateMode & calibMode )
```

Calibrate items.

7.4.2.2 calibrateServe()

```
BError BMeasureApi::BMeasure::calibrateServe (
            const CalibrateMode & calibMode )  [virtual]
```

7.4.2.3 factoryReset()

```
void BMeasureApi::BMeasure::factoryReset (
            const  BInt32 & bootLoader,
            const  BInt32 & resetConfig )
```

Factory reset.

7.4.2.4 factoryResetServe()

```
void BMeasureApi::BMeasure::factoryResetServe (
            const  BInt32 & bootLoader,
            const  BInt32 & resetConfig )  [virtual]
```

7.4.2.5 fileClose()

```
 BError BMeasureApi::BMeasure::fileClose (
            const  BUInt32 & handle )
```

7.4.2.6 fileCloseServe()

```
 BError BMeasureApi::BMeasure::fileCloseServe (
            const  BUInt32 & handle )  [virtual]
```

7.4.2.7 fileDelete()

```
 BError BMeasureApi::BMeasure::fileDelete (
            const  BString & name )
```

7.4.2.8 fileDeleteServe()

```
 BError BMeasureApi::BMeasure::fileDeleteServe (
            const  BString & name )  [virtual]
```

7.4.2.9 fileList()

```
BError BMeasureApi::BMeasure::fileList (
            const BString & path,
            const BUInt32 & pos,
            FileInfo & fileInfo )
```

7.4.2.10 fileListServe()

```
BError BMeasureApi::BMeasure::fileListServe (
            const BString & path,
            const BUInt32 & pos,
            FileInfo & fileInfo ) [virtual]
```

7.4.2.11 fileOpen()

```
BError BMeasureApi::BMeasure::fileOpen (
            const BString & name,
            const BString & mode,
            BUInt32 & handle )
```

7.4.2.12 fileOpenServe()

```
BError BMeasureApi::BMeasure::fileOpenServe (
            const BString & name,
            const BString & mode,
            BUInt32 & handle ) [virtual]
```

7.4.2.13 fileRead()

```
BError BMeasureApi::BMeasure::fileRead (
            const BUInt32 & handle,
            const BUInt32 & pos,
            const BUInt32 & len,
            FileData & data )
```

7.4.2.14 fileReadServe()

```
BError BMeasureApi::BMeasure::fileReadServe (
            const  BUInt32 & handle,
            const  BUInt32 & pos,
            const  BUInt32 & len,
            FileData & data )  [virtual]
```

7.4.2.15 filesysDelete()

```
BError BMeasureApi::BMeasure::filesysDelete (
            const  BString & path,
            const FilesysDeleteType & deleteType )
```

7.4.2.16 filesysDeleteServe()

```
BError BMeasureApi::BMeasure::filesysDeleteServe (
            const  BString & path,
            const FilesysDeleteType & deleteType )  [virtual]
```

7.4.2.17 filesysInfo()

```
BError BMeasureApi::BMeasure::filesysInfo (
            const  BString & path,
            FilesysInfo & filesysInfo )
```

7.4.2.18 filesysInfoServe()

```
BError BMeasureApi::BMeasure::filesysInfoServe (
            const  BString & path,
            FilesysInfo & filesysInfo )  [virtual]
```

7.4.2.19 fileWrite()

```
BError BMeasureApi::BMeasure::fileWrite (
            const  BUInt32 & handle,
            const  BUInt32 & pos,
            const FileData & data )
```

7.4.2.20 fileWriteServe()

```
BError BMeasureApi::BMeasure::fileWriteServe (
            const  BUInt32 & handle,
            const  BUInt32 & pos,
            const FileData & data )  [virtual]
```

7.4.2.21 getAwgConfig()

```
BError BMeasureApi::BMeasure::getAwgConfig (
            AwgConfig & awgConfig )
```

Get AWG Configuration.

7.4.2.22 getAwgConfigServe()

```
BError BMeasureApi::BMeasure::getAwgConfigServe (
            AwgConfig & awgConfig )  [virtual]
```

7.4.2.23 getBoardConfig()

```
BError BMeasureApi::BMeasure::getBoardConfig (
            BoardConfig & config )
```

Get the boards configuration.

7.4.2.24 getBoardConfigServe()

```
BError BMeasureApi::BMeasure::getBoardConfigServe (
            BoardConfig & config )  [virtual]
```

7.4.2.25 getChannelConfig()

```
BError BMeasureApi::BMeasure::getChannelConfig (
            const  BUInt32 & channelNumber,
            ChannelConfig & channelConfig )
```

7.4.2.26 getChannelConfigServe()

```
BError BMeasureApi::BMeasure::getChannelConfigServe (
            const  BUInt32 & channelNumber,
            ChannelConfig & channelConfig )  [virtual]
```

7.4.2.27 getConfig()

```
BError BMeasureApi::BMeasure::getConfig (
            Configuration & config )
```

Should we have this, not generic for different instruments ?

7.4.2.28 getConfigServe()

```
BError BMeasureApi::BMeasure::getConfigServe (
            Configuration & config )  [virtual]
```

7.4.2.29 getDigital()

```
BError BMeasureApi::BMeasure::getDigital (
             BUInt32 & bits )
```

Get digital bits.

7.4.2.30 getDigitalServe()

```
BError BMeasureApi::BMeasure::getDigitalServe (
             BUInt32 & bits )  [virtual]
```

7.4.2.31 getInfoBlock()

```
BError BMeasureApi::BMeasure::getInfoBlock (
            InfoBlock & infoBlock )
```

7.4.2.32 getInfoBlockServe()

```
BError BMeasureApi::BMeasure::getInfoBlockServe (
            InfoBlock & infoBlock )  [virtual]
```

7.4.2.33 getInformation()

```
BError BMeasureApi::BMeasure::getInformation (
            Information & info )
```

7.4.2.34 getInformationServe()

```
BError BMeasureApi::BMeasure::getInformationServe (
            Information & info )  [virtual]
```

7.4.2.35 getMeasurementConfig()

```
BError BMeasureApi::BMeasure::getMeasurementConfig (
            MeasurementConfig & measurementConfig )
```

Get measurement config.

7.4.2.36 getMeasurementConfigServe()

```
BError BMeasureApi::BMeasure::getMeasurementConfigServe (
            MeasurementConfig & measurementConfig )  [virtual]
```

7.4.2.37 getNodeInfo()

```
BError BMeasureApi::BMeasure::getNodeInfo (
            NodeInfo & nodeInfo )
```

Get node information.

7.4.2.38 getNodeInfoServe()

```
BError BMeasureApi::BMeasure::getNodeInfoServe (
            NodeInfo & nodeInfo ) [virtual]
```

7.4.2.39 getStatus()

```
BError BMeasureApi::BMeasure::getStatus (
            NodeStatus & nodeStatus )
```

Get the node status.

7.4.2.40 getStatusServe()

```
BError BMeasureApi::BMeasure::getStatusServe (
            NodeStatus & nodeStatus ) [virtual]
```

7.4.2.41 getSwitch()

```
BError BMeasureApi::BMeasure::getSwitch (
            const BUInt32 & switchNum,
             BInt32 & state )
```

Get digital bits.

7.4.2.42 getSwitchServe()

```
BError BMeasureApi::BMeasure::getSwitchServe (
            const BUInt32 & switchNum,
             BInt32 & state ) [virtual]
```

7.4.2.43 login()

```
BError BMeasureApi::BMeasure::login (
            const BUInt64 & key,
            const BString & user,
            const BString & password )
```

Provides user/password information for secure connection.

7.4.2.44 loginServe()

```
BError BMeasureApi::BMeasure::loginServe (
            const  BUInt64 & key,
            const  BString & user,
            const  BString & password )  [virtual]
```

7.4.2.45 measure()

```
BError BMeasureApi::BMeasure::measure (
            DataBlock & dataBlock )
```

Performs a single measurement.

7.4.2.46 measureServe()

```
BError BMeasureApi::BMeasure::measureServe (
            DataBlock & dataBlock )  [virtual]
```

7.4.2.47 processRequest()

```
BError BMeasureApi::BMeasure::processRequest (
             BTimeout timeoutUs =  BTimeoutForever )  [virtual]
```

Reimplemented from **BoapMc1Comms**.

7.4.2.48 runBoardTest()

```
BError BMeasureApi::BMeasure::runBoardTest (
            const  BString & test,
             BString & result )
```

Runs the given board test.

7.4.2.49 runBoardTestServe()

```
BError BMeasureApi::BMeasure::runBoardTestServe (
            const  BString & test,
             BString & result )  [virtual]
```

7.4.2.50 sendData()

```
void BMeasureApi::BMeasure::sendData (
            const DataBlock & dataBlock )
```

Sends a data block.

7.4.2.51 sendDataServe()

```
void BMeasureApi::BMeasure::sendDataServe (
            const DataBlock & dataBlock )  [virtual]
```

Reimplemented in BMeasureApi::BMeasureUnit.

7.4.2.52 sendDebugMessage()

```
void BMeasureApi::BMeasure::sendDebugMessage (
            const  BUInt32 & source,
            const  BString & message )
```

7.4.2.53 sendDebugMessageServe()

```
void BMeasureApi::BMeasure::sendDebugMessageServe (
            const  BUInt32 & source,
            const  BString & message )  [virtual]
```

Reimplemented in BMeasureApi::BMeasureUnit1.

7.4.2.54 sendInfo()

```
void BMeasureApi::BMeasure::sendInfo (
            const InfoBlock & infoBlock )
```

Sends an info block.

7.4.2.55 sendInfoServe()

```
void BMeasureApi::BMeasure::sendInfoServe (
            const InfoBlock & infoBlock )  [virtual]
```

7.4.2.56 sendStatus()

```
void BMeasureApi::BMeasure::sendStatus (
             const NodeStatus & nodeStatus )
```

Sends the current status.

7.4.2.57 sendStatusServe()

```
void BMeasureApi::BMeasure::sendStatusServe (
             const NodeStatus & nodeStatus )  [virtual]
```

7.4.2.58 sendTime()

```
void BMeasureApi::BMeasure::sendTime (
             const  BTimeUs & time )
```

Sends the current time.

7.4.2.59 sendTimeServe()

```
void BMeasureApi::BMeasure::sendTimeServe (
             const  BTimeUs & time )  [virtual]
```

7.4.2.60 setAwgConfig()

```
 BError BMeasureApi::BMeasure::setAwgConfig (
             const AwgConfig & awgConfig )
```

Configure AWG.

7.4.2.61 setAwgConfigServe()

```
 BError BMeasureApi::BMeasure::setAwgConfigServe (
             const AwgConfig & awgConfig )  [virtual]
```

7.4.2.62 setAwgWaveform()

```
BError BMeasureApi::BMeasure::setAwgWaveform (
            const DataBlock & dataBlock )
```

Configure AWG Arbitrary waveform.

7.4.2.63 setAwgWaveformServe()

```
BError BMeasureApi::BMeasure::setAwgWaveformServe (
            const DataBlock & dataBlock )  [virtual]
```

7.4.2.64 setBoardConfig()

```
BError BMeasureApi::BMeasure::setBoardConfig (
            const BoardConfig & config )
```

Sets the boards configuration.

7.4.2.65 setBoardConfigServe()

```
BError BMeasureApi::BMeasure::setBoardConfigServe (
            const BoardConfig & config )  [virtual]
```

7.4.2.66 setChannelConfig()

```
BError BMeasureApi::BMeasure::setChannelConfig (
            const  BUInt32 & channelNumber,
            const ChannelConfig & channelConfig )
```

7.4.2.67 setChannelConfigFull()

```
BError BMeasureApi::BMeasure::setChannelConfigFull (
            const  BUInt64 & key,
            const  BUInt32 & channelNumber,
            const ChannelConfig & channelConfig )
```

7.4.2.68 setChannelConfigFullServe()

```
BError BMeasureApi::BMeasure::setChannelConfigFullServe (
            const  BUInt64 & key,
            const  BUInt32 & channelNumber,
            const ChannelConfig & channelConfig )  [virtual]
```

7.4.2.69 setChannelConfigServe()

```
BError BMeasureApi::BMeasure::setChannelConfigServe (
            const  BUInt32 & channelNumber,
            const ChannelConfig & channelConfig )  [virtual]
```

7.4.2.70 setConfig()

```
BError BMeasureApi::BMeasure::setConfig (
            const Configuration & config )
```

Should we have this, not generic for different instruments ?

7.4.2.71 setConfigServe()

```
BError BMeasureApi::BMeasure::setConfigServe (
            const Configuration & config )  [virtual]
```

7.4.2.72 setDigital()

```
BError BMeasureApi::BMeasure::setDigital (
            const  BUInt32 & bits )
```

Set digital bits.

7.4.2.73 setDigitalServe()

```
BError BMeasureApi::BMeasure::setDigitalServe (
            const  BUInt32 & bits )  [virtual]
```

7.4.2.74 setMeasurementConfig()

```
BError BMeasureApi::BMeasure::setMeasurementConfig (
            const MeasurementConfig & measurementConfig )
```

Set measurement config.

7.4.2.75 setMeasurementConfigServe()

```
BError BMeasureApi::BMeasure::setMeasurementConfigServe (
            const MeasurementConfig & measurementConfig )  [virtual]
```

7.4.2.76 setMode()

```
BError BMeasureApi::BMeasure::setMode (
            const Mode & mode )
```

Set the current operational mode.

7.4.2.77 setModeServe()

```
BError BMeasureApi::BMeasure::setModeServe (
            const Mode & mode )  [virtual]
```

7.4.2.78 setRelay()

```
BError BMeasureApi::BMeasure::setRelay (
            const  BUInt32 & relayNum,
            const  BInt32 & state )
```

Set relay.

7.4.2.79 setRelayServe()

```
BError BMeasureApi::BMeasure::setRelayServe (
            const  BUInt32 & relayNum,
            const  BInt32 & state )  [virtual]
```

7.4.2.80 setSecureMode()

```
BError BMeasureApi::BMeasure::setSecureMode (
            const BUInt64 & key,
            const SecureMode & secureMode )
```

Set the security mode.

7.4.2.81 setSecureModeServe()

```
BError BMeasureApi::BMeasure::setSecureModeServe (
            const BUInt64 & key,
            const SecureMode & secureMode ) [virtual]
```

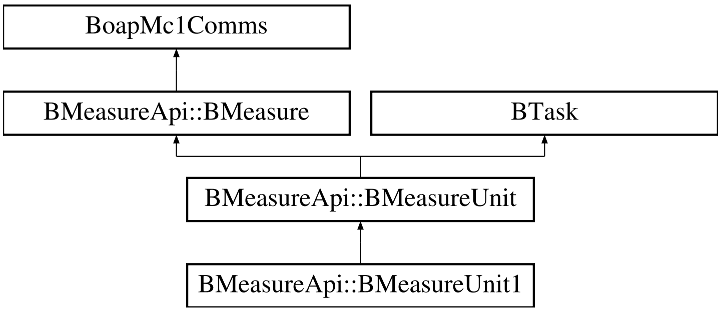
The documentation for this class was generated from the following files:

- BMeasureB.h
- BMeasureB.cpp

7.5 BMeasureApi::BMeasureUnit Class Reference

`#include <BMeasureUnit.h>`

Inheritance diagram for BMeasureApi::BMeasureUnit:



Public Member Functions

- BMeasureUnit (**Bool** threaded=0, **BUInt** reqSize=1024)
- virtual ∼BMeasureUnit ()
- **BError** connect (**BString** device)

  Connect to a device.
- void disconnect ()
- **BString** device ()
- **BString** serialNumber ()
- **BString** info ()
- **BUInt** numChannels ()

  The number of channels of data.
- void run ()

  Threaded run mode.
- virtual void disconnected ()
- virtual void sendDataServe (const DataBlock &dataBlock)
- virtual void sendDataServe1 (const DataBlock &dataBlock)
- virtual **BError** setMeasurementConfig (const MeasurementConfig &configMeasurement)
- virtual **BError** setChannelConfig (const **BUInt8** &channelNumber, const ChannelConfig &channelConfig)

Static Public Member Functions

- static **BError** findDevices (**BList**< BMeasureUnitDevice > &devices)

  Find available devices.
- static **BError** findDevicesUsb (**BList**< BMeasureUnitDevice > &devices)

  Find available devices on USB bus.
- static **BError** findDevicesNetwork (**BList**< BMeasureUnitDevice > &devices)

  Find available devices on Network.

Static Public Attributes

- static int blockNumChannels = 16
- static int blockNumSamples = 13

Protected Attributes

- **BString** odevice
- NodeInfo onodeInfo
- Information oinfo

  Instrument info.
- MeasurementConfig oconfigMeasurement
- **BArray**< ChannelConfig > ochannels
- DataBlock ∗ odataBlock
- **BUInt32** osequenceNext
- **BUInt32** osampleCount
- **BUInt32** oblockCount
- **Bool** odisconnecting

Additional Inherited Members

7.5.1 Constructor & Destructor Documentation

7.5.1.1 BMeasureUnit()

```
BMeasureApi::BMeasureUnit::BMeasureUnit (
              Bool threaded = 0,
              BUInt reqSize = 1024 )
```

7.5.1.2 ∼BMeasureUnit()

```
BMeasureApi::BMeasureUnit::∼BMeasureUnit ( )  [virtual]
```

7.5.2 Member Function Documentation

7.5.2.1 connect()

```
 BError BMeasureApi::BMeasureUnit::connect (
             BString device )
```

Connect to a device.

7.5.2.2 device()

```
 BString BMeasureApi::BMeasureUnit::device ( )
```

7.5.2.3 disconnect()

```
void BMeasureApi::BMeasureUnit::disconnect ( )
```

7.5.2.4 disconnected()

```
void BMeasureApi::BMeasureUnit::disconnected ( )  [virtual]
```

Reimplemented in BMeasureApi::BMeasureUnit1.

7.5.2.5 findDevices()

```
 BError BMeasureApi::BMeasureUnit::findDevices (
             BList< BMeasureUnitDevice > & devices )  [static]
```

Find available devices.

7.5.2.6 findDevicesNetwork()

```
 BError BMeasureApi::BMeasureUnit::findDevicesNetwork (
             BList< BMeasureUnitDevice > & devices )  [static]
```

Find available devices on Network.

7.5.2.7 findDevicesUsb()

```
BError BMeasureApi::BMeasureUnit::findDevicesUsb (
             BList< BMeasureUnitDevice > & devices )  [static]
```

Find available devices on USB bus.

7.5.2.8 info()

```
BString BMeasureApi::BMeasureUnit::info ( )
```

7.5.2.9 numChannels()

```
BUInt BMeasureApi::BMeasureUnit::numChannels ( )
```

The number of channels of data.

7.5.2.10 run()

```
void BMeasureApi::BMeasureUnit::run ( )  [virtual]
```

Threaded run mode.

Reimplemented from **BTask**.

7.5.2.11 sendDataServe()

```
void BMeasureApi::BMeasureUnit::sendDataServe (
             const DataBlock & dataBlock )  [virtual]
```

Reimplemented from BMeasureApi::BMeasure.

7.5.2.12 sendDataServe1()

```
void BMeasureApi::BMeasureUnit::sendDataServe1 (
             const DataBlock & dataBlock )  [virtual]
```

Reimplemented in BMeasureApi::BMeasureUnit1.

7.5.2.13 serialNumber()

```
BString BMeasureApi::BMeasureUnit::serialNumber ( )
```

7.5.2.14 setChannelConfig()

```
BError BMeasureApi::BMeasureUnit::setChannelConfig (
            const  BUInt8 & channelNumber,
            const ChannelConfig & channelConfig )  [virtual]
```

7.5.2.15 setMeasurementConfig()

```
BError BMeasureApi::BMeasureUnit::setMeasurementConfig (
            const MeasurementConfig & configMeasurement )  [virtual]
```

7.5.3 Member Data Documentation

7.5.3.1 blockNumChannels

```
int BMeasureApi::BMeasureUnit::blockNumChannels = 16  [static]
```

7.5.3.2 blockNumSamples

```
int BMeasureApi::BMeasureUnit::blockNumSamples = 13  [static]
```

7.5.3.3 oblockCount

```
BUInt32 BMeasureApi::BMeasureUnit::oblockCount  [protected]
```

7.5.3.4 ochannels

```
BArray<ChannelConfig> BMeasureApi::BMeasureUnit::ochannels  [protected]
```

7.5.3.5 oconfigMeasurement

```
MeasurementConfig BMeasureApi::BMeasureUnit::oconfigMeasurement  [protected]
```

7.5.3.6 odataBlock

```
DataBlock* BMeasureApi::BMeasureUnit::odataBlock  [protected]
```

7.5.3.7 odevice

```
BString BMeasureApi::BMeasureUnit::odevice  [protected]
```

7.5.3.8 odisconnecting

```
Bool BMeasureApi::BMeasureUnit::odisconnecting  [protected]
```

7.5.3.9 oinfo

```
Information BMeasureApi::BMeasureUnit::oinfo  [protected]
```

Instrument info.

7.5.3.10 onodeInfo

```
NodeInfo BMeasureApi::BMeasureUnit::onodeInfo  [protected]
```

7.5.3.11 osampleCount

```
BUInt32 BMeasureApi::BMeasureUnit::osampleCount  [protected]
```

7.5.3.12 osequenceNext

BUInt32 BMeasureApi::BMeasureUnit::osequenceNext [protected]

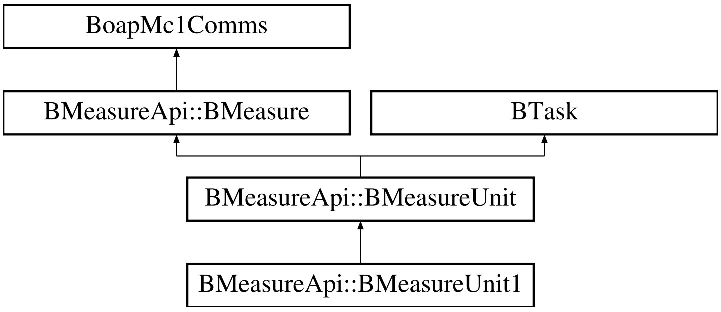
The documentation for this class was generated from the following files:

- BMeasureUnit.h
- BMeasureUnit.cpp

7.6 BMeasureApi::BMeasureUnit1 Class Reference

```
#include <BMeasureUnits.h>
```

Inheritance diagram for BMeasureApi::BMeasureUnit1:



Public Member Functions

- BMeasureUnit1 (BMeasureUnits &measureUnits, **BString** device, **Bool** threaded=0, **BUInt** reqSize=1024)
- **BString** serialNumber ()
- void setSerialNumber (**BString** serialNumber)
- void disconnected ()
- void sendDataServe1 (const DataBlock &dataBlock)
- void sendDebugMessageServe (const **BUInt32** &source, const **BString** &message)

Public Attributes

- BMeasureUnits & omeasureUnits
- **Bool** oenabled
- **Bool** oconnected
- **BUInt** oorder
- **BUInt** osource
- **BString** oserialNumber

Additional Inherited Members

7.6.1 Constructor & Destructor Documentation

7.6.1.1 BMeasureUnit1()

```
BMeasureApi::BMeasureUnit1::BMeasureUnit1 (
             BMeasureUnits & measureUnits,
              BString device,
              Bool threaded = 0,
              BUInt reqSize = 1024 )
```

7.6.2 Member Function Documentation

7.6.2.1 disconnected()

```
void BMeasureApi::BMeasureUnit1::disconnected ( )  [virtual]
```

Reimplemented from BMeasureApi::BMeasureUnit.

7.6.2.2 sendDataServe1()

```
void BMeasureApi::BMeasureUnit1::sendDataServe1 (
             const DataBlock & dataBlock )  [virtual]
```

Reimplemented from BMeasureApi::BMeasureUnit.

7.6.2.3 sendDebugMessageServe()

```
void BMeasureApi::BMeasureUnit1::sendDebugMessageServe (
             const  BUInt32 & source,
             const  BString & message )  [virtual]
```

Reimplemented from BMeasureApi::BMeasure.

7.6.2.4 serialNumber()

```
 BString BMeasureApi::BMeasureUnit1::serialNumber ( )
```

7.6.2.5 setSerialNumber()

```
void BMeasureApi::BMeasureUnit1::setSerialNumber (
              BString serialNumber )
```

7.6.3 Member Data Documentation

7.6.3.1 oconnected

```
Bool BMeasureApi::BMeasureUnit1::oconnected
```

7.6.3.2 oenabled

```
Bool BMeasureApi::BMeasureUnit1::oenabled
```

7.6.3.3 omeasureUnits

```
BMeasureUnits& BMeasureApi::BMeasureUnit1::omeasureUnits
```

7.6.3.4 oorder

```
BUInt BMeasureApi::BMeasureUnit1::oorder
```

7.6.3.5 oserialNumber

```
BString BMeasureApi::BMeasureUnit1::oserialNumber
```

7.6.3.6 osource

```
BUInt BMeasureApi::BMeasureUnit1::osource
```

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

- BMeasureUnits.h
- BMeasureUnits.cpp

7.7 BMeasureApi::BMeasureUnitDevice Class Reference

```
#include <BMeasureUnit.h>
```

Public Member Functions

- BMeasureUnitDevice (**BString** serialNumber="", **BString** device="")

Public Attributes

- **BString** serialNumber
- **BString** device

7.7.1 Constructor & Destructor Documentation

7.7.1.1 BMeasureUnitDevice()

```
BMeasureApi::BMeasureUnitDevice::BMeasureUnitDevice (
              BString serialNumber = "",
              BString device = "" )  [inline]
```

7.7.2 Member Data Documentation

7.7.2.1 device

```
BString BMeasureApi::BMeasureUnitDevice::device
```

7.7.2.2 serialNumber

```
BString BMeasureApi::BMeasureUnitDevice::serialNumber
```

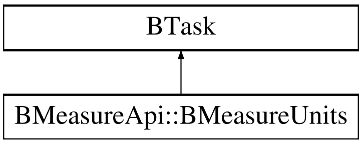
The documentation for this class was generated from the following file:

- BMeasureUnit.h

7.8 BMeasureApi::BMeasureUnits Class Reference

```
#include <BMeasureUnits.h>
```

Inheritance diagram for BMeasureApi::BMeasureUnits:



Public Member Functions

- BMeasureUnits (**Bool** threaded=0)
- virtual ∼BMeasureUnits ()
- void clear ()
- **BError** unitsFind ()
- **BError** unitAdd (**BString** serialNumber, **BString** device)
- **BError** unitDelete (**BString** device)
- **BUInt32** unitsNum ()
- **BUInt32** unitsConnectedNum ()
- BMeasureUnit1 & unit (**BUInt** u)
- BMeasureUnit1 & unitMaster ()
- **BError** unitsConnect ()
- **Bool** unitsConnected ()
- **BError** unitsDisconnect ()
- virtual void disconnected ()
- **BError** unitSetOrder (**BUInt** u, **BUInt** order, **Bool** move)
- **BError** unitSetEnabled (**BUInt** u, **Bool** enable)
- **BError** dataSetNumStreams (**BUInt** num)

  Set the number of data output channels.
- void dataClear ()
- **BUInt** dataAvailable (**BUInt** stream)
- **BError** dataWait (**BUInt** stream, **BTimeout** timeoutUs= **BTimeoutForever**)
- virtual void dataEvent (**BUInt** stream)
- DataBlock ∗ dataRead (**BUInt** stream)
- void dataDone (**BUInt** stream)
- void run ()

  Threaded run mode.
- void sendDataQueue (const DataBlock &dataBlock)
- void sendDataProcess ()
- void sendDataProcessTrigger ()
- void outputBlock (BMeasureUnitsDataBlock ∗block)
- virtual **BUInt** numChannels ()

  The number of channels of data.
- virtual **BError** setMode (const Mode &mode)

  Set the current operational mode.
- virtual **BError** getStatus (NodeStatus &nodeStatus)
- virtual void sendTime (const **BTimeUs** &time)

  Sends the current time.
- virtual **BError** getInformation (Information &info)

- virtual **BError** getInfoBlock (InfoBlock &infoBlock)
- virtual **BError** getChannelConfig (const **BUInt8** &channelNumber, ChannelConfig &channelConfig)
- virtual **BError** setChannelConfig (const **BUInt8** &channelNumber, const ChannelConfig &channelConfig)
- virtual **BError** getConfig (Configuration &config)

  Should we have this, not generic for different instruments ?
- virtual **BError** setConfig (const Configuration &config)

  Should we have this, not generic for different instruments ?
- virtual **BError** getMeasurementConfig (MeasurementConfig &measurement)

  Get measurement config.
- virtual **BError** setMeasurementConfig (const MeasurementConfig &measurement)

  Set measurement config.
- virtual **BError** getAwgConfig (AwgConfig &awgConfig)

  Get AWG Configuration.
- virtual **BError** setAwgConfig (const AwgConfig &awgConfig)

  Configure AWG.
- virtual void sendDataServe1 (const DataBlock &dataBlock)
- virtual void sendDebugMessage (**BUInt32** &source, **BString** &message)
- virtual void sendDebugMessageServe (const **BUInt32** &source, const **BString** &message)
- void debugPrint ()

Private Member Functions

- BMeasureUnitsDataBlock ∗ getFreeBlock (**BUInt** numSamples)

Private Attributes

- **BSemaphoreBool** oprocEnable

  Enable processing.
- **BSemaphoreBool** oprocRunning

  Processing is running.
- **BMutex** olockUnits
- **BList**< BMeasureUnit1 ∗ > ounits
- **BInt** ounitMaster
- **BUInt** onumConnected
- **BUInt** onumChannels
- **BUInt** odataStreamNum
- **BUInt32** ofill
- **BUInt** onumBlocks
- **BMutex** olockInput
- **BList**< BMeasureUnitsDataBlock ∗ > odataBlocksFree
- **BList**< BMeasureUnitsDataBlock ∗ > odataBlocksIn
- **BList**< BMeasureUnitsDataBlock ∗ > odataBlocksProcess
- **BCondInt** odataBlocksProcessNum
- **BMutex** olockOutput
- **BList**< BMeasureUnitsDataBlock ∗ > odataBlocksOut [2]
- **BCondInt** odataBlocksOutCount [2]
- MeasurementConfig olocalTrigger
- **Bool** otriggered
- **BUInt** ostartSample

Additional Inherited Members

7.8.1 Constructor & Destructor Documentation

7.8.1.1 BMeasureUnits()

```
BMeasureApi::BMeasureUnits::BMeasureUnits (
              Bool threaded = 0 )
```

7.8.1.2 ∼BMeasureUnits()

```
BMeasureApi::BMeasureUnits::∼BMeasureUnits ( )  [virtual]
```

7.8.2 Member Function Documentation

7.8.2.1 clear()

```
void BMeasureApi::BMeasureUnits::clear ( )
```

7.8.2.2 dataAvailable()

```
 BUInt BMeasureApi::BMeasureUnits::dataAvailable (
              BUInt stream )
```

7.8.2.3 dataClear()

```
void BMeasureApi::BMeasureUnits::dataClear ( )
```

7.8.2.4 dataDone()

```
void BMeasureApi::BMeasureUnits::dataDone (
              BUInt stream )
```

7.8.2.5 dataEvent()

```
void BMeasureApi::BMeasureUnits::dataEvent (
              BUInt stream )  [virtual]
```

7.8.2.6 dataRead()

```
DataBlock * BMeasureApi::BMeasureUnits::dataRead (
              BUInt stream )
```

7.8.2.7 dataSetNumStreams()

```
 BError BMeasureApi::BMeasureUnits::dataSetNumStreams (
              BUInt num )
```

Set the number of data output channels.

7.8.2.8 dataWait()

```
 BError BMeasureApi::BMeasureUnits::dataWait (
              BUInt stream,
              BTimeout timeoutUs =  BTimeoutForever )
```

7.8.2.9 debugPrint()

```
void BMeasureApi::BMeasureUnits::debugPrint ( )
```

7.8.2.10 disconnected()

```
void BMeasureApi::BMeasureUnits::disconnected ( )  [virtual]
```

7.8.2.11 getAwgConfig()

```
 BError BMeasureApi::BMeasureUnits::getAwgConfig (
             AwgConfig & awgConfig )  [virtual]
```

Get AWG Configuration.

7.8.2.12 getChannelConfig()

```
BError BMeasureApi::BMeasureUnits::getChannelConfig (
            const  BUInt8 & channelNumber,
            ChannelConfig & channelConfig )  [virtual]
```

7.8.2.13 getConfig()

```
BError BMeasureApi::BMeasureUnits::getConfig (
            Configuration & config )  [virtual]
```

Should we have this, not generic for different instruments ?

7.8.2.14 getFreeBlock()

```
BMeasureUnitsDataBlock * BMeasureApi::BMeasureUnits::getFreeBlock (
              BUInt numSamples )  [private]
```

7.8.2.15 getInfoBlock()

```
BError BMeasureApi::BMeasureUnits::getInfoBlock (
            InfoBlock & infoBlock )  [virtual]
```

7.8.2.16 getInformation()

```
BError BMeasureApi::BMeasureUnits::getInformation (
            Information & info )  [virtual]
```

7.8.2.17 getMeasurementConfig()

```
BError BMeasureApi::BMeasureUnits::getMeasurementConfig (
            MeasurementConfig & measurement )  [virtual]
```

Get measurement config.

7.8.2.18 getStatus()

```
BError BMeasureApi::BMeasureUnits::getStatus (
            NodeStatus & nodeStatus )  [virtual]
```

7.8.2.19 numChannels()

```
BUInt BMeasureApi::BMeasureUnits::numChannels ( )  [virtual]
```

The number of channels of data.

7.8.2.20 outputBlock()

```
void BMeasureApi::BMeasureUnits::outputBlock (
            BMeasureUnitsDataBlock * block )
```

7.8.2.21 run()

```
void BMeasureApi::BMeasureUnits::run ( )  [virtual]
```

Threaded run mode.

Reimplemented from **BTask**.

7.8.2.22 sendDataProcess()

```
void BMeasureApi::BMeasureUnits::sendDataProcess ( )
```

7.8.2.23 sendDataProcessTrigger()

```
void BMeasureApi::BMeasureUnits::sendDataProcessTrigger ( )
```

7.8.2.24 sendDataQueue()

```
void BMeasureApi::BMeasureUnits::sendDataQueue (
            const DataBlock & dataBlock )
```

7.8.2.25 sendDataServe1()

```
void BMeasureApi::BMeasureUnits::sendDataServe1 (
            const DataBlock & dataBlock )  [virtual]
```

7.8.2.26 sendDebugMessage()

```
void BMeasureApi::BMeasureUnits::sendDebugMessage (
             BUInt32 & source,
             BString & message )  [virtual]
```

7.8.2.27 sendDebugMessageServe()

```
void BMeasureApi::BMeasureUnits::sendDebugMessageServe (
            const  BUInt32 & source,
            const  BString & message )  [virtual]
```

7.8.2.28 sendTime()

```
void BMeasureApi::BMeasureUnits::sendTime (
            const  BTimeUs & time )  [virtual]
```

Sends the current time.

7.8.2.29 setAwgConfig()

```
 BError BMeasureApi::BMeasureUnits::setAwgConfig (
            const AwgConfig & awgConfig )  [virtual]
```

Configure AWG.

7.8.2.30 setChannelConfig()

```
 BError BMeasureApi::BMeasureUnits::setChannelConfig (
            const  BUInt8 & channelNumber,
            const ChannelConfig & channelConfig )  [virtual]
```

7.8.2.31 setConfig()

```
BError BMeasureApi::BMeasureUnits::setConfig (
            const Configuration & config )  [virtual]
```

Should we have this, not generic for different instruments ?

7.8.2.32 setMeasurementConfig()

```
BError BMeasureApi::BMeasureUnits::setMeasurementConfig (
            const MeasurementConfig & measurement )  [virtual]
```

Set measurement config.

7.8.2.33 setMode()

```
BError BMeasureApi::BMeasureUnits::setMode (
            const Mode & mode )  [virtual]
```

Set the current operational mode.

7.8.2.34 unit()

```
BMeasureUnit1 & BMeasureApi::BMeasureUnits::unit (
              BUInt u )
```

7.8.2.35 unitAdd()

```
BError BMeasureApi::BMeasureUnits::unitAdd (
             BString serialNumber,
             BString device )
```

7.8.2.36 unitDelete()

```
BError BMeasureApi::BMeasureUnits::unitDelete (
             BString device )
```

7.8.2.37 unitMaster()

```
BMeasureUnit1 & BMeasureApi::BMeasureUnits::unitMaster ( )
```

7.8.2.38 unitsConnect()

```
BError BMeasureApi::BMeasureUnits::unitsConnect ( )
```

7.8.2.39 unitsConnected()

```
Bool BMeasureApi::BMeasureUnits::unitsConnected ( )
```

7.8.2.40 unitsConnectedNum()

```
BUInt BMeasureApi::BMeasureUnits::unitsConnectedNum ( )
```

7.8.2.41 unitsDisconnect()

```
BError BMeasureApi::BMeasureUnits::unitsDisconnect ( )
```

7.8.2.42 unitSetEnabled()

```
BError BMeasureApi::BMeasureUnits::unitSetEnabled (
             BUInt u,
             Bool enable )
```

7.8.2.43 unitSetOrder()

```
BError BMeasureApi::BMeasureUnits::unitSetOrder (
             BUInt u,
             BUInt order,
             Bool move )
```

7.8.2.44 unitsFind()

```
BError BMeasureApi::BMeasureUnits::unitsFind ( )
```

7.8.2.45 unitsNum()

```
BUInt BMeasureApi::BMeasureUnits::unitsNum ( )
```

7.8.3 Member Data Documentation

7.8.3.1 odataBlocksFree

```
BList<BMeasureUnitsDataBlock*> BMeasureApi::BMeasureUnits::odataBlocksFree  [private]
```

7.8.3.2 odataBlocksIn

```
BList<BMeasureUnitsDataBlock*> BMeasureApi::BMeasureUnits::odataBlocksIn  [private]
```

7.8.3.3 odataBlocksOut

```
BList<BMeasureUnitsDataBlock*> BMeasureApi::BMeasureUnits::odataBlocksOut[2]  [private]
```

7.8.3.4 odataBlocksOutCount

```
BCondInt BMeasureApi::BMeasureUnits::odataBlocksOutCount[2]  [private]
```

7.8.3.5 odataBlocksProcess

```
BList<BMeasureUnitsDataBlock*> BMeasureApi::BMeasureUnits::odataBlocksProcess  [private]
```

7.8.3.6 odataBlocksProcessNum

BCondInt BMeasureApi::BMeasureUnits::odataBlocksProcessNum [private]

7.8.3.7 odataStreamNum

BUInt BMeasureApi::BMeasureUnits::odataStreamNum [private]

7.8.3.8 ofill

BUInt32 BMeasureApi::BMeasureUnits::ofill [private]

7.8.3.9 olocalTrigger

MeasurementConfig BMeasureApi::BMeasureUnits::olocalTrigger [private]

7.8.3.10 olockInput

BMutex BMeasureApi::BMeasureUnits::olockInput [private]

7.8.3.11 olockOutput

BMutex BMeasureApi::BMeasureUnits::olockOutput [private]

7.8.3.12 olockUnits

BMutex BMeasureApi::BMeasureUnits::olockUnits [private]

7.8.3.13 onumBlocks

BUInt BMeasureApi::BMeasureUnits::onumBlocks [private]

7.8.3.14 onumChannels

```
BUInt BMeasureApi::BMeasureUnits::onumChannels [private]
```

7.8.3.15 onumConnected

```
BUInt BMeasureApi::BMeasureUnits::onumConnected [private]
```

7.8.3.16 oprocEnable

```
BSemaphoreBool BMeasureApi::BMeasureUnits::oprocEnable [private]
```

Enable processing.

7.8.3.17 oprocRunning

```
BSemaphoreBool BMeasureApi::BMeasureUnits::oprocRunning [private]
```

Processing is running.

7.8.3.18 ostartSample

```
BUInt BMeasureApi::BMeasureUnits::ostartSample [private]
```

7.8.3.19 otriggered

```
Bool BMeasureApi::BMeasureUnits::otriggered [private]
```

7.8.3.20 ounitMaster

```
BInt BMeasureApi::BMeasureUnits::ounitMaster [private]
```

7.8.3.21 ounits

```
BList<BMeasureUnit1*> BMeasureApi::BMeasureUnits::ounits  [private]
```

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

- BMeasureUnits.h
- BMeasureUnits.cpp

7.9 BMeasureApi::BMeasureUnitsDataBlock Class Reference

```
#include <BMeasureUnits.h>
```

Public Member Functions

- BMeasureUnitsDataBlock (**BUInt** numChannels=0, **BUInt** numSamples=0)
- ∼BMeasureUnitsDataBlock ()
- void init (**BUInt** numChannels, **BUInt** numSamples)

Public Attributes

- DataBlock ∗ odataBlock
- **BUInt32** ofill
- **BUInt** oinUse

7.9.1 Constructor & Destructor Documentation

7.9.1.1 BMeasureUnitsDataBlock()

```
BMeasureApi::BMeasureUnitsDataBlock::BMeasureUnitsDataBlock (
              BUInt numChannels = 0,
              BUInt numSamples = 0 )
```

7.9.1.2 ∼BMeasureUnitsDataBlock()

```
BMeasureApi::BMeasureUnitsDataBlock::∼BMeasureUnitsDataBlock ( )
```

7.9.2 Member Function Documentation

7.9.2.1 init()

```
void BMeasureApi::BMeasureUnitsDataBlock::init (
              BUInt numChannels,
              BUInt numSamples )
```

7.9.3 Member Data Documentation

7.9.3.1 odataBlock

```
DataBlock* BMeasureApi::BMeasureUnitsDataBlock::odataBlock
```

7.9.3.2 ofill

```
BUInt32 BMeasureApi::BMeasureUnitsDataBlock::ofill
```

7.9.3.3 oinUse

```
BUInt BMeasureApi::BMeasureUnitsDataBlock::oinUse
```

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

- BMeasureUnits.h
- BMeasureUnits.cpp

7.10 BMeasureApi::BoardConfig Class Reference

```
#include <BMeasureD.h>
```

Static Public Member Functions

- static const **BObjMember** * getMembers ()

Public Attributes

- **BUInt32** magic
- Version hardwareVersion
- **BUInt8** serialNumber [12]
- **BTime** buildTime
- **BUInt8** macAddress [6]
- **BUInt8** spare0 [2]
- **BFloat64** calibOffsets [8]
- **BFloat64** calibScales0 [8]
- **BFloat64** calibScales1 [8]

7.10.1 Member Function Documentation

7.10.1.1 getMembers()

```
const BObjMember * BMeasureApi::BoardConfig::getMembers ( ) [static]
```

7.10.2 Member Data Documentation

7.10.2.1 buildTime

```
BTime BMeasureApi::BoardConfig::buildTime
```

7.10.2.2 calibOffsets

```
BFloat64 BMeasureApi::BoardConfig::calibOffsets[8]
```

7.10.2.3 calibScales0

```
BFloat64 BMeasureApi::BoardConfig::calibScales0[8]
```

7.10.2.4 calibScales1

```
BFloat64 BMeasureApi::BoardConfig::calibScales1[8]
```

7.10.2.5 hardwareVersion

```
Version BMeasureApi::BoardConfig::hardwareVersion
```

7.10.2.6 macAddress

```
BUInt8 BMeasureApi::BoardConfig::macAddress[6]
```

7.10.2.7 magic

```
BUInt32 BMeasureApi::BoardConfig::magic
```

7.10.2.8 serialNumber

```
BUInt8 BMeasureApi::BoardConfig::serialNumber[12]
```

7.10.2.9 spare0

```
BUInt8 BMeasureApi::BoardConfig::spare0[2]
```

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

- BMeasureD.h
- BMeasureD.cpp

7.11 BMeasureApi::ChannelConfig Class Reference

```
#include <BMeasureD.h>
```

Static Public Member Functions

- static const **BObjMember** $*$ getMembers ()

Public Attributes

- **BUInt8** number

  The channel number.
- **BUInt8** enabled

  Channel is enabled.
- **BUInt8** attenuator

  Attenuator number in use.
- ChannelType type

  The channel type.
- SampleType sampleType

  The sample type.
- **BUInt8** spare0 [3]
- **BUInt32** dataChannel

  Data channel.
- **BChar** id [16]
- **BChar** name [16]
- **BChar** siUnits [8]
- **BFloat64** calibOffset

  The calibration data offset.
- **BFloat64** calibScale

  The calibration data scale factor to volts.
- **BFloat64** calibScaleAtten1

  Attenuator 1 scaling.
- **BFloat64** pgaGain

  The PGA gain.
- **BFloat64** scale

  The user data scale factor.
- **BFloat64** offset

  The user data offset.
- **BChar** process [32]

7.11.1 Member Function Documentation

7.11.1.1 getMembers()

```
const  BObjMember * BMeasureApi::ChannelConfig::getMembers ( )  [static]
```

7.11.2 Member Data Documentation

7.11.2.1 attenuator

```
BUInt8 BMeasureApi::ChannelConfig::attenuator
```

Attenuator number in use.

7.11.2.2 calibOffset

```
BFloat64 BMeasureApi::ChannelConfig::calibOffset
```

The calibration data offset.

7.11.2.3 calibScale

```
BFloat64 BMeasureApi::ChannelConfig::calibScale
```

The calibration data scale factor to volts.

7.11.2.4 calibScaleAtten1

```
BFloat64 BMeasureApi::ChannelConfig::calibScaleAtten1
```

Attenuator 1 scaling.

7.11.2.5 dataChannel

```
BUInt32 BMeasureApi::ChannelConfig::dataChannel
```

Data channel.

7.11.2.6 enabled

```
BUInt8 BMeasureApi::ChannelConfig::enabled
```

Channel is enabled.

7.11.2.7 id

```
BChar BMeasureApi::ChannelConfig::id[16]
```

7.11.2.8 name

```
BChar BMeasureApi::ChannelConfig::name[16]
```

7.11.2.9 number

```
BUInt8 BMeasureApi::ChannelConfig::number
```

The channel number.

7.11.2.10 offset

```
BFloat64 BMeasureApi::ChannelConfig::offset
```

The user data offset.

7.11.2.11 pgaGain

```
BFloat64 BMeasureApi::ChannelConfig::pgaGain
```

The PGA gain.

7.11.2.12 process

```
BChar BMeasureApi::ChannelConfig::process[32]
```

7.11.2.13 sampleType

```
SampleType BMeasureApi::ChannelConfig::sampleType
```

The sample type.

7.11.2.14 scale

```
BFloat64 BMeasureApi::ChannelConfig::scale
```

The user data scale factor.

7.11.2.15 siUnits

```
BChar BMeasureApi::ChannelConfig::siUnits[8]
```

7.11.2.16 spare0

```
BUInt8 BMeasureApi::ChannelConfig::spare0[3]
```

7.11.2.17 type

```
ChannelType BMeasureApi::ChannelConfig::type
```

The channel type.

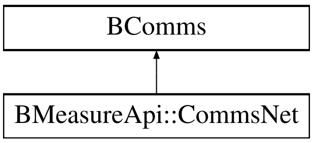
The documentation for this class was generated from the following files:

- BMeasureD.h
- BMeasureD.cpp

7.12 BMeasureApi::CommsNet Class Reference

```
#include <CommsNet.h>
```

Inheritance diagram for BMeasureApi::CommsNet:



Public Member Functions

- CommsNet (**BUInt** rxFifoSize=1024, **BUInt** txFifoSize=1024)
- ∼CommsNet ()
- **BError** init ()
- **BError** connect (**BString** host, **BUInt16** port)
- **BError** disconnect ()
- **BUInt** readAvailable ()
- **BError** wait (**BEventWaitSet** events, **BTimeout** timeout=-1, **BUInt32** num=1)
- **BError** read (void ∗ **data**, **BUInt32** num, **BUInt32** &nt)
- **BUInt** writeAvailable ()
- **BError** write (const void ∗ **data**, **BUInt32** nBytes, **BUInt32** &nt)
- **BError** writeChunks (const **BDataChunk** ∗chunks, **BUInt** nChunks, **BUInt32** &nt)

Protected Attributes

- **BSocket** osocket

Additional Inherited Members

7.12.1 Constructor & Destructor Documentation

7.12.1.1 CommsNet()

```
BMeasureApi::CommsNet::CommsNet (
              BUInt rxFifoSize = 1024,
              BUInt txFifoSize = 1024 )
```

7.12.1.2 ∼CommsNet()

```
BMeasureApi::CommsNet::∼CommsNet ( )
```

7.12.2 Member Function Documentation

7.12.2.1 connect()

```
 BError BMeasureApi::CommsNet::connect (
              BString host,
              BUInt16 port )
```

7.12.2.2 disconnect()

```
BError BMeasureApi::CommsNet::disconnect ( )  [virtual]
```

Reimplemented from **BComms**.

7.12.2.3 init()

```
BError BMeasureApi::CommsNet::init ( )  [virtual]
```

Reimplemented from **BComms**.

7.12.2.4 read()

```
BError BMeasureApi::CommsNet::read (
            void * data,
             BUInt32 num,
             BUInt32 & nt )  [virtual]
```

Implements **BComms**.

7.12.2.5 readAvailable()

```
BUInt BMeasureApi::CommsNet::readAvailable ( )  [virtual]
```

Reimplemented from **BComms**.

7.12.2.6 wait()

```
BError BMeasureApi::CommsNet::wait (
             BEventWaitSet events,
             BTimeout timeout = -1,
             BUInt32 num = 1 )  [virtual]
```

Reimplemented from **BComms**.

7.12.2.7 write()

```
BError BMeasureApi::CommsNet::write (
            const void * data,
             BUInt32 nBytes,
             BUInt32 & nt )  [virtual]
```

Implements **BComms**.

7.12.2.8 writeAvailable()

```
BUInt BMeasureApi::CommsNet::writeAvailable ( )  [virtual]
```

Reimplemented from **BComms**.

7.12.2.9 writeChunks()

```
BError BMeasureApi::CommsNet::writeChunks (
            const  BDataChunk * chunks,
             BUInt nChunks,
             BUInt32 & nt )  [virtual]
```

Reimplemented from **BComms**.

7.12.3 Member Data Documentation

7.12.3.1 osocket

```
BSocket BMeasureApi::CommsNet::osocket  [protected]
```

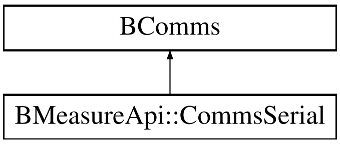
The documentation for this class was generated from the following files:

- CommsNet.h
- CommsNet.cpp

7.13 BMeasureApi::CommsSerial Class Reference

```
#include <CommsSerial.h>
```

Inheritance diagram for BMeasureApi::CommsSerial:



Public Member Functions

- CommsSerial ()
- ∼CommsSerial ()
- **BError** connect (**BString** device)
- **BError** disconnect ()
- **BUInt** readAvailable ()
- **BError** read (void ∗ **data**, **BUInt32** num, **BUInt32** &nTrans)
- **BError** write (const void ∗ **data**, **BUInt32** num, **BUInt32** &nTrans)
- **BError** wait (**BEventWaitSet** events, **BTimeout** timeout=-1, **BUInt32** num=1)

Private Attributes

- **BString** odevice
- int oserialPort

Additional Inherited Members

7.13.1 Constructor & Destructor Documentation

7.13.1.1 CommsSerial()

```
BMeasureApi::CommsSerial::CommsSerial ( )
```

7.13.1.2 ∼CommsSerial()

```
BMeasureApi::CommsSerial::∼CommsSerial ( )
```

7.13.2 Member Function Documentation

7.13.2.1 connect()

```
BError BMeasureApi::CommsSerial::connect (
            BString device )
```

7.13.2.2 disconnect()

```
BError BMeasureApi::CommsSerial::disconnect ( )  [virtual]
```

Reimplemented from **BComms**.

7.13.2.3 read()

```
BError BMeasureApi::CommsSerial::read (
            void * data,
             BUInt32 num,
             BUInt32 & nTrans )  [virtual]
```

Implements **BComms**.

7.13.2.4 readAvailable()

```
BUInt BMeasureApi::CommsSerial::readAvailable ( )  [virtual]
```

Reimplemented from **BComms**.

7.13.2.5 wait()

```
BError BMeasureApi::CommsSerial::wait (
             BEventWaitSet events,
             BTimeout timeout = -1,
             BUInt32 num = 1 )  [virtual]
```

Reimplemented from **BComms**.

7.13.2.6 write()

```
BError BMeasureApi::CommsSerial::write (
            const void * data,
             BUInt32 num,
             BUInt32 & nTrans )  [virtual]
```

Implements **BComms**.

7.13.3 Member Data Documentation

7.13.3.1 odevice

```
BString BMeasureApi::CommsSerial::odevice  [private]
```

7.13.3.2 oserialPort

```
int BMeasureApi::CommsSerial::oserialPort  [private]
```

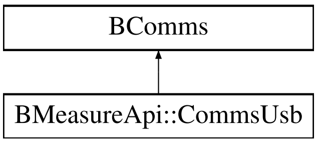
The documentation for this class was generated from the following file:

- CommsSerial.h

7.14 BMeasureApi::CommsUsb Class Reference

```
#include <CommsUsb.h>
```

Inheritance diagram for BMeasureApi::CommsUsb:



Public Member Functions

- CommsUsb ()
- ∼CommsUsb ()
- **BError** connect (**BString** device)
- **BError** disconnect ()
- **BUInt** readAvailable ()
- **BError** read (void ∗ **data**, **BUInt32** num, **BUInt32** &nTrans)
- **BError** write (const void ∗ **data**, **BUInt32** num, **BUInt32** &nTrans)
- **BError** wait (**BEventWaitSet** events, **BTimeout** timeout=-1, **BUInt32** num=1)

Private Member Functions

- **BError** readChunk ()

Private Attributes

- **BString** odevice
- libusb_context ∗ ocontext
- libusb_device_handle ∗ odev
- char obuffer [102400]
- **BUInt** onum
- **Bool** oterminated

Additional Inherited Members

7.14.1 Constructor & Destructor Documentation

7.14.1.1 CommsUsb()

```
BMeasureApi::CommsUsb::CommsUsb ( )
```

7.14.1.2 ∼CommsUsb()

```
BMeasureApi::CommsUsb::∼CommsUsb ( )
```

7.14.2 Member Function Documentation

7.14.2.1 connect()

```
 BError BMeasureApi::CommsUsb::connect (
              BString device )
```

7.14.2.2 disconnect()

```
 BError BMeasureApi::CommsUsb::disconnect ( )  [virtual]
```

Reimplemented from **BComms**.

7.14.2.3 read()

```
 BError BMeasureApi::CommsUsb::read (
             void * data,
              BUInt32 num,
              BUInt32 & nTrans )  [virtual]
```

Implements **BComms**.

7.14.2.4 readAvailable()

```
BUInt BMeasureApi::CommsUsb::readAvailable ( )  [virtual]
```

Reimplemented from **BComms**.

7.14.2.5 readChunk()

```
BError BMeasureApi::CommsUsb::readChunk ( )  [private]
```

7.14.2.6 wait()

```
BError BMeasureApi::CommsUsb::wait (
            BEventWaitSet events,
            BTimeout timeout = -1,
            BUInt32 num = 1 )  [virtual]
```

Reimplemented from **BComms**.

7.14.2.7 write()

```
BError BMeasureApi::CommsUsb::write (
           const void * data,
            BUInt32 num,
            BUInt32 & nTrans )  [virtual]
```

Implements **BComms**.

7.14.3 Member Data Documentation

7.14.3.1 obuffer

```
char BMeasureApi::CommsUsb::obuffer[102400]  [private]
```

7.14.3.2 ocontext

```
libusb_context* BMeasureApi::CommsUsb::ocontext  [private]
```

7.14.3.3 odev

```
libusb_device_handle* BMeasureApi::CommsUsb::odev  [private]
```

7.14.3.4 odevice

```
BString BMeasureApi::CommsUsb::odevice  [private]
```

7.14.3.5 onum

```
BUInt BMeasureApi::CommsUsb::onum  [private]
```

7.14.3.6 oterminated

```
Bool BMeasureApi::CommsUsb::oterminated  [private]
```

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

- CommsUsb.h
- CommsUsb.cpp

7.15 BMeasureApi::ConfigItem Class Reference

```
#include <BMeasureD.h>
```

Static Public Member Functions

- static const **BObjMember** ∗ getMembers ()

Public Attributes

- **BChar** name [16]
- **BUInt8** type

  The type of data.
- **BChar** value [16]

7.15.1 Member Function Documentation

7.15.1.1 getMembers()

```
const BObjMember * BMeasureApi::ConfigItem::getMembers ( ) [static]
```

7.15.2 Member Data Documentation

7.15.2.1 name

```
BChar BMeasureApi::ConfigItem::name[16]
```

7.15.2.2 type

```
BUInt8 BMeasureApi::ConfigItem::type
```

The type of data.

7.15.2.3 value

```
BChar BMeasureApi::ConfigItem::value[16]
```

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

- BMeasureD.h
- BMeasureD.cpp

7.16 BMeasureApi::Configuration Class Reference

```
#include <BMeasureD.h>
```

Static Public Member Functions

- static const **BObjMember** * getMembers ()

Public Attributes

- **BUInt32** version

  The configuration version.
- Mode mode

  The overall run mode.
- **BChar** name [16]
- **BChar** location [16]
- **BUInt8** captureData

  Capture the data.
- **BUInt8** logData

  Log the data.
- **BUInt8** logDataMode

  Log data mode.
- **BUInt8** logDataDevice

  The device to store data.
- **BUInt8** source

  The source number if multiple units are in use.
- **BUInt8** sampleFrequencyMode

  The base sample frequency mode.
- **BUInt8** spare1 [2]
- DigitalMode digitalMode

  The digital mode.
- **BUInt8** ethernetEnable

  Enable ethernet interface.
- **BUInt8** wifiEnable

  Enable wifi interface.
- **BUInt8** usbaEnable

  Enable USB-A interface.
- **BUInt8** usbbEnable

  Enable USB-B interface.
- **BUInt8** networkMode

  The network mode (0 - dhcp, 1 - static)
- **BUInt8** spare3 [3]
- **BUInt32** networkAddress

  The network IP address.
- **BUInt32** networkMask

  The network netmask.
- **BUInt32** networkGateway

  The network gateway.
- **BUInt32** networkTimeServer

  The network time server.
- **BUInt32** rs485BaudRate

  The RS485 baud rate.
- **BUInt8** rs485Bits

  The RS485 number of bits.
- **BUInt8** rs485StopBits

  The RS485 stop bits.
- **BChar** wifiAp1 [32]
- **BChar** wifiAp2 [32]

7.16.1 Member Function Documentation

7.16.1.1 getMembers()

```
const BObjMember * BMeasureApi::Configuration::getMembers ( ) [static]
```

7.16.2 Member Data Documentation

7.16.2.1 captureData

```
BUInt8 BMeasureApi::Configuration::captureData
```

Capture the data.

7.16.2.2 digitalMode

```
DigitalMode BMeasureApi::Configuration::digitalMode
```

The digital mode.

7.16.2.3 ethernetEnable

```
BUInt8 BMeasureApi::Configuration::ethernetEnable
```

Enable ethernet interface.

7.16.2.4 location

```
BChar BMeasureApi::Configuration::location[16]
```

7.16.2.5 logData

```
BUInt8 BMeasureApi::Configuration::logData
```

Log the data.

7.16.2.6 logDataDevice

```
BUInt8 BMeasureApi::Configuration::logDataDevice
```

The device to store data.

7.16.2.7 logDataMode

```
BUInt8 BMeasureApi::Configuration::logDataMode
```

Log data mode.

7.16.2.8 mode

```
Mode BMeasureApi::Configuration::mode
```

The overall run mode.

7.16.2.9 name

```
BChar BMeasureApi::Configuration::name[16]
```

7.16.2.10 networkAddress

```
BUInt32 BMeasureApi::Configuration::networkAddress
```

The network IP address.

7.16.2.11 networkGateway

```
BUInt32 BMeasureApi::Configuration::networkGateway
```

The network gateway.

7.16.2.12 networkMask

```
BUInt32 BMeasureApi::Configuration::networkMask
```

The network netmask.

7.16.2.13 networkMode

```
BUInt8 BMeasureApi::Configuration::networkMode
```

The network mode (0 - dhcp, 1 - static)

7.16.2.14 networkTimeServer

```
BUInt32 BMeasureApi::Configuration::networkTimeServer
```

The network time server.

7.16.2.15 rs485BaudRate

```
BUInt32 BMeasureApi::Configuration::rs485BaudRate
```

The RS485 baud rate.

7.16.2.16 rs485Bits

```
BUInt8 BMeasureApi::Configuration::rs485Bits
```

The RS485 number of bits.

7.16.2.17 rs485StopBits

```
BUInt8 BMeasureApi::Configuration::rs485StopBits
```

The RS485 stop bits.

7.16.2.18 sampleFrequencyMode

```
BUInt8 BMeasureApi::Configuration::sampleFrequencyMode
```

The base sample frequency mode.

7.16.2.19 source

```
BUInt8 BMeasureApi::Configuration::source
```

The source number if multiple units are in use.

7.16.2.20 spare1

```
BUInt8 BMeasureApi::Configuration::spare1[2]
```

7.16.2.21 spare3

```
BUInt8 BMeasureApi::Configuration::spare3[3]
```

7.16.2.22 usbaEnable

```
BUInt8 BMeasureApi::Configuration::usbaEnable
```

Enable USB-A interface.

7.16.2.23 usbbEnable

```
BUInt8 BMeasureApi::Configuration::usbbEnable
```

Enable USB-B interface.

7.16.2.24 version

```
BUInt32 BMeasureApi::Configuration::version
```

The configuration version.

7.16.2.25 wifiAp1

```
BChar BMeasureApi::Configuration::wifiAp1[32]
```

7.16.2.26 wifiAp2

```
BChar BMeasureApi::Configuration::wifiAp2[32]
```

7.16.2.27 wifiEnable

```
BUInt8 BMeasureApi::Configuration::wifiEnable
```

Enable wifi interface.

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

- BMeasureD.h
- BMeasureD.cpp

7.17 BMeasureApi::DataBlock Class Reference

```
#include <BMeasureD.h>
```

Static Public Member Functions

- static const **BObjMember** ∗ getMembers ()

Public Attributes

- **BUInt64** time

  The time in microseconds since 1970-01-01 to TAI.
- **BUInt16** source

  The source unit,.
- **BUInt16** status
- **BUInt16** numChannels

  The number of data channels.
- **BUInt16** numSamples

  The number of samples.
- **BUInt32** sequence

  The sequence number.
- **BUInt8** sampleTypes [8]
- **BFloat32** data [117]

7.17.1 Member Function Documentation

7.17.1.1 getMembers()

```
const BObjMember * BMeasureApi::DataBlock::getMembers ( ) [static]
```

7.17.2 Member Data Documentation

7.17.2.1 data

```
BFloat32 BMeasureApi::DataBlock::data[117]
```

7.17.2.2 numChannels

```
BUInt16 BMeasureApi::DataBlock::numChannels
```

The number of data channels.

7.17.2.3 numSamples

```
BUInt16 BMeasureApi::DataBlock::numSamples
```

The number of samples.

7.17.2.4 sampleTypes

```
BUInt8 BMeasureApi::DataBlock::sampleTypes[8]
```

7.17.2.5 sequence

```
BUInt32 BMeasureApi::DataBlock::sequence
```

The sequence number.

7.17.2.6 source

```
BUInt16 BMeasureApi::DataBlock::source
```

The source unit,.

7.17.2.7 status

```
BUInt16 BMeasureApi::DataBlock::status
```

7.17.2.8 time

```
BUInt64 BMeasureApi::DataBlock::time
```

The time in microseconds since 1970-01-01 to TAI.

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

- BMeasureD.h
- BMeasureD.cpp

7.18 BMeasureApi::DataFile Class Reference

```
#include <DataFile.h>
```

Public Member Functions

- DataFile ()
- ∼DataFile ()
- void init ()

  Initialise.
- **BError** open (**BString** fileName, **BString** mode, **BString** format="")

  Open the file for read or write.
- **BError** close ()

  Close the file.
- **BString** getFileName ()

  Return the file name.
- **BError** writeInfo (const InfoBlock &infoBlock, const ChannelConfigs &channels)
- **BError** writeData (DataBlock ∗ **data**)

  Write a block of data.
- **BError** writeEnd ()
- **BError** readInfo (**BString** &format, InfoBlock &infoBlock, ChannelConfigs &channels)
- **BError** readData (DataBlock ∗ **data**)

  Read a block of data.

Private Member Functions

- **BError** validateFormat (**BString** format)
- **BError** writeInfoTdms (const InfoBlock &infoBlock, const ChannelConfigs &channels)
- **BError** writeInfoBMeas (const InfoBlock &infoBlock, const ChannelConfigs &channels)

Private Attributes

- **BString** ofileName
- **BString** omode
- **BString** oformat
- **BFile** ofile
- **BUInt32** opacketLen
- **BoapMc1Packet** ∗ opacket

7.18.1 Constructor & Destructor Documentation

7.18.1.1 DataFile()

```
BMeasureApi::DataFile::DataFile ( )
```

7.18.1.2 ∼DataFile()

```
BMeasureApi::DataFile::∼DataFile ( )
```

7.18.2 Member Function Documentation

7.18.2.1 close()

```
BError BMeasureApi::DataFile::close ( )
```

Close the file.

7.18.2.2 getFileName()

```
BString BMeasureApi::DataFile::getFileName ( )
```

Return the file name.

7.18.2.3 init()

```
void BMeasureApi::DataFile::init ( )
```

Initialise.

7.18.2.4 open()

```
BError BMeasureApi::DataFile::open (
             BString fileName,
             BString mode,
             BString format = "" )
```

Open the file for read or write.

7.18.2.5 readData()

```
BError BMeasureApi::DataFile::readData (
            DataBlock * data )
```

Read a block of data.

7.18.2.6 readInfo()

```
BError BMeasureApi::DataFile::readInfo (
            BString & format,
            InfoBlock & infoBlock,
            ChannelConfigs & channels )
```

7.18.2.7 validateFormat()

```
BError BMeasureApi::DataFile::validateFormat (
            BString format )  [private]
```

7.18.2.8 writeData()

```
BError BMeasureApi::DataFile::writeData (
            DataBlock * data )
```

Write a block of data.

7.18.2.9 writeEnd()

```
BError BMeasureApi::DataFile::writeEnd ( )
```

7.18.2.10 writeInfo()

```
BError BMeasureApi::DataFile::writeInfo (
            const InfoBlock & infoBlock,
            const ChannelConfigs & channels )
```

7.18.2.11 writeInfoBMeas()

```
BError BMeasureApi::DataFile::writeInfoBMeas (
            const InfoBlock & infoBlock,
            const ChannelConfigs & channels )  [private]
```

7.18.2.12 writeInfoTdms()

```
BError BMeasureApi::DataFile::writeInfoTdms (
            const InfoBlock & infoBlock,
            const ChannelConfigs & channels )  [private]
```

7.18.3 Member Data Documentation

7.18.3.1 ofile

```
BFile BMeasureApi::DataFile::ofile  [private]
```

7.18.3.2 ofileName

```
BString BMeasureApi::DataFile::ofileName  [private]
```

7.18.3.3 oformat

```
BString BMeasureApi::DataFile::oformat  [private]
```

7.18.3.4 omode

```
BString BMeasureApi::DataFile::omode  [private]
```

7.18.3.5 opacket

```
BoapMc1Packet* BMeasureApi::DataFile::opacket  [private]
```

7.18.3.6 opacketLen

```
BUInt32 BMeasureApi::DataFile::opacketLen  [private]
```

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

- DataFile.h
- DataFile.cpp

7.19 BMeasureApi::FileData Class Reference

```
#include <BMeasureD.h>
```

Static Public Member Functions

- static const **BObjMember** ∗ getMembers ()

Public Attributes

- **BUInt32** length

    The data length.
- **BUInt8** data [256]

7.19.1 Member Function Documentation

7.19.1.1 getMembers()

```
const BObjMember * BMeasureApi::FileData::getMembers ( ) [static]
```

7.19.2 Member Data Documentation

7.19.2.1 data

```
BUInt8 BMeasureApi::FileData::data[256]
```

7.19.2.2 length

```
BUInt32 BMeasureApi::FileData::length
```

The data length.

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

- BMeasureD.h
- BMeasureD.cpp

7.20 BMeasureApi::FileInfo Class Reference

```
#include <BMeasureD.h>
```

Static Public Member Functions

- static const **BObjMember** ∗ getMembers ()

Public Attributes

- **BChar** name [128]
- **BTime** time

  The file date/time.
- FileType fileType

  The file type.
- **BUInt64** fileLength

  The file length.

7.20.1 Member Function Documentation

7.20.1.1 getMembers()

```
const BObjMember ∗ BMeasureApi::FileInfo::getMembers ( ) [static]
```

7.20.2 Member Data Documentation

7.20.2.1 fileLength

```
BUInt64 BMeasureApi::FileInfo::fileLength
```

The file length.

7.20.2.2 fileType

```
FileType BMeasureApi::FileInfo::fileType
```

The file type.

7.20.2.3 name

```
BChar BMeasureApi::FileInfo::name[128]
```

7.20.2.4 time

```
BTime BMeasureApi::FileInfo::time
```

The file date/time.

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

- BMeasureD.h
- BMeasureD.cpp

7.21 BMeasureApi::FilesysInfo Class Reference

```
#include <BMeasureD.h>
```

Static Public Member Functions

- static const **BObjMember** ∗ getMembers ()

Public Attributes

- **BChar** name [128]
- **BUInt64** size

  The store size.
- **BUInt64** free

  The store free space.

7.21.1 Member Function Documentation

7.21.1.1 getMembers()

```
const BObjMember * BMeasureApi::FilesysInfo::getMembers ( ) [static]
```

7.21.2 Member Data Documentation

7.21.2.1 free

```
BUInt64 BMeasureApi::FilesysInfo::free
```

The store free space.

7.21.2.2 name

```
BChar BMeasureApi::FilesysInfo::name[128]
```

7.21.2.3 size

```
BUInt64 BMeasureApi::FilesysInfo::size
```

The store size.

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

- BMeasureD.h
- BMeasureD.cpp

7.22 BMeasureApi::InfoBlock Class Reference

```
#include <BMeasureD.h>
```

Static Public Member Functions

- static const **BObjMember** ∗ getMembers ()

Public Attributes

- **BUInt64** time

  The time in microseconds since 1970-01-01 to TAI.
- **BUInt16** source

  The source unit.
- **BUInt16** numChannels

  The number of data channels.
- **BUInt16** version

  The info/data version.
- **BUInt16** spare0
- **BChar** name [16]
- **BChar** location [16]
- **BChar** description [256]
- NodeInfo nodeInfo

  Information on the unit.
- MeasurementConfig measureConfig

  The measurement configuration.

7.22.1 Member Function Documentation

7.22.1.1 getMembers()

```
const  BObjMember * BMeasureApi::InfoBlock::getMembers ( )  [static]
```

7.22.2 Member Data Documentation

7.22.2.1 description

```
BChar BMeasureApi::InfoBlock::description[256]
```

7.22.2.2 location

```
BChar BMeasureApi::InfoBlock::location[16]
```

7.22.2.3 measureConfig

```
MeasurementConfig BMeasureApi::InfoBlock::measureConfig
```

The measurement configuration.

7.22.2.4 name

```
BChar BMeasureApi::InfoBlock::name[16]
```

7.22.2.5 nodeInfo

```
NodeInfo BMeasureApi::InfoBlock::nodeInfo
```

Information on the unit.

7.22.2.6 numChannels

```
BUInt16 BMeasureApi::InfoBlock::numChannels
```

The number of data channels.

7.22.2.7 source

```
BUInt16 BMeasureApi::InfoBlock::source
```

The source unit.

7.22.2.8 spare0

```
BUInt16 BMeasureApi::InfoBlock::spare0
```

7.22.2.9 time

```
BUInt64 BMeasureApi::InfoBlock::time
```

The time in microseconds since 1970-01-01 to TAI.

7.22.2.10 version

```
BUInt16 BMeasureApi::InfoBlock::version
```

The info/data version.

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

- BMeasureD.h
- BMeasureD.cpp

7.23 BMeasureApi::Information Class Reference

```
#include <BMeasureD.h>
```

Static Public Member Functions

- static const **BObjMember** ∗ getMembers ()

Public Attributes

- NodeInfo nodeInfo
- **BUInt8** numConfigItems

  The number of config items.
- **BUInt8** numChannels

  The number of channels.
- **BUInt32** logMemorySize

  SD card memory size, 0 - not present.

7.23.1 Member Function Documentation

7.23.1.1 getMembers()

```
const BObjMember ∗ BMeasureApi::Information::getMembers ( ) [static]
```

7.23.2 Member Data Documentation

7.23.2.1 logMemorySize

```
BUInt32 BMeasureApi::Information::logMemorySize
```

SD card memory size, 0 - not present.

7.23.2.2 nodeInfo

```
NodeInfo BMeasureApi::Information::nodeInfo
```

7.23.2.3 numChannels

```
BUInt8 BMeasureApi::Information::numChannels
```

The number of channels.

7.23.2.4 numConfigItems

```
BUInt8 BMeasureApi::Information::numConfigItems
```

The number of config items.

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

- BMeasureD.h
- BMeasureD.cpp

7.24 BMeasureApi::MeasurementConfig Class Reference

```
#include <BMeasureD.h>
```

Static Public Member Functions

- static const **BObjMember** ∗ getMembers ()

Public Attributes

- MeasureMode measureMode
- TriggerMode triggerMode
- TriggerConfig triggerConfig

  Trigger config including direction, filters etc.
- **BUInt8** triggerChannel
- **BFloat64** triggerLevel
- **BInt32** triggerDelay

  Trigger delay in samples.
- **BFloat64** sampleRate
- **BUInt32** numSamples0

  The number of samples in a chunk for display and/or repeat.
- **BUInt32** numSamples1

  The number of samples to capture. 0 is continuous.
- **BUInt32** measurePeriod

  Time in seconds between measurement sample bursts. 0 is continuous.

7.24.1 Member Function Documentation

7.24.1.1 getMembers()

```
const BObjMember * BMeasureApi::MeasurementConfig::getMembers ( ) [static]
```

7.24.2 Member Data Documentation

7.24.2.1 measureMode

```
MeasureMode BMeasureApi::MeasurementConfig::measureMode
```

7.24.2.2 measurePeriod

```
BUInt32 BMeasureApi::MeasurementConfig::measurePeriod
```

Time in seconds between measurement sample bursts. 0 is continuous.

7.24.2.3 numSamples0

```
BUInt32 BMeasureApi::MeasurementConfig::numSamples0
```

The number of samples in a chunk for display and/or repeat.

7.24.2.4 numSamples1

```
BUInt32 BMeasureApi::MeasurementConfig::numSamples1
```

The number of samples to capture. 0 is continuous.

7.24.2.5 sampleRate

```
BFloat64 BMeasureApi::MeasurementConfig::sampleRate
```

7.24.2.6 triggerChannel

```
BUInt8 BMeasureApi::MeasurementConfig::triggerChannel
```

7.24.2.7 triggerConfig

```
TriggerConfig BMeasureApi::MeasurementConfig::triggerConfig
```

Trigger config including direction, filters etc.

7.24.2.8 triggerDelay

```
BInt32 BMeasureApi::MeasurementConfig::triggerDelay
```

Trigger delay in samples.

7.24.2.9 triggerLevel

```
BFloat64 BMeasureApi::MeasurementConfig::triggerLevel
```

7.24.2.10 triggerMode

```
TriggerMode BMeasureApi::MeasurementConfig::triggerMode
```

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

- BMeasureD.h
- BMeasureD.cpp

7.25 BMeasureApi::NodeInfo Class Reference

```
#include <BMeasureD.h>
```

Static Public Member Functions

- static const **BObjMember** $*$ getMembers ()

Public Attributes

- **BUInt32** apiVersion
- Version hardwareVersion
- Version fpgaVersion
- Version softwareVersion
- **BChar** serialNumber [12]

7.25.1 Member Function Documentation

7.25.1.1 getMembers()

```
const BObjMember * BMeasureApi::NodeInfo::getMembers ( ) [static]
```

7.25.2 Member Data Documentation

7.25.2.1 apiVersion

```
BUInt32 BMeasureApi::NodeInfo::apiVersion
```

7.25.2.2 fpgaVersion

```
Version BMeasureApi::NodeInfo::fpgaVersion
```

7.25.2.3 hardwareVersion

```
Version BMeasureApi::NodeInfo::hardwareVersion
```

7.25.2.4 serialNumber

```
BChar BMeasureApi::NodeInfo::serialNumber[12]
```

7.25.2.5 softwareVersion

```
Version BMeasureApi::NodeInfo::softwareVersion
```

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

- BMeasureD.h
- BMeasureD.cpp

7.26 BMeasureApi::NodeStatus Class Reference

```
#include <BMeasureD.h>
```

Static Public Member Functions

- static const **BObjMember** ∗ getMembers ()

Public Attributes

- **BTimeUs** time
- **BUInt32** status
- **BUInt32** error
- **BChar** errorStr [32]

7.26.1 Member Function Documentation

7.26.1.1 getMembers()

```
const BObjMember * BMeasureApi::NodeStatus::getMembers ( ) [static]
```

7.26.2 Member Data Documentation

7.26.2.1 error

```
BUInt32 BMeasureApi::NodeStatus::error
```

7.26.2.2 errorStr

```
BChar BMeasureApi::NodeStatus::errorStr[32]
```

7.26.2.3 status

```
BUInt32 BMeasureApi::NodeStatus::status
```

7.26.2.4 time

BTimeUs BMeasureApi::NodeStatus::time

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

- BMeasureD.h
- BMeasureD.cpp

7.27 BMeasureApi::Version Class Reference

```
#include <BMeasureD.h>
```

Static Public Member Functions

- static const **BObjMember** ∗ getMembers ()

Public Attributes

- **BUInt8** type
- **BUInt8** ver0
- **BUInt8** ver1
- **BUInt8** ver2

7.27.1 Member Function Documentation

7.27.1.1 getMembers()

const **BObjMember** ∗ BMeasureApi::Version::getMembers ( ) [static]

7.27.2 Member Data Documentation

7.27.2.1 type

BUInt8 BMeasureApi::Version::type

7.27.2.2 ver0

```
BUInt8 BMeasureApi::Version::ver0
```

7.27.2.3 ver1

```
BUInt8 BMeasureApi::Version::ver1
```

7.27.2.4 ver2

```
BUInt8 BMeasureApi::Version::ver2
```

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

- BMeasureD.h
- BMeasureD.cpp

Chapter 8

File Documentation

8.1 BMdns.cpp File Reference

```
#include <BMdns.h>
#include <BDebug.h>
#include <stdio.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <net/if.h>
```

Macros

- #define BDEBUGL1 0

Enumerations

- enum MdnsRecordType {
  MDNS_RECORDTYPE_IGNORE = 0, MDNS_RECORDTYPE_A = 1, MDNS_RECORDTYPE_PTR = 12,
  MDNS_RECORDTYPE_TXT = 16,
  MDNS_RECORDTYPE_AAAA = 28, MDNS_RECORDTYPE_SRV = 33 }
- enum MdnsEntryType { MDNS_ENTRYTYPE_ANSWER = 1, MDNS_ENTRYTYPE_AUTHORITY = 2, MDNS_ENTRYTYPE_ADDITIONAL = 3 }
- enum MdnsClass { MDNS_CLASS_IN = 1 }

Functions

- static int mdns_write_string (**BUInt8** ∗buffer, **BUInt8** ∗p, **BString** str)
- static int mdns_read_string (void ∗buffer, **BUInt8** ∗p, **BString** &str)
- static int mdns_read_strings (void ∗buffer, **BUInt8** ∗p, **BString** &str)

8.1.1 Macro Definition Documentation

8.1.1.1 BDEBUGL1

```
#define BDEBUGL1 0
```

8.1.2 Enumeration Type Documentation

8.1.2.1 MdnsClass

```
enum MdnsClass
```

Enumerator

MDNS_CLASS_IN	

8.1.2.2 MdnsEntryType

```
enum MdnsEntryType
```

Enumerator

MDNS_ENTRYTYPE_ANSWER	
MDNS_ENTRYTYPE_AUTHORITY	
MDNS_ENTRYTYPE_ADDITIONAL	

8.1.2.3 MdnsRecordType

```
enum MdnsRecordType
```

Enumerator

MDNS_RECORDTYPE_IGNORE	
MDNS_RECORDTYPE_A	
MDNS_RECORDTYPE_PTR	
MDNS_RECORDTYPE_TXT	
MDNS_RECORDTYPE_AAAA	
MDNS_RECORDTYPE_SRV	

8.1.3 Function Documentation

8.1.3.1 mdns_read_string()

```
static int mdns_read_string (
             void * buffer,
              BUInt8 * p,
              BString & str )  [static]
```

8.1.3.2 mdns_read_strings()

```
static int mdns_read_strings (
             void * buffer,
              BUInt8 * p,
              BString & str )  [static]
```

8.1.3.3 mdns_write_string()

```
static int mdns_write_string (
              BUInt8 * buffer,
              BUInt8 * p,
              BString str )  [static]
```

8.2 BMdns.h File Reference

```
#include <BSocket.h>
```

Classes

- class BMdnsService
- class BMdns

8.3 BMeasureB.cpp File Reference

```
#include <BMeasureB.h>
#include <string.h>
```

Namespaces

- BMeasureApi

8.4 BMeasureB.h File Reference

```
#include <BTypes.h>
#include <BComplex.h>
#include <BoapMc1.h>
#include <BMeasureD.h>
```

Classes

- class BMeasureApi::BMeasure

Namespaces

- BMeasureApi

Variables

- const **BUInt32** BMeasureApi::apiVersion = 0

8.5 BMeasureD.cpp File Reference

```
#include <BMeasureD.h>
```

Namespaces

- BMeasureApi

Macros

- #define boffsetof(T, F) ((**BUInt**)((char∗)&((T∗)0L)->F - (char∗)0L))

8.5.1 Macro Definition Documentation

8.5.1.1 boffsetof

```
#define boffsetof(
              T,
              F ) (( BUInt)((char*)&((T*)0L)->F - (char*)0L))
```

8.6 BMeasureD.h File Reference

```
#include <BTypes.h>
#include <BObj.h>
#include <BTime.h>
#include <BTimeUs.h>
#include <BArray.h>
#include <BComplex.h>
#include <BoapMc.h>
```

Classes

- class BMeasureApi::Version
- class BMeasureApi::NodeInfo
- class BMeasureApi::NodeStatus
- class BMeasureApi::BoardConfig
- class BMeasureApi::ChannelConfig
- class BMeasureApi::Information
- class BMeasureApi::Configuration
- class BMeasureApi::ConfigItem
- class BMeasureApi::MeasurementConfig
- class BMeasureApi::DataBlock
- class BMeasureApi::InfoBlock
- class BMeasureApi::AwgConfig
- class BMeasureApi::FilesysInfo
- class BMeasureApi::FileInfo
- class BMeasureApi::FileData

Namespaces

- BMeasureApi

Enumerations

- enum BMeasureApi::ErrorNum { BMeasureApi::ErrorSystem = 64, BMeasureApi::ErrorDataOverrun = 65 }
- enum BMeasureApi::NodeType { BMeasureApi::NodeTypeNone = 0, BMeasureApi::NodeTypeBMeasure1 = 1 }
- enum BMeasureApi::SecureMode { BMeasureApi::SecureModeOpen, BMeasureApi::SecureMoteRemote, BMeasureApi::SecureModeFull }
- enum BMeasureApi::Status {
  BMeasureApi::StatusNone = 0x00, BMeasureApi::StatusError = 0x01, BMeasureApi::StatusWarning = 0x02, BMeasureApi::StatusRun = 0x04,
  BMeasureApi::StatusDataOverrun = 0x08, BMeasureApi::StatusEnd0 = 0x10, BMeasureApi::StatusEnd1 = 0x20, BMeasureApi::StatusFpgaOverrun = 0x40 }

- enum BMeasureApi::Mode {
  BMeasureApi::ModeSleep = 0, BMeasureApi::ModeIdle = 1, BMeasureApi::ModeRun = 2, BMeasureApi::ModeRunProgram = 3,
  BMeasureApi::ModeInternal = 4, BMeasureApi::ModeDemo1 = 5 }
- enum BMeasureApi::BlockTypes { BMeasureApi::BlockTypeInfo = 0x424E4531, BMeasureApi::BlockTypeData = 0x424E4532 }
- enum BMeasureApi::ChannelType {
  BMeasureApi::ChannelTypeNone = 0, BMeasureApi::ChannelTypeAnalogueIn = 1, BMeasureApi::ChannelTypeAnalogueOut = 0x81, BMeasureApi::ChannelTypeDigitalIn = 2,
  BMeasureApi::ChannelTypeDigitalOut = 0x82 }
- enum BMeasureApi::SampleType {
  BMeasureApi::SampleTypeNone = 0, BMeasureApi::SampleTypeBool = 1, BMeasureApi::SampleTypeInt8 = 2, BMeasureApi::SampleTypeInt16 = 3,
  BMeasureApi::SampleTypeInt32 = 4, BMeasureApi::SampleTypeFloat32 = 5, BMeasureApi::SampleTypeFloat64 = 6 }
- enum BMeasureApi::SyncMode { BMeasureApi::SyncModeOff = 0, BMeasureApi::SyncModeMaster = 1, BMeasureApi::SyncModeSlave = 2 }
- enum BMeasureApi::MeasureMode { BMeasureApi::MeasureModeOff = 0, BMeasureApi::MeasureModeOneShot = 1, BMeasureApi::MeasureModeRepeat = 2, BMeasureApi::MeasureModeContinuous = 3 }
- enum BMeasureApi::TriggerMode { BMeasureApi::TriggerModeOff = 0, BMeasureApi::TriggerModePositive = 1, BMeasureApi::TriggerModeNegative = 2 }
- enum BMeasureApi::TriggerConfig { BMeasureApi::TriggerConfigNone = 0 }
- enum BMeasureApi::DigitalMode {
  BMeasureApi::DigitalModeInput = 0, BMeasureApi::DigitalModeOutput = 1, BMeasureApi::DigitalInOut = 2,
  BMeasureApi::DigitalModeSyncMaster = 3,
  BMeasureApi::DigitalModeSyncSlave = 4 }
- enum BMeasureApi::Waveform {
  BMeasureApi::WaveformNone, BMeasureApi::WaveformSine, BMeasureApi::WaveformSquare, BMeasureApi::WaveformTriang
  BMeasureApi::WaveformNoise, BMeasureApi::WaveformArbitrary }
- enum BMeasureApi::CalibrateMode { BMeasureApi::CalibrateModeNone, BMeasureApi::CalibrateModeOffsets }
- enum BMeasureApi::FileType { BMeasureApi::FileTypeNone, BMeasureApi::FileTypeFile, BMeasureApi::FileTypeDir }
- enum BMeasureApi::FilesysDeleteType { BMeasureApi::FilesysDeleteTypeNone, BMeasureApi::FilesysDeleteTypeData, BMeasureApi::FilesysDeleteTypeFormat }
- enum BMeasureApi::LogDataMode { BMeasureApi::LogDataModeNormal, BMeasureApi::LogDataModeDeleteOld }

8.7 BMeasureLib.cpp File Reference

```
#include <BMeasureLib.h>
#include <BDebug.h>
```

Namespaces

- BMeasureApi

Macros

- #define BDEBUGL1 0
- #define BDEBUGL2 0

8.7.1 Macro Definition Documentation

8.7.1.1 BDEBUGL1

```
#define BDEBUGL1 0
```

8.7.1.2 BDEBUGL2

```
#define BDEBUGL2 0
```

8.8 BMeasureLib.h File Reference

```
#include <BMeasureD.h>
```

Namespaces

- BMeasureApi

Typedefs

- typedef **BArray**< ChannelConfig > BMeasureApi::ChannelConfigs

8.9 BMeasureS.cpp File Reference

```
#include <BMeasureS.h>
#include <string.h>
```

Namespaces

- BMeasureApi

8.10 BMeasureUnit.cpp File Reference

```
#include <BMeasureUnit.h>
#include <CommsSerial.h>
#include <CommsNet.h>
#include <CommsUsb.h>
#include <BDir.h>
#include <BSys.h>
#include <libusb-1.0/libusb.h>
#include <BMdns.h>
#include <BDebug.h>
```

Namespaces

- BMeasureApi

Macros

- #define BDEBUGL1 0
- #define BDEBUGL2 0
- #define CONVERT_FLOAT 0

  Convert to floating point.

Functions

- const char ∗ BMeasureApi::channelTypeString (ChannelType type)
- const char ∗ BMeasureApi::sampleTypeString (SampleType type)
- **BFloat32** BMeasureApi::toFloat (**BUInt32** v)

8.10.1 Macro Definition Documentation

8.10.1.1 BDEBUGL1

```
#define BDEBUGL1 0
```

8.10.1.2 BDEBUGL2

```
#define BDEBUGL2 0
```

8.10.1.3 CONVERT_FLOAT

```
#define CONVERT_FLOAT 0
```

Convert to floating point.

8.11 BMeasureUnit.h File Reference

```
#include <BMeasureD.h>
#include <BMeasureB.h>
#include <BTask.h>
```

Classes

- class BMeasureApi::BMeasureUnitDevice
- class BMeasureApi::BMeasureUnit

Namespaces

- BMeasureApi

Functions

- const char ∗ BMeasureApi::channelTypeString (ChannelType type)
- const char ∗ BMeasureApi::sampleTypeString (SampleType type)

8.12 BMeasureUnits.cpp File Reference

```
#include <BMeasureUnits.h>
#include <BDebug.h>
#include <unistd.h>
```

Namespaces

- BMeasureApi

Macros

- #define BDEBUGL1 0
- #define BDEBUGL2 0
- #define BDEBUGL3 0

Functions

- static int BMeasureApi::unitSort (BMeasureUnit1 ∗&u1, BMeasureUnit1 ∗&u2)

8.12.1 Macro Definition Documentation

8.12.1.1 BDEBUGL1

```
#define BDEBUGL1 0
```

8.12.1.2 BDEBUGL2

```
#define BDEBUGL2 0
```

8.12.1.3 BDEBUGL3

```
#define BDEBUGL3 0
```

8.13 BMeasureUnits.h File Reference

```
#include <BMeasureUnit.h>
#include <BMutex.h>
#include <BSemaphore.h>
```

Classes

- class BMeasureApi::BMeasureUnit1
- class BMeasureApi::BMeasureUnitsDataBlock
- class BMeasureApi::BMeasureUnits

Namespaces

- BMeasureApi

8.14 CommsNet.cpp File Reference

```
#include <CommsNet.h>
#include <BPoll.h>
#include <BDebug.h>
#include <string.h>
```

Namespaces

- BMeasureApi

Macros

- #define BDEBUGL1 0
- #define BDEBUGL2 0
- #define BDEBUGL3 0

8.14.1 Macro Definition Documentation

8.14.1.1 BDEBUGL1

```
#define BDEBUGL1 0
```

8.14.1.2 BDEBUGL2

```
#define BDEBUGL2 0
```

8.14.1.3 BDEBUGL3

```
#define BDEBUGL3 0
```

8.15 CommsNet.h File Reference

```
#include <BComms.h>
#include <BSocket.h>
```

Classes

- class BMeasureApi::CommsNet

Namespaces

- BMeasureApi

8.16 CommsSerial.cpp File Reference

8.17 CommsSerial.h File Reference

```
#include <BComms.h>
```

Classes

- class BMeasureApi::CommsSerial

Namespaces

- BMeasureApi

8.18 CommsUsb.cpp File Reference

```
#include <CommsUsb.h>
#include <BSys.h>
#include <libusb-1.0/libusb.h>
#include <stdio.h>
#include <stdlib.h>
#include <BDebug.h>
```

Namespaces

- BMeasureApi

Macros

- #define BDEBUGL1 0
- #define BDEBUGL2 0

8.18.1 Macro Definition Documentation

8.18.1.1 BDEBUGL1

```
#define BDEBUGL1 0
```

8.18.1.2 BDEBUGL2

```
#define BDEBUGL2 0
```

8.19 CommsUsb.h File Reference

```
#include <BComms.h>
#include <BMutex.h>
#include <libusb-1.0/libusb.h>
```

Classes

- class BMeasureApi::CommsUsb

Namespaces

- BMeasureApi

8.20 DataFile.cpp File Reference

```
#include <DataFile.h>
#include <BoapMc1.h>
#include <BBuffer.h>
#include <BDebug.h>
```

Namespaces

- BMeasureApi

Macros

- #define BDEBUGL1 0
- #define BDEBUGL2 0

Enumerations

- enum BMeasureApi::TdsDataType {
BMeasureApi::TdsTypeVoid, BMeasureApi::TdsTypeI8, BMeasureApi::TdsTypeI16, BMeasureApi::TdsTypeI32,
BMeasureApi::TdsTypeI64, BMeasureApi::TdsTypeU8, BMeasureApi::TdsTypeU16, BMeasureApi::TdsTypeU32,
BMeasureApi::TdsTypeU64, BMeasureApi::TdsTypeSingleFloat, BMeasureApi::TdsTypeDoubleFloat,
BMeasureApi::TdsTypeExtendedFloat,
BMeasureApi::TdsTypeSingleFloatWithUnit =0x19, BMeasureApi::TdsTypeDoubleFloatWithUnit, BMeasureApi::TdsTypeExtend
BMeasureApi::TdsTypeString =0x20,
BMeasureApi::TdsTypeBoolean =0x21, BMeasureApi::TdsTypeTimeStamp =0x44, BMeasureApi::TdsTypeFixedPoint =0x4F, BMeasureApi::TdsTypeComplexSingleFloat =0x08000c,
BMeasureApi::TdsTypeComplexDoubleFloat =0x10000d, BMeasureApi::TdsTypeDAQmxRawData =0xFF↩FFFFFF }

Functions

- const **BUInt32** BMeasureApi::TocMetaData (1<< 1)
- const **BUInt32** BMeasureApi::TocNewObjList (1<< 2)
- const **BUInt32** BMeasureApi::TocRawData (1<< 3)
- const **BUInt32** BMeasureApi::TocInterleavedData (1<< 5)
- const **BUInt32** BMeasureApi::TocBigEndian (1<< 6)
- const **BUInt32** BMeasureApi::TocDaqRawData (1<< 7)
- **BUInt32** BMeasureApi::round512 (**BUInt32** s)

8.20.1 Macro Definition Documentation

8.20.1.1 BDEBUGL1

```
#define BDEBUGL1 0
```

8.20.1.2 BDEBUGL2

```
#define BDEBUGL2 0
```

8.21 DataFile.h File Reference

```
#include <BString.h>
#include <BFile.h>
#include <BMeasureLib.h>
#include <BoapMc1.h>
```

Classes

- class BMeasureApi::DataFile

Namespaces

- BMeasureApi

8.22 overview.dox File Reference

Index