LibTmsApi

2.0.0

Generated by Doxygen 1.8.5

Mon Dec 11 2017 13:24:55

Contents

Chapter 1

Main Page

Author

Dr Terry Barnaby

Version

2.0.0

Date

2013-02-01

1.1 Introduction

This document covers the BEAM LibTms software API for the CERN trajectory measurement system. This API provides the ability to control and receive data from the TMS System. The API is an object orientated API implemented in 'C++' with a number of object classes. The API operates over a network type interface using an RPC type mechanism.

The LibTms API makes use of the BEAM standard class library. The BEAM standard class library provides a small set of low level 'C++' classes for strings, lists and system interface functions. There is some brief information on the BEAM class library later on in this page.

1.2 Overview

Generally users of the system are only concerned with the top level, System Controller API. This is the API that control and data gathering clients use to control and gather data from the system. The System Controller API (TmsApi) is implemented using a simple, object orientated, RPC mechanism. Two main objects, the **Tms::TmsControl** and **Tms::TmsProcess** objects, provide the full API.

The TmsApi has been developed using the BOAP (BEAM Object Access Protocol). This provides a simple but powerful Object Orientated RPC mechanism. The TmsApi is written in a high level interface definition language (IDL). The bidl tool generates the client and server side 'C++' interface and implementation files for the API. These are then provided as a set of 'C++' header files and a binary library file for the clients to link to. The BOAP system employs a simple BOAP name server process that provides a translation between object names and IPAddress/Socket numbers. The BOAP name server runs on the System Controller. More information on the BOAP system can be found in the libBeam documentation.

There are two main Objects that are used by clients of the TMS API. They are the **Tms::TmsControl** and the **Tms::TmsProcess** objects. The **Tms::TmsControl** object is used for system configuration, testing and diagnostics. The

Tms::TmsProcess object is used for normal clients for Proton Synchrotron (PS) Cycle information configuration and data access. There is some example client code in the tmsExamples of the source code and displayed later on this page. These objects communicate through a network connection with the TmsServer process running on the TMS System Controller. The TMS System Controller operates as a multi-threaded process and can communicate with multiple clients simultaneously.

The TMS system takes most of its system timing signals from digital timing lines connected to the TMS rack hardware. The only timing information that external software needs to supply is the next cycle number and cycle type information. The cycle number is a 32bit unsigned number identifying the next Proton Synchrotron (PS) machine cycle. The cycle type is and ASCII string defining the type of BEAM present in the PS machine. The cycle type defines a set of state/phase tables to be loaded in order to measure the BEAM in the machine. The CERN client software needs to provide this information by calling the setNextCycle() function before the next PS cycle is initiated.

The TMS system keeps a library of state/phase tables indexed by the cycle type. These are loaded into the individual PUPE engines FPGA's during the CYCLE_STOP to CYCLE_START period. The API provides the **setControlInfo** and **delControlInfo** calls to maintain this database.

A client would generally use the **Tms::TmsProcess** object for its interface to the TMS system. It would use call getData() to fetch the required data from the system. There is also an event based data interface implemented using the requestData() call and dataEvent() event call.

Each of the TMS API calls return an error object. If there is an error, an appropriate error number will be given together with an ASCII string describing the error.

1.3 BEAM class library

The BEAM class library implements some basic low level classes and is used by the TMS API implementation itself. The main class functionaly includes:

- BString - A simple string class
- BList - A templated list class
- BArray - A templated array class
- BError - An error return class containg an integer and string
- BSocket - A Network socket access class
- BThread - A thread implementation class
- BPoll - A file descriptor event polling class
- BMutex - A mutex lock
- BRWLock - A read/write lock
- BSema - A semaphore
- BCondInt - An integer condition class
- BFile - A simple file access class
- BDir - A simple directory access class
- BEntry - A name/value pair list class
- BNameValue - A name/value pair class
- BRtc - A realtime clock
- BTimer - A simple timer class
- BUrl - URL access class

1.4 Examples

There are some examples of client applications using the TmsApi in the **tmsExamples** directory of the source code. A simple Data Access client example is listed bellow:

```
/*******************************************************************************
 *      TmsDataClient.cpp       TMS API example code for a Data Client
 *                      T.Barnaby,      BEAM Ltd,       2007-02-07
 *******************************************************************************
 *
 *      This is a very basic example of using the TmsApi from a clients perspective.
 *      It is designed to give an overview of using the API.
 */
#include <iostream>
#include <stdio.h>
#include <TmsD.h>
#include <TmsC.h>

using namespace Tms;
using namespace std;


// Function to reads some data
BError tmsTest(TmsProcess& tmsProcess){
        BError                    err;
        DataInfo                dataInfo;
        Data                        data;
        BUInt32                  cn = 0;
        BString                  ct;

        // Find out the current cycle number and type
        if(err = tmsProcess.getCycleInfo(cn, ct)){
                return err.set(1, BString("Error: Getting Cycle Number: ") + err.
      getString());
        }

        printf("Getting data for cycles starting at cycle: %u\n", cn);

        for(; ; cn++){
                // Setup dataInfo
                printf("GetData: Cycle Number: %u\n", cn);
                dataInfo.cycleNumber = cn;
                dataInfo.channel = 1;
                dataInfo.cyclePeriod = CyclePeriodEvent0;
                dataInfo.startTime     = 0;
                dataInfo.orbitNumber = 0;
                dataInfo.bunchNumber = 0;
                dataInfo.function       = DataFunctionRaw;
                dataInfo.argument       = 0;
                dataInfo.numValues     = 1024;
                dataInfo.beyondPeriod       = 1;

                if(err = tmsProcess.getData(dataInfo, data)){
                        return err.set(1, BString("Error: Getting Data: ") + err.
      getString());
                }
                printf("Data: NumValues: %d\n", data.numValues);
        }

        return err;
}

int main(int argc, char** argv){
        BError                    err;
        BString                  host = "localhost";
        TmsProcess            tmsProcess;

        if(argc == 2)
                host = argv[1];

        // Connect to the Process service
        if(err = tmsProcess.connectService(BString("//") + host + "/tmsProcess1")){
                cerr << "Error: " << err.getString() << "\n";
                return 1;
        }

        // Run a normal data gathering cycle as a normal client would.
        if(err = tmsTest(tmsProcess)){
                cerr << "Error: " << err.getString() << "\n";
                return 1;
        }

        return 0;
}
```

A simple Control client example is listed below:

```
/*******************************************************************************
 *      TmsControlClient1.cpp   TMS API example code
 *                      T.Barnaby,      BEAM Ltd,       2007-02-07
 *******************************************************************************
 *
 *      This is a very basic example of using the TmsApi from a clients perspective.
 *      It is designed to give an overview of using the API.
 */
#include <iostream>
#include <stdio.h>
#include <TmsD.h>
#include <TmsC.h>

using namespace Tms;
using namespace std;

const BUInt32    tmsStateNum = 16;
const BUInt32    tmsPickupNum = 40;

// Initialise and test the TMS system
BError tmsInit(TmsControl& tmsControl){
        BError                    err;
        ConfigInfo            configInfo;
        BIter                      i;
        BList<BError>              errorList;
        BList<NameValue>        nvList;
        BString                  version;

        // Get Version
        if(err = tmsControl.getVersion(version)){
                return err.set(1, BString("Error: initialising TMS: ") + err.
      getString());
        }
        cout << "Version: " << version << "\n";

        // Initialise TMS system
        if(err = tmsControl.init()){
                return err.set(1, BString("Error: initialising TMS: ") + err.
      getString());
        }

        // Test TMS system
        if(err = tmsControl.test(errorList)){
                return err.set(1, BString("Error: testing TMS: ") + err.
      getString());
        }

        for(errorList.start(i); !errorList.isEnd(i); errorList.next(i)){
                cout << "Warning: " << errorList[i].getString() << "\n";
        }

        // Get Status of TMS system
        if(err = tmsControl.getStatus(nvList)){
                return err.set(1, BString("Error: getting status: ") + err.
      getString());
        }

        for(nvList.start(i); !nvList.isEnd(i); nvList.next(i)){
                cout << nvList[i].name << ":\t" << nvList[i].value << "\n";
        }

        return err;
}

int main(int argc, char** argv){
        BError                    err;
        BString                  host = "localhost";
        TmsControl            tmsControl;
        TmsProcess            tmsProcess;

        if(argc == 2)
                host = argv[1];

        // Connect to the Control service
        if(err = tmsControl.connectService(BString("//") + host + "/tmsControl1")){
                cerr << "Error: " << err.getString() << "\n";
                return 1;
        }

        // Connect to the Process service
        if(err = tmsProcess.connectService(BString("//") + host + "/tmsProcess1")){
                cerr << "Error: " << err.getString() << "\n";
                return 1;
        }
        // Initialise and test the TMS system. Normally carried out by a configuration
        // and test client program.
        if(err = tmsInit(tmsControl)){
                cerr << "Error: " << err.getString() << "\n";
                return 1;
        }

        return 0;
}
```

A simple Control client to set the next cycle information example is listed below:

```
/*******************************************************************************
 *      TmsControlClient2.cpp   TMS API example code
 *                      T.Barnaby,      BEAM Ltd,       2007-02-07
 *******************************************************************************
 *
 *      This is a very basic example of using the TmsApi to set the
 *      TMS's cycleNumber and cycleType.
 *      It is designed to give an overview of using the API.
 */
#include <iostream>
#include <stdio.h>
#include <unistd.h>
#include <TmsD.h>
#include <TmsC.h>

using namespace Tms;
using namespace std;

// Loop sending next cycle information
BError tmsControlLoop(TmsControl& tmsControl){
        BError                  err;
        BUInt32                 cn = 0;
        BString                 ct = "Beam3";

        while(1){
                // Wait for next cycle information
                usleep(1200000);

                // Set next cycle information
                cn = cn + 1;
                ct = "Beam3";

                printf("SendNextCycle\n");
                // Send the next cycle information to the TMS server
                if(err = tmsControl.setNextCycle(cn, ct)){
                        cerr << "Error: " << err.getString() << "\n";
                }
        }

        return err;
}

int main(int argc, char** argv){
        BError                  err;
        BString                 host = "localhost";
        TmsControl              tmsControl;

        if(argc == 2)
                host = argv[1];

        // Connect to the Control service
        if(err = tmsControl.connectService(BString("//") + host + "/tmsControl1")){
                cerr << "Error: " << err.getString() << "\n";
                return 1;
        }

        // Set the network priority high
        if(err = tmsControl.setPriority(BSocket::PriorityHigh)){
                cerr << "Error: " << err.getString() << "\n";
                return 1;
        }

        // Set the TmsServer thread priority high
        if(err = tmsControl.setProcessPriority(PriorityHigh)){
                cerr << "Error: " << err.getString() << "\n";
                return 1;
        }

        if(err = tmsControlLoop(tmsControl)){
                cerr << "Error: " << err.getString() << "\n";
                return 1;
        }
        return 0;
}
```

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 Boapns Namespace Reference

Classes

- class Boapns
- class BoapEntry

Variables

- const BUInt32 apiVersion = 0

6.1.1 Variable Documentation

6.1.1.1 const BUInt32 Boapns::apiVersion = 0

6.2 Tms Namespace Reference

Classes

- class PuControl

  This class defines the parameters for a test data capture.
- class PuProcess

  This interface provides functions to configure and capture data from individual pick-up.
- class TmsControl

  This interface provides functions to control, test and get statistics from the TMS as a whole.
- class TmsProcess

  This interface provides functions to capture data from the TMS as a whole.
- class TmsEvent

  This interface provides functions for events to be sent to clients from the TMS as a whole.
- class CycleParamState
- class CycleParamEdit

  Cycle Parameter management class.
- class NameValue
- class PuChannel

  This class stores a Physical Pick-Up channel id.
- class PuStatus

This class stores the status of an individual Pick-Up.

- class ConfigInfo

  This class describes the configuration of the TMS.
- class DataInfo

  This class defines the data to be acquired and/or fetched.
- class DataValue

  This is the definition of a single data value.
- class Data

  This class stores the raw data.
- class PuStateTable

  This class defines the Pick-Up state table.
- class CycleParam

  This class defines the parameters for a PS processing cycle.
- class CycleParamItem
- class TestCaptureInfo

  This class defines the parameters for a test data capture.
- class PupeConfig
- class CycleInformationPeriod

  Cycle information.
- class CycleInformation
- class CycleTypeInformationPeriod

  Cycle Type information.
- class CycleTypeInformation
- class Simulation
- class TmsEventServerList
- union TmsState

  The Tms State entry.
- union TmsPhase

  The Tms Phase Table Entry.
- class CycleParamDb

  Internal CycleParameter management class.

Enumerations

- enum CyclePeriod {
  CyclePeriodAll, CyclePeriodCalibration, CyclePeriodEvent0, CyclePeriodEvent1,
  CyclePeriodEvent2, CyclePeriodEvent3, CyclePeriodEvent4, CyclePeriodEvent5,
  CyclePeriodEvent6, CyclePeriodEvent7, CyclePeriodEvent8, CyclePeriodEvent9 }
- enum DataType { DataTypeRaw }
- enum DataFunction {
  DataFunctionRaw, DataFunctionMean, DataFunctionMeanAll, DataFunctionMean0,
  DataFunctionMean1 }
- enum TestOutput { TestOutputFrefLocal, TestOutputPllL1, TestOutputPllL2 }
- enum Priority { PriorityLow, PriorityNormal, PriorityHigh }
- enum TimingSig {
  TimingSigClock = 0x01, TimingSigCycleStart = 0x02, TimingSigCycleStop = 0x04, TimingSigCalStart = 0x08,
  TimingSigCalStop = 0x10, TimingSigInjection = 0x20, TimingSigHChange = 0x40, TimingSigFRef = 0x80 }

  The timing signal bits.
- enum CaptureClock {
  ClkAdcDiv_1 = 0x00, ClkAdcDiv_2 = 0x01, ClkAdcDiv_5 = 0x02, ClkAdcDiv_10 = 0x03,
  ClkAdcDiv_20 = 0x04, ClkAdcDiv_50 = 0x05, ClkAdcDiv_100 = 0x06, ClkAdcDiv_200 = 0x07,
  ClkAdcDiv_500 = 0x08, ClkAdcDiv_1000 = 0x09, ClkAdcDiv_2000 = 0x0A, ClkAdcDiv_5000 = 0x0B,
  ClkAdcDiv_10000 = 0x0C, ClkAdcDiv_20000 = 0x0D, ClkAdcDiv_50000 = 0x0E, ClkAdcDiv_100000 = 0x0F,
  ClkMs = 0x10, ClkFref = 0x11 }

  The Diagnostics Capture Clock settings.

Variables

- const BUInt32 apiVersion = 0
- const unsigned int tmsNumPickups = 40

  The default number of pick ups.
- const unsigned int tmsPhaseTableSize = 512

  The size of the Phase Table.

6.2.1 Enumeration Type Documentation

6.2.1.1 enum Tms::CaptureClock

The Diagnostics Capture Clock settings.

Enumerator

ClkAdcDiv_1 ADC Clock.

ClkAdcDiv_2 ADC Clock divided by 2.

ClkAdcDiv_5 ADC Clock divided by 5.

ClkAdcDiv_10 ADC Clock divided by 10.

ClkAdcDiv_20 ADC Clock divided by 20.

ClkAdcDiv_50 ADC Clock divided by 50.

ClkAdcDiv_100 ADC Clock divided by 100.

ClkAdcDiv_200 ADC Clock divided by 200.

ClkAdcDiv_500 ADC Clock divided by 500.

ClkAdcDiv_1000 ADC Clock divided by 1000.

ClkAdcDiv_2000 ADC Clock divided by 2000.

ClkAdcDiv_5000 ADC Clock divided by 5000.

ClkAdcDiv_10000 ADC Clock divided by 10000.

ClkAdcDiv_20000 ADC Clock divided by 20000.

ClkAdcDiv_50000 ADC Clock divided by 50000.

ClkAdcDiv_100000 ADC Clock divided by 100000.

ClkMs Millisecond Clock.

ClkFref FREF.

6.2.1.2 enum Tms::CyclePeriod

Enumerator

CyclePeriodAll

CyclePeriodCalibration

CyclePeriodEvent0

CyclePeriodEvent1

CyclePeriodEvent2

CyclePeriodEvent3

CyclePeriodEvent4

CyclePeriodEvent5

CyclePeriodEvent6

CyclePeriodEvent7

CyclePeriodEvent8

CyclePeriodEvent9

6.2.1.3 enum Tms::DataFunction

Enumerator

DataFunctionRaw

DataFunctionMean

DataFunctionMeanAll

DataFunctionMean0

DataFunctionMean1

6.2.1.4 enum Tms::DataType

Enumerator

DataTypeRaw

6.2.1.5 enum Tms::Priority

Enumerator

PriorityLow

PriorityNormal

PriorityHigh

6.2.1.6 enum Tms::TestOutput

Enumerator

TestOutputFrefLocal

TestOutputPllL1

TestOutputPllL2

6.2.1.7 enum Tms::TimingSig

The timing signal bits.

Enumerator

TimingSigClock 10MHz System Clock

TimingSigCycleStart CYCLE_START event.

TimingSigCycleStop CYCLE_STOP event.

TimingSigCalStart CAL_START event.

TimingSigCalStop CAL_STOP event.

TimingSigInjection INJECTION event.

TimingSigHChange HCHANGE event.

TimingSigFRef FREF signal.

6.2.2 Variable Documentation

6.2.2.1 const BUInt32 Tms::apiVersion = 0

6.2.2.2 const unsigned int Tms::tmsNumPickups = 40

The default number of pick ups.

6.2.2.3 const unsigned int Tms::tmsPhaseTableSize = 512

The size of the Phase Table.

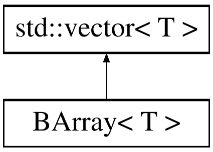
Chapter 7

Class Documentation

7.1 BArray< T > Class Template Reference

```
#include <BArray.h>
```

Inheritance diagram for BArray< T >:



Public Types

- typedef int(∗ SortFunc)(T &a, T &b)

  Prototype for sorting function.

Public Member Functions

- BArray ()
- BArray (BSize size, T value=T())
- BArray (const BArray &array)
- BUInt number () const
- void append (const T &value)
- void append (const BArray< T > &array)
- void insert (BUInt pos, const T &value)
- void del (BUInt pos, BUInt num=1)
- T & rear ()
- void sort ()

7.1.1 Detailed Description

template<class T>class BArray< T >

Template based Array class. This is based on the Standard C++ library vector class and has all of the functionality of that class.

7.1.2 Member Typedef Documentation

7.1.2.1 template<class T> typedef int(∗ BArray< T >::SortFunc)(T &a, T &b)

Prototype for sorting function.

7.1.3 Constructor & Destructor Documentation

7.1.3.1 template<class T> BArray< T >::BArray ( ) `[inline]`

7.1.3.2 template<class T> BArray< T >::BArray (BSize *size*, T *value* = `T()`) `[inline]`

7.1.3.3 template<class T> BArray< T >::BArray (const BArray< T > & *array*) `[inline]`

7.1.4 Member Function Documentation

7.1.4.1 template<class T> void BArray< T >::append (const T & *value*) `[inline]`

7.1.4.2 template<class T> void BArray< T >::append (const BArray< T > & *array*)

7.1.4.3 template<class T> void BArray< T >::del (BUInt *pos*, BUInt *num* = `1`) `[inline]`

7.1.4.4 template<class T> void BArray< T >::insert (BUInt *pos*, const T & *value*) `[inline]`

7.1.4.5 template<class T> BUInt BArray< T >::number ( ) const `[inline]`

7.1.4.6 template<class T> T& BArray< T >::rear ( ) `[inline]`

7.1.4.7 template<class T> void BArray< T >::sort ( ) `[inline]`

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

- /src/cern/tms/beam/libBeam/BArray.h

7.2 BAtomic< Type > Class Template Reference

BAtomic class.

`#include <BAtomic.h>`

Public Member Functions

- BAtomic (Type value=0)
- Type getValue () const
- Type add (long value)
- Type operator++ (int)
- Type operator++ ()
- Type operator-- (int)
- Type operator-- ()
- operator Type () const

Private Attributes

- Type ovalue

7.2.1 Detailed Description

template<class Type>class BAtomic< Type >

BAtomic class.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 template<class Type > BAtomic< Type >::BAtomic (Type *value* = 0) [inline]

7.2.3 Member Function Documentation

7.2.3.1 template<class Type > Type BAtomic< Type >::add (long *value*) [inline]

7.2.3.2 template<class Type > Type BAtomic< Type >::getValue ( ) const [inline]

7.2.3.3 template<class Type > BAtomic< Type >::operator Type ( ) const [inline]

7.2.3.4 template<class Type > Type BAtomic< Type >::operator++ (int) [inline]

7.2.3.5 template<class Type > Type BAtomic< Type >::operator++ ( ) [inline]

7.2.3.6 template<class Type > Type BAtomic< Type >::operator-- (int) [inline]

7.2.3.7 template<class Type > Type BAtomic< Type >::operator-- ( ) [inline]

7.2.4 Member Data Documentation

7.2.4.1 template<class Type > Type BAtomic< Type >::ovalue [mutable], [private]

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

- /src/cern/tms/beam/libBeam/BAtomic.h

7.3 BAtomicCount Class Reference

BAtomicCount class.

```
#include <BAtomicCount.h>
```

Public Member Functions

- BAtomicCount (long value=0)
- long getValue () const
- long add (long value)
- long operator++ (int)
- long operator++ ()
- long operator-- (int)
- long operator-- ()
- operator long () const

Private Attributes

- _Atomic_word ovalue

7.3.1 Detailed Description

BAtomicCount class.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 BAtomicCount::BAtomicCount (long *value* = 0) [inline]

7.3.3 Member Function Documentation

7.3.3.1 long BAtomicCount::add (long *value*) [inline]

7.3.3.2 long BAtomicCount::getValue ( ) const [inline]

7.3.3.3 BAtomicCount::operator long ( ) const [inline]

7.3.3.4 long BAtomicCount::operator++ (int) [inline]

7.3.3.5 long BAtomicCount::operator++ ( ) [inline]

7.3.3.6 long BAtomicCount::operator-- (int) [inline]

7.3.3.7 long BAtomicCount::operator-- ( ) [inline]

7.3.4 Member Data Documentation

7.3.4.1 _Atomic_word BAtomicCount::ovalue [mutable], [private]

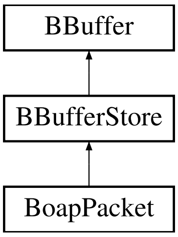
The documentation for this class was generated from the following file:

- /src/cern/tms/beam/libBeam/BAtomicCount.h

7.4 BBuffer Class Reference

```
#include <BBuffer.h>
```

Inheritance diagram for BBuffer:



Public Member Functions

- BBuffer (BUInt size=0)

    Create and manipulate a data buffer. On creation the buffer size defaults to 1024 bytes.
- ∼BBuffer ()
- int setSize (BUInt32 size)

Sets the bufer size.

- int setData (const void ∗data, BUInt32 size)

  Sets buffer data resized to contain the data.
- int writeData (BUInt32 pos, const void ∗data, BUInt32 size)

  Writes data into buffer from offset pos.
- char ∗ data ()

  The data.
- BUInt32 size ()

  Size of the buffer in bytes.
- int resize (BUInt32 size)

  Alternative to setSize()

Protected Attributes

- BUInt32 odataSize
- char ∗ odata
- BUInt32 osize

7.4.1 Constructor & Destructor Documentation

7.4.1.1 BBuffer::BBuffer (BUInt *size* = 0)

Create and manipulate a data buffer. On creation the buffer size defaults to 1024 bytes.

7.4.1.2 BBuffer::∼BBuffer ( )

7.4.2 Member Function Documentation

7.4.2.1 char ∗ BBuffer::data ( )

The data.

7.4.2.2 int BBuffer::resize (BUInt32 *size*) [inline]

Alternative to setSize()

7.4.2.3 int BBuffer::setData (const void ∗ *data,* BUInt32 *size*)

Sets buffer data resized to contain the data.

7.4.2.4 int BBuffer::setSize (BUInt32 *size*)

Sets the bufer size.

7.4.2.5 BUInt32 BBuffer::size ( )

Size of the buffer in bytes.

7.4.2.6 int BBuffer::writeData (**BUInt32** *pos,* const void ∗ *data,* **BUInt32** *size*)

Writes data into buffer from offset pos.

7.4.3 Member Data Documentation

7.4.3.1 char∗ BBuffer::odata [protected]

7.4.3.2 BUInt32 BBuffer::odataSize [protected]

7.4.3.3 BUInt32 BBuffer::osize [protected]

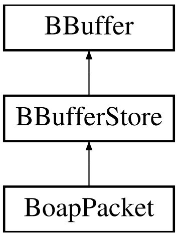
The documentation for this class was generated from the following files:

- /src/cern/tms/beam/libBeam/BBuffer.h
- /src/cern/tms/beam/libBeam/BBuffer.cpp

7.5 BBufferStore Class Reference

`#include <BBuffer.h>`

Inheritance diagram for BBufferStore:



Public Member Functions

- BBufferStore (BUInt size=0, int swapBytes=BBigEndian)
- ∼BBufferStore ()
- BUInt32 getPos ()
- void setPos (BUInt32 pos)
- BString getHexString ()
- void setHexString (BString s)
- int push (BInt8 v)
- int push (BUInt8 v)
- int push (BInt16 v)
- int push (BUInt16 v)
- int push (BInt32 v)
- int push (BUInt32 v)
- int push (BInt64 v)
- int push (BUInt64 v)
- int push (BFloat32 v)
- int push (BFloat64 v)
- int push (const BString &v)
- int push (const BError &v)
- int push (const BTimeStamp &v)

- int push (const BComplex &v)
- int push (BUInt32 nBytes, const void ∗data, const char ∗swapType="1")
- int pop (BInt8 &v)
- int pop (BUInt8 &v)
- int pop (BInt16 &v)
- int pop (BUInt16 &v)
- int pop (BInt32 &v)
- int pop (BUInt32 &v)
- int pop (BInt64 &v)
- int pop (BUInt64 &v)
- int pop (BFloat32 &v)
- int pop (BFloat64 &v)
- int pop (BString &v)
- int pop (BError &v)
- int pop (BTimeStamp &v)
- int pop (BComplex &v)
- int pop (BUInt32 nBytes, void ∗data, const char ∗swapType="1")

Protected Attributes

- BUInt32 opos
- int oswapBytes

7.5.1 Constructor & Destructor Documentation

7.5.1.1 BBufferStore::BBufferStore (**BUInt** *size* = 0, int *swapBytes* = **BBigEndian**)

7.5.1.2 BBufferStore::∼BBufferStore ( )

7.5.2 Member Function Documentation

7.5.2.1 **BString** BBufferStore::getHexString ( )

7.5.2.2 **BUInt32** BBufferStore::getPos ( )

7.5.2.3 int BBufferStore::pop (**BInt8** & *v*)

7.5.2.4 int BBufferStore::pop (**BUInt8** & *v*)

7.5.2.5 int BBufferStore::pop (**BInt16** & *v*)

7.5.2.6 int BBufferStore::pop (**BUInt16** & *v*)

7.5.2.7 int BBufferStore::pop (**BInt32** & *v*)

7.5.2.8 int BBufferStore::pop (**BUInt32** & *v*)

7.5.2.9 int BBufferStore::pop (**BInt64** & *v*)

7.5.2.10 int BBufferStore::pop (**BUInt64** & *v*)

7.5.2.11 int BBufferStore::pop (**BFloat32** & *v*)

7.5.2.12 int BBufferStore::pop (**BFloat64** & *v*)

7.5.2.13 int BBufferStore::pop (BString & *v*)

7.5.2.14 int BBufferStore::pop (BError & *v*)

7.5.2.15 int BBufferStore::pop (BTimeStamp & *v*)

7.5.2.16 int BBufferStore::pop (BComplex & *v*)

7.5.2.17 int BBufferStore::pop (BUInt32 *nBytes*, void ∗ *data*, const char ∗ *swapType* = "1")

7.5.2.18 int BBufferStore::push (BInt8 *v*)

7.5.2.19 int BBufferStore::push (BUInt8 *v*)

7.5.2.20 int BBufferStore::push (BInt16 *v*)

7.5.2.21 int BBufferStore::push (BUInt16 *v*)

7.5.2.22 int BBufferStore::push (BInt32 *v*)

7.5.2.23 int BBufferStore::push (BUInt32 *v*)

7.5.2.24 int BBufferStore::push (BInt64 *v*)

7.5.2.25 int BBufferStore::push (BUInt64 *v*)

7.5.2.26 int BBufferStore::push (BFloat32 *v*)

7.5.2.27 int BBufferStore::push (BFloat64 *v*)

7.5.2.28 int BBufferStore::push (const BString & *v*)

7.5.2.29 int BBufferStore::push (const BError & *v*)

7.5.2.30 int BBufferStore::push (const BTimeStamp & *v*)

7.5.2.31 int BBufferStore::push (const BComplex & *v*)

7.5.2.32 int BBufferStore::push (BUInt32 *nBytes*, const void ∗ *data*, const char ∗ *swapType* = "1")

7.5.2.33 void BBufferStore::setHexString (BString *s*)

7.5.2.34 void BBufferStore::setPos (BUInt32 *pos*)

7.5.3 Member Data Documentation

7.5.3.1 BUInt32 BBufferStore::opos [protected]

7.5.3.2 int BBufferStore::oswapBytes [protected]

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

- /src/cern/tms/beam/libBeam/BBuffer.h
- /src/cern/tms/beam/libBeam/BBuffer.cpp

7.6 BComms Class Reference

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

Public Types

- enum Wait { WaitNone = 0x00, WaitRead = 0x01, WaitWrite = 0x02, WaitError = 0x04 }

Public Member Functions

- BComms ()
- virtual ∼BComms ()
- virtual BError init ()
- virtual BError setPacketMode (Bool packetMode)

  Set packet mode.
- virtual Bool packetMode ()

  Device is in packet mode.
- virtual BError setTimeout (BInt timeoutMs)

  Set communication timeout.
- virtual BUInt writeAvailable ()
- virtual BError write (const void ∗data, BUInt32 nBytes, BUInt32 &nTrans)=0
- virtual BUInt readAvailable ()
- virtual BError read (void ∗data, BUInt32 num, BUInt32 &nTrans)=0
- virtual BError wait (BUInt8 events, BInt timeout=-1, BUInt32 num=1)
- virtual void eventQueue (BEventQueue ∗eventQueue, BInt32 event, BUInt num=1)

Protected Attributes

- Bool opacketMode
- BInt32 otimeout
- BEventQueue ∗ oeventQueue
- BInt32 oevent
- BUInt oeventNum

7.6.1 Member Enumeration Documentation

7.6.1.1 enum BComms::Wait

Enumerator

WaitNone

WaitRead

WaitWrite

WaitError

7.6.2 Constructor & Destructor Documentation

7.6.2.1 BComms::BComms ( )

7.6.2.2 BComms::∼BComms ( ) [virtual]

7.6.3 Member Function Documentation

7.6.3.1 void BComms::eventQueue (BEventQueue ∗ *eventQueue,* BInt32 *event,* BUInt *num* = 1) [virtual]

7.6.3.2 BError BComms::init ( ) [virtual]

7.6.3.3 Bool BComms::packetMode ( ) [virtual]

Device is in packet mode.

7.6.3.4 virtual BError BComms::read (void ∗ *data,* BUInt32 *num,* BUInt32 & *nTrans*) [pure virtual]

7.6.3.5 BUInt BComms::readAvailable ( ) [virtual]

7.6.3.6 BError BComms::setPacketMode (Bool *packetMode*) [virtual]

Set packet mode.

7.6.3.7 BError BComms::setTimeout (BInt *timeoutMs*) [virtual]

Set communication timeout.

7.6.3.8 BError BComms::wait (BUInt8 *events,* BInt *timeout* = −1, BUInt32 *num* = 1) [virtual]

7.6.3.9 virtual BError BComms::write (const void ∗ *data,* BUInt32 *nBytes,* BUInt32 & *nTrans*) [pure virtual]

7.6.3.10 BUInt BComms::writeAvailable ( ) [virtual]

7.6.4 Member Data Documentation

7.6.4.1 BInt32 BComms::oevent [protected]

7.6.4.2 BUInt BComms::oeventNum [protected]

7.6.4.3 BEventQueue∗ BComms::oeventQueue [protected]

7.6.4.4 Bool BComms::opacketMode [protected]

7.6.4.5 BInt32 BComms::otimeout [protected]

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

- /src/cern/tms/beam/libBeam/BComms.h
- /src/cern/tms/beam/libBeam/BComms.cpp

7.7 BCond Class Reference

```
#include <BCond.h>
```

Public Member Functions

- BCond ()

    Thread conditional variable.
- ∼BCond ()
- int signal ()
- int wait ()
- int timedWait (int timeOutUs)

Private Attributes

- pthread_mutex_t omutex
- pthread_cond_t ocond

7.7.1 Constructor & Destructor Documentation

7.7.1.1 BCond::BCond ( )

Thread conditional variable.

7.7.1.2 BCond::∼BCond ( )

7.7.2 Member Function Documentation

7.7.2.1 int BCond::signal ( )

7.7.2.2 int BCond::timedWait (int *timeOutUs*)

7.7.2.3 int BCond::wait ( )

7.7.3 Member Data Documentation

7.7.3.1 pthread_cond_t BCond::ocond `[private]`

7.7.3.2 pthread_mutex_t BCond::omutex `[private]`

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

- /src/cern/tms/beam/libBeam/BCond.h
- /src/cern/tms/beam/libBeam/BCond.cpp

7.8 BCondBool Class Reference

Thread conditional boolean.

```
#include <BCondInt.h>
```

Public Member Functions

- BCondBool ()
- ∼BCondBool ()
- int set ()

  Set value. Wakes waiting.
- int clear ()

  Clear Value.
- int value ()

  Current value.
- int wait ()

  Wait until value is true.
- int timedWait (int timeOutUs)

  Wait until set, with timeout.
- operator int ()

Private Attributes

- pthread_mutex_t omutex
- pthread_cond_t ocond
- int ovalue

7.8.1 Detailed Description

Thread conditional boolean.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 BCondBool::BCondBool ( )

7.8.2.2 BCondBool::∼BCondBool ( )

7.8.3 Member Function Documentation

7.8.3.1 int BCondBool::clear ( )

Clear Value.

7.8.3.2 BCondBool::operator int ( ) [inline]

7.8.3.3 int BCondBool::set ( )

Set value. Wakes waiting.

7.8.3.4 int BCondBool::timedWait (int *timeOutUs*)

Wait until set, with timeout.

7.8.3.5 int BCondBool::value ( )

Current value.

7.8.3.6 int BCondBool::wait ( )

Wait until value is true.

7.8.4 Member Data Documentation

7.8.4.1 pthread_cond_t BCondBool::ocond [private]

7.8.4.2 pthread_mutex_t BCondBool::omutex [private]

7.8.4.3 int BCondBool::ovalue [private]

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

- /src/cern/tms/beam/libBeam/BCondInt.h
- /src/cern/tms/beam/libBeam/BCondInt.cpp

7.9 BCondInt Class Reference

Thread conditional value.

`#include <BCondInt.h>`

Public Member Functions

- BCondInt ()
- ∼BCondInt ()
- void setValue (BInt value)

  Set the value. Wakes waiting.
- BInt value () const

  Current value.
- BInt increment (BInt v=1)

  Increment. Wakes waiting.
- BInt decrement (BInt v=1)

  Decrement. Wakes waiting.
- Bool waitMoreThanOrEqual (BInt v, Bool decrement=0, BTimeout timeoutUs=BTimeoutForever)

  Wait until value is at least the value given.
- Bool waitLessThanOrEqual (BInt v, Bool increment=0, BTimeout timeoutUs=BTimeoutForever)

  Wait until value is equal to or below the value given.
- Bool waitLessThan (BInt v, BTimeout timeoutUs=BTimeoutForever)

  Wait until value is equal to or below the value given.
- void operator+= (int v)

  Add to value. Wakes waiting.
- void operator-= (int v)

  Subtract from value. Wakes waiting.
- void operator++ (int)

  Increment value. Wakes waiting.
- void operator-- (int)

  Decrement value. Wakes waiting.

Private Attributes

- pthread_mutex_t omutex
- pthread_cond_t ocond
- BInt ovalue

7.9.1 Detailed Description

Thread conditional value.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 BCondInt::BCondInt ( )

7.9.2.2 BCondInt::∼BCondInt ( )

7.9.3 Member Function Documentation

7.9.3.1 BInt BCondInt::decrement (BInt *v* = 1)

Decrement. Wakes waiting.

7.9.3.2 BInt BCondInt::increment (BInt *v* = 1)

Increment. Wakes waiting.

7.9.3.3 void BCondInt::operator++ (int) [inline]

Increment value. Wakes waiting.

7.9.3.4 void BCondInt::operator+= (int *v*) [inline]

Add to value. Wakes waiting.

7.9.3.5 void BCondInt::operator-- (int) [inline]

Decrement value. Wakes waiting.

7.9.3.6 void BCondInt::operator-= (int *v*) [inline]

Subtract from value. Wakes waiting.

7.9.3.7 void BCondInt::setValue (BInt *value*)

Set the value. Wakes waiting.

7.9.3.8 BInt BCondInt::value ( ) const

Current value.

7.9.3.9 Bool BCondInt::waitLessThan (BInt *v*, BTimeout *timeoutUs* = BTimeoutForever)

Wait until value is equal to or below the value given.

7.9.3.10 Bool BCondInt::waitLessThanOrEqual (BInt *v*, Bool *increment* = `0`, BTimeout *timeoutUs* = BTimeoutForever)

Wait until value is equal to or below the value given.

7.9.3.11 Bool BCondInt::waitMoreThanOrEqual (BInt *v*, Bool *decrement* = `0`, BTimeout *timeoutUs* = BTimeoutForever)

Wait until value is at least the value given.

7.9.4 Member Data Documentation

7.9.4.1 pthread_cond_t BCondInt::ocond `[private]`

7.9.4.2 pthread_mutex_t BCondInt::omutex `[private]`

7.9.4.3 BInt BCondInt::ovalue `[private]`

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

- /src/cern/tms/beam/libBeam/BCondInt.h
- /src/cern/tms/beam/libBeam/BCondInt.cpp

7.10 BCondResource Class Reference

Resource lock.

`#include <BCondInt.h>`

Public Member Functions

- BCondResource ()
- ∼BCondResource ()
- int lock (uint32_t timeOutUs=0)

  Lock the resource, will wait for all usage to be 0.
- int unlock ()

  Unlock the resource.
- int start (uint32_t timeOutUs=0)

  Start using the resource.
- int end ()

  Finish using the resource.
- int locked ()
- int inUse ()

Private Attributes

- pthread_mutex_t omutex
- pthread_cond_t ocond
- int olock
- int ouse

7.10.1 Detailed Description

Resource lock.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 BCondResource::BCondResource ( )

7.10.2.2 BCondResource::∼BCondResource ( )

7.10.3 Member Function Documentation

7.10.3.1 int BCondResource::end ( )

Finish using the resource.

7.10.3.2 int BCondResource::inUse ( )

7.10.3.3 int BCondResource::lock (uint32_t *timeOutUs* = `0`)

Lock the resource, will wait for all usage to be 0.

7.10.3.4 int BCondResource::locked ( )

7.10.3.5 int BCondResource::start (uint32_t *timeOutUs* = `0`)

Start using the resource.

7.10.3.6 int BCondResource::unlock ( )

Unlock the resource.

7.10.4 Member Data Documentation

7.10.4.1 pthread_cond_t BCondResource::ocond `[private]`

7.10.4.2 int BCondResource::olock `[private]`

7.10.4.3 pthread_mutex_t BCondResource::omutex `[private]`

7.10.4.4 int BCondResource::ouse `[private]`

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

- /src/cern/tms/beam/libBeam/BCondInt.h
- /src/cern/tms/beam/libBeam/BCondInt.cpp

7.11 BCondValue Class Reference

Thread conditional value.

`#include <BCondInt.h>`

Public Member Functions

- BCondValue ()
- ∼BCondValue ()
- void setValue (int value)

  Set the value. Wakes waiting.
- int value ()

  Current value.
- int increment (int v=1)

  Increment. Wakes waiting.
- int decrement (int v=1)

  Decrement. Wakes waiting.
- int waitMoreThanOrEqual (int v, int decrement=0, int timeOutUs=0)

  Wait until value is at least the value given.
- int waitLessThanOrEqual (int v, int increment=0, int timeOutUs=0)

  Wait until value is equal to or below the value given.
- int waitLessThan (int v, int timeOutUs=0)

  Wait until value is equal to or below the value given.
- void operator+= (int v)

  Add to value. Wakes waiting.
- void operator-= (int v)

  Subtract from value. Wakes waiting.
- void operator++ (int)

  Increment value. Wakes waiting.
- void operator-- (int)

  Decrement value. Wakes waiting.

Private Attributes

- pthread_mutex_t omutex
- pthread_cond_t ocond
- int ovalue

7.11.1 Detailed Description

Thread conditional value.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 BCondValue::BCondValue ( )

7.11.2.2 BCondValue::∼BCondValue ( )

7.11.3 Member Function Documentation

7.11.3.1 int BCondValue::decrement (int *v* = `1`)

Decrement. Wakes waiting.

7.11.3.2 int BCondValue::increment (int *v* = `1`)

Increment. Wakes waiting.

7.11.3.3 void BCondValue::operator++ (int) [inline]

Increment value. Wakes waiting.

7.11.3.4 void BCondValue::operator+= (int *v*) [inline]

Add to value. Wakes waiting.

7.11.3.5 void BCondValue::operator-- (int) [inline]

Decrement value. Wakes waiting.

7.11.3.6 void BCondValue::operator-= (int *v*) [inline]

Subtract from value. Wakes waiting.

7.11.3.7 void BCondValue::setValue (int *value*)

Set the value. Wakes waiting.

7.11.3.8 int BCondValue::value ( )

Current value.

7.11.3.9 int BCondValue::waitLessThan (int *v*, int *timeOutUs* = 0)

Wait until value is equal to or below the value given.

7.11.3.10 int BCondValue::waitLessThanOrEqual (int *v*, int *increment* = 0, int *timeOutUs* = 0)

Wait until value is equal to or below the value given.

7.11.3.11 int BCondValue::waitMoreThanOrEqual (int *v*, int *decrement* = 0, int *timeOutUs* = 0)

Wait until value is at least the value given.

7.11.4 Member Data Documentation

7.11.4.1 pthread_cond_t BCondValue::ocond [private]

7.11.4.2 pthread_mutex_t BCondValue::omutex [private]

7.11.4.3 int BCondValue::ovalue [private]

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

- /src/cern/tms/beam/libBeam/BCondInt.h
- /src/cern/tms/beam/libBeam/BCondInt.cpp

7.12 BCondWrap Class Reference

```
#include <BCondInt.h>
```

Public Member Functions

- BCondWrap ()
- ∼BCondWrap ()
- void setValue (uint32_t value)

  Set the value. Wakes waiting.
- uint32_t value ()

  Current value.
- uint32_t increment (uint32_t v=1)

  Increment. Wakes waiting.
- uint32_t decrement (uint32_t v=1)

  Decrement. Wakes waiting.
- int waitMoreThanOrEqual (uint32_t v, uint32_t decrement=0, uint32_t timeOutUs=0)

  Wait until value is at least the value given.
- int waitLessThanOrEqual (uint32_t v, uint32_t increment=0, uint32_t timeOutUs=0)

  Wait until value is equal to or below the value given.
- int waitLessThan (uint32_t v, uint32_t timeOutUs=0)

  Wait until value is equal to or below the value given.
- void operator+= (int v)

  Add to value. Wakes waiting.
- void operator-= (int v)

  Subtract from value. Wakes waiting.
- void operator++ (int)

  Increment value. Wakes waiting.
- void operator-- (int)

  Decrement value. Wakes waiting.

Private Member Functions

- int diff (uint32_t v)

Private Attributes

- pthread_mutex_t omutex
- pthread_cond_t ocond
- uint32_t ovalue

7.12.1 Constructor & Destructor Documentation

7.12.1.1 BCondWrap::BCondWrap ( )

7.12.1.2 BCondWrap::∼BCondWrap ( )

7.12.2 Member Function Documentation

7.12.2.1 uint32_t BCondWrap::decrement (uint32_t *v* = `1`)

Decrement. Wakes waiting.

7.12.2.2 int BCondWrap::diff (uint32_t *v*) [private]

7.12.2.3 uint32_t BCondWrap::increment (uint32_t *v* = 1)

Increment. Wakes waiting.

7.12.2.4 void BCondWrap::operator++ (int) [inline]

Increment value. Wakes waiting.

7.12.2.5 void BCondWrap::operator+= (int *v*) [inline]

Add to value. Wakes waiting.

7.12.2.6 void BCondWrap::operator-- (int) [inline]

Decrement value. Wakes waiting.

7.12.2.7 void BCondWrap::operator-= (int *v*) [inline]

Subtract from value. Wakes waiting.

7.12.2.8 void BCondWrap::setValue (uint32_t *value*)

Set the value. Wakes waiting.

7.12.2.9 uint32_t BCondWrap::value ( )

Current value.

7.12.2.10 int BCondWrap::waitLessThan (uint32_t *v*, uint32_t *timeOutUs* = 0)

Wait until value is equal to or below the value given.

7.12.2.11 int BCondWrap::waitLessThanOrEqual (uint32_t *v*, uint32_t *increment* = 0, uint32_t *timeOutUs* = 0)

Wait until value is equal to or below the value given.

7.12.2.12 int BCondWrap::waitMoreThanOrEqual (uint32_t *v*, uint32_t *decrement* = 0, uint32_t *timeOutUs* = 0)

Wait until value is at least the value given.

7.12.3 Member Data Documentation

7.12.3.1 pthread_cond_t BCondWrap::ocond [private]

7.12.3.2 pthread_mutex_t BCondWrap::omutex [private]

7.12.3.3 uint32_t BCondWrap::ovalue [private]

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

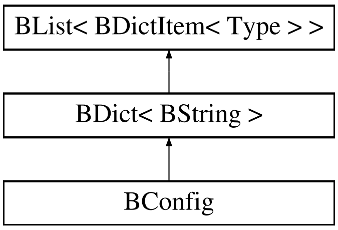
- /src/cern/tms/beam/libBeam/BCondInt.h
- /src/cern/tms/beam/libBeam/BCondInt.cpp

7.13 BConfig Class Reference

This class implements the configuration file access.

```
#include <BConfig.h>
```

Inheritance diagram for BConfig:



Public Member Functions

- BError open (BString fileName, BString mode="r")
- void close ()
- BError read ()
- BError write ()
- BString findValue (BString name)
- BString fileName ()

Private Attributes

- BMutex olock
- BString ofileName
- BFile ofile

Additional Inherited Members

7.13.1 Detailed Description

This class implements the configuration file access.

7.13.2 Member Function Documentation

7.13.2.1 void BConfig::close ( )

7.13.2.2 BString BConfig::fileName ( )

7.13.2.3 BString BConfig::findValue (BString *name*)

7.13.2.4 BError BConfig::open (BString *fileName*, BString *mode* = "r")

7.13.2.5 BError BConfig::read ( )

7.13.2.6 BError BConfig::write ( )

7.13.3 Member Data Documentation

7.13.3.1 BFile BConfig::ofile [private]

7.13.3.2 BString BConfig::ofileName [private]

7.13.3.3 BMutex BConfig::olock [private]

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

- /src/cern/tms/beam/libBeam/BConfig.h
- /src/cern/tms/beam/libBeam/BConfig.cpp

7.14 BDate Class Reference

`#include <BDate.h>`

Public Member Functions

- BDate (int year=0, int month=1, int day=1)
- BDate (BString str)
- ∼BDate ()
- void clear ()

  Clear the date/time.
- void setFirst ()

  Set the first date available.
- void setLast ()

  Set the last date available.
- void set (time_t time)

  Set time using Unix time (seconds from 1970-01-01)
- void set (int year=0, int month=1, int day=1)
- void setYDay (int year=0, int yday=0)
- void setNow ()

  Set the timeStamp to now.
- int year ()
- int yday ()
- int month ()
- int day ()
- void getDate (int &year, int &mon, int &day)
- BString getString ()

  Get the time as an ISO date/time string.
- BString getStringFormatted (BString format)

  Gets the time in a string form as per the format. Format syntax as per strftime()
- BError setString (BString str)

  Set the time from an ISO date/time.
- int isSet ()

- int compare (const BDate &date) const

  Compare two dates.
- operator BString ()
- int operator== (const BDate &date) const
- int operator!= (const BDate &date) const
- int operator> (const BDate &date) const
- int operator>= (const BDate &date) const
- int operator< (const BDate &date) const
- int operator<= (const BDate &date) const

Static Public Member Functions

- static int isLeap (int year)
- static int daysInMonth (int year, int month)

Public Attributes

- uint16_t oyear

  Year (0 .. 65535)
- uint16_t oyday

  Day in year (0 .. 365)

7.14.1 Constructor & Destructor Documentation

7.14.1.1 BDate::BDate (int *year* = `0`, int *month* = `1`, int *day* = `1`)

7.14.1.2 BDate::BDate (BString *str*)

7.14.1.3 BDate::∼BDate ( )

7.14.2 Member Function Documentation

7.14.2.1 void BDate::clear ( )

Clear the date/time.

7.14.2.2 int BDate::compare (const BDate & *date*) const

Compare two dates.

7.14.2.3 int BDate::day ( )

7.14.2.4 int BDate::daysInMonth (int *year,* int *month*) `[static]`

7.14.2.5 void BDate::getDate (int & *year,* int & *mon,* int & *day*)

7.14.2.6 BString BDate::getString ( )

Get the time as an ISO date/time string.

7.14.2.7 BString BDate::getStringFormatted (BString *format*)

Gets the time in a string form as per the format. Format syntax as per strftime()

7.14.2.8 int BDate::isLeap (int *year*) [static]

7.14.2.9 int BDate::isSet ( ) [inline]

7.14.2.10 int BDate::month ( )

7.14.2.11 BDate::operator BString ( ) [inline]

7.14.2.12 int BDate::operator!= (const BDate & *date*) const [inline]

7.14.2.13 int BDate::operator< (const BDate & *date*) const [inline]

7.14.2.14 int BDate::operator<= (const BDate & *date*) const [inline]

7.14.2.15 int BDate::operator== (const BDate & *date*) const [inline]

7.14.2.16 int BDate::operator> (const BDate & *date*) const [inline]

7.14.2.17 int BDate::operator>= (const BDate & *date*) const [inline]

7.14.2.18 void BDate::set (time_t *time*)

Set time using Unix time (seconds from 1970-01-01)

7.14.2.19 void BDate::set (int *year* = 0, int *month* = 1, int *day* = 1)

7.14.2.20 void BDate::setFirst ( )

Set the first date available.

7.14.2.21 void BDate::setLast ( )

Set the last date available.

7.14.2.22 void BDate::setNow ( )

Set the timeStamp to now.

7.14.2.23 BError BDate::setString (BString *str*)

Set the time from an ISO date/time.

7.14.2.24 void BDate::setYDay (int *year* = 0, int *yday* = 0)

7.14.2.25 int BDate::yday ( )

7.14.2.26 int BDate::year ( )

7.14.3 Member Data Documentation

7.14.3.1 uint16_t BDate::oyday

Day in year (0 .. 365)

7.14.3.2 uint16_t BDate::oyear

Year (0 .. 65535)

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

- /src/cern/tms/beam/libBeam/BDate.h
- /src/cern/tms/beam/libBeam/BDate.cpp

7.15 BDebugBacktrace Class Reference

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

Public Member Functions

- BDebugBacktrace ()
- ∼BDebugBacktrace ()
- void dumpBacktraceStdout (char ∗comment)
- int dumpBacktraceFile (char ∗fileName, char ∗comment)
- void dumpBacktraceSyslog (char ∗comment)
- void dumpBacktrace (char ∗strBuf, int strBufLen, char ∗comment)

7.15.1 Constructor & Destructor Documentation

7.15.1.1 BDebugBacktrace::BDebugBacktrace ( )

7.15.1.2 BDebugBacktrace::∼BDebugBacktrace ( )

7.15.2 Member Function Documentation

7.15.2.1 void BDebugBacktrace::dumpBacktrace (char ∗ *strBuf,* int *strBufLen,* char ∗ *comment*)

7.15.2.2 int BDebugBacktrace::dumpBacktraceFile (char ∗ *fileName,* char ∗ *comment*)

7.15.2.3 void BDebugBacktrace::dumpBacktraceStdout (char ∗ *comment*)

7.15.2.4 void BDebugBacktrace::dumpBacktraceSyslog (char ∗ *comment*)

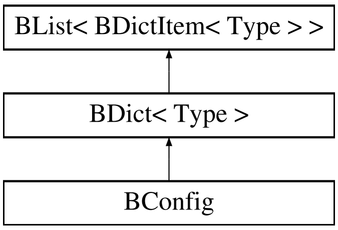
The documentation for this class was generated from the following files:

- /src/cern/tms/beam/libBeam/BDebug.h
- /src/cern/tms/beam/libBeam/BDebug.cpp

7.16 BDict< Type > Class Template Reference

```
#include <BDict.h>
```

Inheritance diagram for BDict< Type >:



Public Types

- typedef BIter iterator

Public Member Functions

- BDict (int hashSize=100)
- BDict (const BDict< Type > &dict)
- int hasKey (const BString &k) const
- BString key (const BIter &i) const
- void clear ()

  Clear the list.
- void insert (BIter &i, const BDictItem< Type > &item)

  Insert item before item.
- void append (const BDictItem< Type > &item)
- void append (const BDict< Type > &dict)
- void del (const BString &k)
- void del (BIter &i)

  Delete specified item.
- BIter find (const BString &k) const
- Type & operator[] (const BString &i)
- Type & operator[] (const BIter &i)
- const Type & operator[] (const BIter &i) const
- BDict< Type > operator+ (const BDict< Type > &dict) const
- BDict< Type > & operator= (const BDict< Type > &dict)
- void hashPrint ()

Private Member Functions

- void hashAdd (const BString &k, BIter iter)
- void hashDelete (const BString &k, BIter iter)
- int hashFind (const BString &k, BIter &iter) const

Private Attributes

- int ohashSize
- BArray< BList< BIter > > ohashLists

Additional Inherited Members

7.16.1 Member Typedef Documentation

7.16.1.1 template<class Type > typedef BIter BDict< Type >::iterator

7.16.2 Constructor & Destructor Documentation

7.16.2.1 template<class Type > BDict< Type >::BDict (int *hashSize* = `100`)

7.16.2.2 template<class Type > BDict< Type >::BDict (const BDict< Type > & *dict*)

7.16.3 Member Function Documentation

7.16.3.1 template<class Type > void BDict< Type >::append (const BDictItem< Type > & *item*)

7.16.3.2 template<class Type > void BDict< Type >::append (const BDict< Type > & *dict*)

7.16.3.3 template<class Type > void BDict< Type >::clear ( ) `[virtual]`

Clear the list.

Reimplemented from BList< BDictItem< Type > >.

7.16.3.4 template<class Type > void BDict< Type >::del (const BString & *k*)

7.16.3.5 template<class Type > void BDict< Type >::del (BIter & *i*) `[virtual]`

Delete specified item.

Reimplemented from BList< BDictItem< Type > >.

7.16.3.6 template<class Type > BIter BDict< Type >::find (const BString & *k*) const

7.16.3.7 template<class Type > void BDict< Type >::hashAdd (const BString & *k*, BIter *iter*) `[private]`

7.16.3.8 template<class Type > void BDict< Type >::hashDelete (const BString & *k*, BIter *iter*) `[private]`

7.16.3.9 template<class Type > int BDict< Type >::hashFind (const BString & *k*, BIter & *iter*) const `[private]`

7.16.3.10 template<class Type > void BDict< Type >::hashPrint ( )

7.16.3.11 template<class Type > int BDict< Type >::hasKey (const BString & *k*) const

7.16.3.12 template<class Type > void BDict< Type >::insert (BIter & *i*, const BDictItem< Type > & *item*) `[virtual]`

Insert item before item.

Reimplemented from BList< BDictItem< Type > >.

7.16.3.13 template<class Type > BString BDict< Type >::key (const BIter & *i*) const

7.16.3.14 template<class Type > BDict< Type > BDict< Type >::operator+ (const BDict< Type > & *dict*) const

7.16.3.15 template< class Type > BDict< Type > & BDict< Type >::operator= (const BDict< Type > & *dict*)

7.16.3.16 template< class Type > Type & BDict< Type >::operator[] (const BString & *i*)

7.16.3.17 template< class Type > Type & BDict< Type >::operator[] (const BIter & *i*)

7.16.3.18 template< class Type > const Type & BDict< Type >::operator[] (const BIter & *i*) const

7.16.4 Member Data Documentation

7.16.4.1 template< class Type > BArray<BList<BIter> > BDict< Type >::ohashLists [private]

7.16.4.2 template< class Type > int BDict< Type >::ohashSize [private]

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

- /src/cern/tms/beam/libBeam/BDict.h

7.17 BDictItem< Type > Class Template Reference

Template based Dictionary class.

`#include <BDict.h>`

Public Member Functions

- BDictItem (BString k="", Type v=Type())

Public Attributes

- BString key
- Type value

7.17.1 Detailed Description

template<class Type>class BDictItem< Type >

Template based Dictionary class.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 template< class Type > BDictItem< Type >::BDictItem (BString *k* = "", Type *v* = Type()) [inline]

7.17.3 Member Data Documentation

7.17.3.1 template< class Type > BString BDictItem< Type >::key

7.17.3.2 template< class Type > Type BDictItem< Type >::value

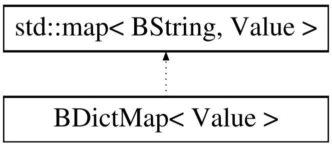
The documentation for this class was generated from the following file:

- /src/cern/tms/beam/libBeam/BDict.h

7.18 BDictMap< Value > Class Template Reference

```
#include <BDictMap.h>
```

Inheritance diagram for BDictMap< Value >:



Public Types

- typedef BDictMap< Value >::iterator iterator

Public Member Functions

- void clear ()
- int hasKey (const BString &k)
- BString key (iterator &i)
- unsigned int size ()
- void start (iterator &i)
- int isEnd (iterator &i)
- void next (iterator &i)
- void del (const iterator &i)
- void del (const BString &k)
- Value & operator[] (iterator &i)
- Value & operator[] (const BString &i)

7.18.1 Detailed Description

template<typename Value>class BDictMap< Value >

Template based Array class. This is based on the Standard C++ library map class and has all of the functionality of that class.

7.18.2 Member Typedef Documentation

7.18.2.1 template<typename Value > typedef BDictMap<Value>::iterator BDictMap< Value >::iterator

7.18.3 Member Function Documentation

7.18.3.1 template<typename Value > void BDictMap< Value >::clear ( ) [inline]

7.18.3.2 template<typename Value > void BDictMap< Value >::del (const iterator & *i*) [inline]

7.18.3.3 template<typename Value > void BDictMap< Value >::del (const BString & *k*) [inline]

7.18.3.4 template<typename Value > int BDictMap< Value >::hasKey (const BString & *k*) [inline]

7.18.3.5 template<typename Value > int BDictMap< Value >::isEnd (iterator & *i*) [inline]

7.18.3.6 template<typename Value > BString BDictMap< Value >::key (iterator & *i*) [inline]

7.18.3.7 template<typename Value > void BDictMap< Value >::next (iterator & *i*) [inline]

7.18.3.8 template<typename Value > Value& BDictMap< Value >::operator[] (iterator & *i*) [inline]

7.18.3.9 template<typename Value > Value& BDictMap< Value >::operator[] (const BString & *i*) [inline]

7.18.3.10 template<typename Value > unsigned int BDictMap< Value >::size ( ) [inline]

7.18.3.11 template<typename Value > void BDictMap< Value >::start (iterator & *i*) [inline]

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

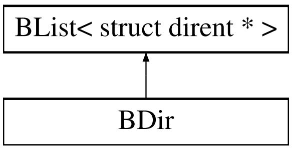
- /src/cern/tms/beam/libBeam/BDictMap.h

7.19 BDir Class Reference

File system directory class.

`#include <BDir.h>`

Inheritance diagram for BDir:



Public Member Functions

- BDir ()
- BDir (BString name)
- ∼BDir ()
- BError open (BString name)

  Reads named directory.
- BError error ()

  Current value of error.
- BError read ()

  read/re-reads directory
- void clear ()

  Clears list.
- void setWild (BString wild)

  Set wildcard filter string used on read.
- void setSort (int on)

  Set alpha sort on/off.
- BString entryName (BIter i)

  Get filename.
- struct stat entryStat (BIter i)

  Get file stats.
- struct stat64 entryStat64 (BIter i)

  Get file stats 64.

Private Attributes

- BError oerror
- BString odirname
- BString owild
- int osort

Additional Inherited Members

7.19.1 Detailed Description

File system directory class.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 BDir::BDir ( )

7.19.2.2 BDir::BDir (**BString** *name*)

7.19.2.3 BDir::∼BDir ( )

7.19.3 Member Function Documentation

7.19.3.1 void BDir::clear ( ) [virtual]

Clears list.

Reimplemented from BList< struct dirent ∗ >.

7.19.3.2 **BString** BDir::entryName (**BIter** *i*)

Get filename.

7.19.3.3 struct stat BDir::entryStat (**BIter** *i*)

Get file stats.

7.19.3.4 struct stat64 BDir::entryStat64 (**BIter** *i*)

Get file stats 64.

7.19.3.5 **BError** BDir::error ( )

Current value of error.

7.19.3.6 **BError** BDir::open (**BString** *name*)

Reads named directory.

7.19.3.7 **BError** BDir::read ( )

read/re-reads directory

7.19.3.8 void BDir::setSort (int *on*)

Set alpha sort on/off.

7.19.3.9 void BDir::setWild (BString *wild*)

Set wildcard filter string used on read.

7.19.4 Member Data Documentation

7.19.4.1 BString BDir::odirname `[private]`

7.19.4.2 BError BDir::oerror `[private]`

7.19.4.3 int BDir::osort `[private]`

7.19.4.4 BString BDir::owild `[private]`

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

- /src/cern/tms/beam/libBeam/BDir.h
- /src/cern/tms/beam/libBeam/BDir.cpp

7.20 BDuration Class Reference

`#include <BDuration.h>`

Public Member Functions

- BDuration (int hour=0, int minute=0, int second=0, int microsecond=0)
- BDuration (BString str)
- ∼BDuration ()
- void clear ()

  Clear the duration.
- void set (int hour=0, int minute=0, int second=0, int microsecond=0)
- void addMilliSeconds (int64_t milliSeconds)

  Add the given number of milli seconds.
- void addMicroSeconds (int64_t microSeconds)

  Add the given number of micro seconds.
- void addSeconds (int seconds)

  Add the given number of seconds.
- uint32_t getSeconds ()

  Get number of seconds.
- uint64_t getMicroSeconds ()

  Get number of micro seconds.
- int hour ()
- int minute ()
- int second ()
- int microSecond ()
- BString getString ()

  Get the time as an ISO date/time string.
- BError setString (BString time)

  Set the time from an ISO date/time.

Private Attributes

- uint8_t ohour

    Hour (0 .. 23)
- uint8_t ominute

    Minute (0 .. 59)
- uint8_t osecond

    Second (0 .. 59)
- uint8_t ospare

    Padding.
- uint32_t omicroSecond

    MicroSecond (0 .. 999999)

7.20.1 Constructor & Destructor Documentation

7.20.1.1 BDuration::BDuration (int *hour* = 0, int *minute* = 0, int *second* = 0, int *microsecond* = 0)

7.20.1.2 BDuration::BDuration (BString *str*)

7.20.1.3 BDuration::∼BDuration ( )

7.20.2 Member Function Documentation

7.20.2.1 void BDuration::addMicroSeconds (int64_t *microSeconds*)

Add the given number of micro seconds.

7.20.2.2 void BDuration::addMilliSeconds (int64_t *milliSeconds*)

Add the given number of milli seconds.

7.20.2.3 void BDuration::addSeconds (int *seconds*)

Add the given number of seconds.

7.20.2.4 void BDuration::clear ( )

Clear the duration.

7.20.2.5 uint64_t BDuration::getMicroSeconds ( )

Get number of micro seconds.

7.20.2.6 uint32_t BDuration::getSeconds ( )

Get number of seconds.

7.20.2.7 BString BDuration::getString ( )

Get the time as an ISO date/time string.

7.20.2.8 int BDuration::hour ( )

7.20.2.9 int BDuration::microSecond ( )

7.20.2.10 int BDuration::minute ( )

7.20.2.11 int BDuration::second ( )

7.20.2.12 void BDuration::set (int *hour* = `0`, int *minute* = `0`, int *second* = `0`, int *microsecond* = `0`)

7.20.2.13 BError BDuration::setString (BString *time*)

Set the time from an ISO date/time.

7.20.3 Member Data Documentation

7.20.3.1 uint8_t BDuration::ohour `[private]`

Hour (0 .. 23)

7.20.3.2 uint32_t BDuration::omicroSecond `[private]`

MicroSecond (0 .. 999999)

7.20.3.3 uint8_t BDuration::ominute `[private]`

Minute (0 .. 59)

7.20.3.4 uint8_t BDuration::osecond `[private]`

Second (0 .. 59)

7.20.3.5 uint8_t BDuration::ospare `[private]`

Padding.

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

- /src/cern/tms/beam/libBeam/BDuration.h
- /src/cern/tms/beam/libBeam/BDuration.cpp

7.21 BEntry Class Reference

Manipulate a name value pair.

`#include <BEntry.h>`

Public Member Functions

- BEntry ()
- BEntry (BString name, BString value)

  Set name and value.

- BEntry (BString line)

  Set name and value from white space deliminated string.
- BString getName ()

  Get the name.
- BString getValue ()

  Get the value.
- void setLine (BString line)

  Set name and value from white space deliminated string.
- void setName (BString name)

  Set the name.
- void setValue (BString value)

  Set the value.
- BString line ()

  Return name and value as padded single string.
- void print ()

  Print name and value.

Private Attributes

- BString oname
- BString ovalue

7.21.1 Detailed Description

Manipulate a name value pair.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 BEntry::BEntry ( )

7.21.2.2 BEntry::BEntry (BString *name*, BString *value*)

Set name and value.

7.21.2.3 BEntry::BEntry (BString *line*)

Set name and value from white space deliminated string.

7.21.3 Member Function Documentation

7.21.3.1 BString BEntry::getName ( )

Get the name.

7.21.3.2 BString BEntry::getValue ( )

Get the value.

7.21.3.3 BString BEntry::line ( )

Return name and value as padded single string.

7.21.3.4 void BEntry::print ( )

Print name and value.

7.21.3.5 void BEntry::setLine (BString *line*)

Set name and value from white space deliminated string.

7.21.3.6 void BEntry::setName (BString *name*)

Set the name.

7.21.3.7 void BEntry::setValue (BString *value*)

Set the value.

7.21.4 Member Data Documentation

7.21.4.1 BString BEntry::oname `[private]`

7.21.4.2 BString BEntry::ovalue `[private]`

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

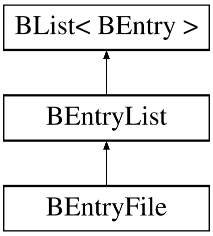
- /src/cern/tms/beam/libBeam/BEntry.h
- /src/cern/tms/beam/libBeam/BEntry.cpp

7.22 BEntryFile Class Reference

File of Entries.

`#include <BEntry.h>`

Inheritance diagram for BEntryFile:



Public Member Functions

- BEntryFile ()
- BEntryFile (BString filename)

Opens entryfile.

- ∼BEntryFile ()
- int open (BString filename)

  Opens entryfile.
- int read ()

  Reads entry file and builds list.
- int write ()

  Writes list to entryfile.
- int writeList (BEntryList &l)

  Writes specified list to file.
- void clear ()

  Clears current list.
- BString filename ()

  Returns the filename.

Private Attributes

- BString ofilename
- BString ocomments

Additional Inherited Members

7.22.1 Detailed Description

File of Entries.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 BEntryFile::BEntryFile ( )

7.22.2.2 BEntryFile::BEntryFile (BString *filename*)

Opens entryfile.

7.22.2.3 BEntryFile::∼BEntryFile ( )

7.22.3 Member Function Documentation

7.22.3.1 void BEntryFile::clear ( ) [virtual]

Clears current list.

Reimplemented from BEntryList.

7.22.3.2 BString BEntryFile::filename ( )

Returns the filename.

7.22.3.3 int BEntryFile::open (BString *filename*)

Opens entryfile.

7.22.3.4 int BEntryFile::read ( )

Reads entry file and builds list.

7.22.3.5 int BEntryFile::write ( )

Writes list to entryfile.

7.22.3.6 int BEntryFile::writeList (BEntryList & *l*)

Writes specified list to file.

7.22.4 Member Data Documentation

7.22.4.1 BString BEntryFile::ocomments [private]

7.22.4.2 BString BEntryFile::ofilename [private]

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

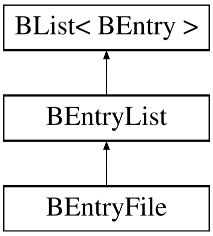
- /src/cern/tms/beam/libBeam/BEntry.h
- /src/cern/tms/beam/libBeam/BEntry.cpp

7.23 BEntryList Class Reference

List of Entries. Where an entry is a name value pair.

`#include <BEntry.h>`

Inheritance diagram for BEntryList:



Public Member Functions

- BEntryList ()
- int isSet (BString name)

  1 if name is in list and value is set
- BEntry * find (BString name)

  Returns entry if name is found otherwise NULL.
- BString findValue (BString name)

  Returns value of name. Returns "" if name not found.
- int setValue (BString name, BString value)

  Set the value of name. Returns 0 if name not found.
- int setValueRaw (BString name, BString value)

Raw setting of value without looking up existing entry.

- void deleteEntry (BString name)

  Deletes the entry.
- void print ()

  Print list.
- BString getString ()

  Return list as string. Each Entry padded and on a new line.
- void insert (BIter &i, const BEntry &item)

  Insert item before item.
- void del (BIter &i)

  Delete specified item.
- void clear ()

  Clear the list.
- BEntryList & operator= (const BEntryList &l)

Private Attributes

- BIter olastPos

Additional Inherited Members

7.23.1 Detailed Description

List of Entries. Where an entry is a name value pair.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 BEntryList::BEntryList ( )

7.23.3 Member Function Documentation

7.23.3.1 void BEntryList::clear ( ) [virtual]

Clear the list.

Reimplemented from BList< BEntry >.

Reimplemented in BEntryFile.

7.23.3.2 void BEntryList::del (BIter & *i*) [virtual]

Delete specified item.

Reimplemented from BList< BEntry >.

7.23.3.3 void BEntryList::deleteEntry (BString *name*)

Deletes the entry.

7.23.3.4 BEntry ∗ BEntryList::find (BString *name*)

Returns entry if name is found otherwise NULL.

7.23.3.5 BString BEntryList::findValue (BString *name*)

Returns value of name. Returns "" if name not found.

7.23.3.6 BString BEntryList::getString ( )

Return list as string. Each Entry padded and on a new line.

7.23.3.7 void BEntryList::insert (BIter & *i*, const BEntry & *item*) [virtual]

Insert item before item.

Reimplemented from BList< BEntry >.

7.23.3.8 int BEntryList::isSet (BString *name*)

1 if name is in list and value is set

7.23.3.9 BEntryList & BEntryList::operator= (const BEntryList & *l*)

7.23.3.10 void BEntryList::print ( )

Print list.

7.23.3.11 int BEntryList::setValue (BString *name*, BString *value*)

Set the value of name. Returns 0 if name not found.

7.23.3.12 int BEntryList::setValueRaw (BString *name*, BString *value*)

Raw setting of value without looking up existing entry.

7.23.4 Member Data Documentation

7.23.4.1 BIter BEntryList::olastPos [private]

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

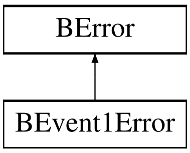
- /src/cern/tms/beam/libBeam/BEntry.h
- /src/cern/tms/beam/libBeam/BEntry.cpp

7.24 BError Class Reference

Error return class.

`#include <BError.h>`

Inheritance diagram for BError:



Public Member Functions

- BError (int errNo=ErrorOk, BString errStr="")

  Create object.
- BError (BString errStr)

  Create with error set and error string.
- BError copy ()

  Return an independant copy.
- BError & set (int errNo, BString errStr="")

  Set error number and message.
- BError & clear ()

  Clear the error.
- BError & setError (BString errStr="")

  Set error type ERROR with optional message.
- BString getString () const

  Get error message.
- int getNumber () const

  Get The error number.
- int num () const

  Get The error number.
- const char ∗ str () const

  Return a char∗ string.
- int getErrorNo () const

  Get The error number.
- operator int () const

  Return error number.

Private Attributes

- int oerrNo
- BString oerrStr

7.24.1 Detailed Description

Error return class.

7.24.2 Constructor & Destructor Documentation

7.24.2.1 BError::BError (int *errNo* = ErrorOk, BString *errStr* = " ")

Create object.

7.24.2.2 BError::BError (BString *errStr*)

Create with error set and error string.

7.24.3 Member Function Documentation

7.24.3.1 BError & BError::clear ( )

Clear the error.

7.24.3.2 BError BError::copy ( )

Return an independant copy.

7.24.3.3 int BError::getErrorNo ( ) const

Get The error number.

7.24.3.4 int BError::getNumber ( ) const

Get The error number.

7.24.3.5 BString BError::getString ( ) const

Get error message.

7.24.3.6 int BError::num ( ) const

Get The error number.

7.24.3.7 BError::operator int ( ) const [inline]

Return error number.

7.24.3.8 BError & BError::set (int *errNo*, BString *errStr* = "")

Set error number and message.

7.24.3.9 BError & BError::setError (BString *errStr* = "")

Set error type ERROR with optional message.

7.24.3.10 const char * BError::str ( ) const

Return a char* string.

7.24.4 Member Data Documentation

7.24.4.1 int BError::oerrNo [private]

7.24.4.2 BString BError::oerrStr [private]

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

- /src/cern/tms/beam/libBeam/BError.h
- /src/cern/tms/beam/libBeam/BError.cpp

7.25 BErrorTime Class Reference

Error return class.

```
#include <BErrorTime.h>
```

Public Types

- enum Type { None = 0, Error = 1 }

Public Member Functions

- BErrorTime (int errNo=None, BTimeStamp errTime=BTimeStamp(), BString errStr="")

  Create object.
- BErrorTime & set (int errNo, BTimeStamp errTime=BTimeStamp(), BString errStr="")

  Set error number and message.
- BErrorTime & clear ()

  Clear the error.
- int getErrorNo () const

  Get The error number.
- BTimeStamp getTime () const

  Get time.
- BString getString () const

  Get error message.
- BErrorTime copy ()

  Return an independant copy.
- operator int () const

  Return error number.

Private Attributes

- int oerrNo
- BTimeStamp oerrTime
- BString oerrStr

7.25.1 Detailed Description

Error return class.

7.25.2 Member Enumeration Documentation

7.25.2.1 enum **BErrorTime::Type**

Enumerator

None

Error

7.25.3 Constructor & Destructor Documentation

7.25.3.1 BErrorTime::BErrorTime (int *errNo* = None, BTimeStamp *errTime* = BTimeStamp(), BString *errStr* = "")

Create object.

7.25.4 Member Function Documentation

7.25.4.1 BErrorTime & BErrorTime::clear ( )

Clear the error.

7.25.4.2 BErrorTime BErrorTime::copy ( )

Return an independant copy.

7.25.4.3 int BErrorTime::getErrorNo ( ) const

Get The error number.

7.25.4.4 BString BErrorTime::getString ( ) const

Get error message.

7.25.4.5 BTimeStamp BErrorTime::getTime ( ) const

Get time.

7.25.4.6 BErrorTime::operator int ( ) const

Return error number.

7.25.4.7 BErrorTime & BErrorTime::set (int *errNo*, BTimeStamp *errTime* = BTimeStamp(), BString *errStr* = "")

Set error number and message.

7.25.5 Member Data Documentation

7.25.5.1 int BErrorTime::oerrNo `[private]`

7.25.5.2 BString BErrorTime::oerrStr `[private]`

7.25.5.3 BTimeStamp BErrorTime::oerrTime [private]

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

- /src/cern/tms/beam/libBeam/BErrorTime.h
- /src/cern/tms/beam/libBeam/BErrorTime.cpp

7.26 BEvent Class Reference

```
#include <BEvent.h>
```

Public Member Functions

- BEvent (BUInt32 type=BEventTypeNone, BUInt32 arg=0)
- BUInt32 type ()
- BUInt32 arg ()

Private Attributes

- BUInt32 otype

  The events type.
- BUInt32 oarg

  The events argument.

7.26.1 Constructor & Destructor Documentation

7.26.1.1 BEvent::BEvent (BUInt32 *type* = BEventTypeNone, BUInt32 *arg* = 0)

7.26.2 Member Function Documentation

7.26.2.1 BUInt32 BEvent::arg ( )

7.26.2.2 BUInt32 BEvent::type ( )

7.26.3 Member Data Documentation

7.26.3.1 BUInt32 BEvent::oarg [private]

The events argument.

7.26.3.2 BUInt32 BEvent::otype [private]

The events type.

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

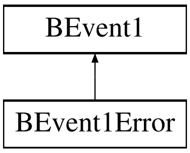
- /src/cern/tms/beam/libBeam/BEvent.h
- /src/cern/tms/beam/libBeam/BEvent.cpp

7.27 BEvent1 Class Reference

This class provides a base class for all event objects that can be sent over the events interface.

`#include <BEvent1.h>`

Inheritance diagram for BEvent1:



Public Member Functions

- BEvent1 (uint32_t type)
- virtual ∼BEvent1 ()
- uint32_t getType ()
- virtual BError getBinary (void ∗data, uint32_t &size)
- virtual BError setBinary (void ∗data, uint32_t &size)

Private Attributes

- uint32_t otype

  The event type.

7.27.1 Detailed Description

This class provides a base class for all event objects that can be sent over the events interface.

7.27.2 Constructor & Destructor Documentation

7.27.2.1 BEvent1::BEvent1 (uint32_t *type*)

7.27.2.2 BEvent1::∼BEvent1 ( ) `[virtual]`

7.27.3 Member Function Documentation

7.27.3.1 BError BEvent1::getBinary (void ∗ *data,* uint32_t & *size*) `[virtual]`

Reimplemented in BEvent1Error.

7.27.3.2 uint32_t BEvent1::getType ( )

7.27.3.3 BError BEvent1::setBinary (void ∗ *data,* uint32_t & *size*) `[virtual]`

Reimplemented in BEvent1Error.

7.27.4 Member Data Documentation

7.27.4.1 uint32_t BEvent1::otype [private]

The event type.

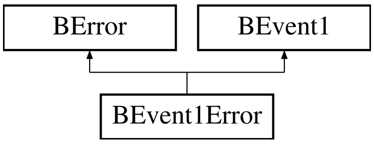
The documentation for this class was generated from the following files:

- /src/cern/tms/beam/libBeam/BEvent1.h
- /src/cern/tms/beam/libBeam/BEvent1.cpp

7.28 BEvent1Error Class Reference

```
#include <BEvent1.h>
```

Inheritance diagram for BEvent1Error:



Public Member Functions

- BEvent1Error (int errNo=ErrorOk, BString errStr="")
- BError getBinary (void ∗data, uint32_t &size)
- BError setBinary (void ∗data, uint32_t &size)

7.28.1 Constructor & Destructor Documentation

7.28.1.1 BEvent1Error::BEvent1Error (int *errNo* = ErrorOk, BString *errStr* = "")

7.28.2 Member Function Documentation

7.28.2.1 BError BEvent1Error::getBinary (void ∗ *data*, uint32_t & *size*) [virtual]

Reimplemented from BEvent1.

7.28.2.2 BError BEvent1Error::setBinary (void ∗ *data*, uint32_t & *size*) [virtual]

Reimplemented from BEvent1.

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

- /src/cern/tms/beam/libBeam/BEvent1.h
- /src/cern/tms/beam/libBeam/BEvent1.cpp

7.29 BEvent1Int Class Reference

This class provides an interface for sending simple integer events via a file descriptor. This allows threads to send events that can be picked up by the poll system call.

```
#include <BEvent1.h>
```

Public Member Functions

- BEvent1Int ()
- ∼BEvent1Int ()
- void clear ()

  Clear events pending.
- BError sendEvent (int event)

  Send an event.
- BError getEvent (int &event, int timeOutUs=-1)

  Receive the event.
- int getFd ()

Private Attributes

- int ofds [2]

  File descriptors for pipe.

7.29.1 Detailed Description

This class provides an interface for sending simple integer events via a file descriptor. This allows threads to send events that can be picked up by the poll system call.

7.29.2 Constructor & Destructor Documentation

7.29.2.1 BEvent1Int::BEvent1Int ( )

7.29.2.2 BEvent1Int::∼BEvent1Int ( )

7.29.3 Member Function Documentation

7.29.3.1 void BEvent1Int::clear ( )

Clear events pending.

7.29.3.2 BError BEvent1Int::getEvent (int & *event,* int *timeOutUs* = `-1`)

Receive the event.

7.29.3.3 int BEvent1Int::getFd ( )

7.29.3.4 BError BEvent1Int::sendEvent (int *event*)

Send an event.

7.29.4 Member Data Documentation

7.29.4.1 int BEvent1Int::ofds[2] `[private]`

File descriptors for pipe.

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

- /src/cern/tms/beam/libBeam/BEvent1.h
- /src/cern/tms/beam/libBeam/BEvent1.cpp

7.30 BEvent1Pipe Class Reference

This class provides a base interface for sending events via a pipe. This allows threads to send events that can be picked up by the poll system call.

```
#include <BEvent1.h>
```

Public Member Functions

- BEvent1Pipe ()
- ∼BEvent1Pipe ()
- void clear ()

  Clear events pending.
- BError sendEvent (BEvent1 ∗event)

  Send an event.
- BError getEvent (BEvent1 ∗event, int timeOutUs=-1)

  Receive the event.
- int getReceiveFd ()

  returns the receive file descriptor for the poll system call

Private Attributes

- int ofds [2]

  File descriptors for pipe.

7.30.1 Detailed Description

This class provides a base interface for sending events via a pipe. This allows threads to send events that can be picked up by the poll system call.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 BEvent1Pipe::BEvent1Pipe ( )

7.30.2.2 BEvent1Pipe::∼BEvent1Pipe ( )

7.30.3 Member Function Documentation

7.30.3.1 void BEvent1Pipe::clear ( )

Clear events pending.

7.30.3.2 BError BEvent1Pipe::getEvent (BEvent1 ∗ *event,* int *timeOutUs* = `-1`)

Receive the event.

7.30.3.3 int BEvent1Pipe::getReceiveFd ( )

returns the receive file descriptor for the poll system call

7.30.3.4 BError BEvent1Pipe::sendEvent (BEvent1 * *event*)

Send an event.

7.30.4 Member Data Documentation

7.30.4.1 int BEvent1Pipe::ofds[2] [private]

File descriptors for pipe.

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

- /src/cern/tms/beam/libBeam/BEvent1.h
- /src/cern/tms/beam/libBeam/BEvent1.cpp

7.31 BEventPipe Class Reference

This class provides an interface for sending simple integer events via a pipe file descriptor.

`#include <BEvent.h>`

Public Member Functions

- BEventPipe ()
- ∼BEventPipe ()
- void clear ()

  Clear events pending.
- int getFd ()
- BUInt writeAvailable () const
- BError write (const BEvent &event, BTimeout timeout=BTimeoutForever)

  Append an item onto the queue.
- BUInt readAvailable () const
- BError read (BEvent &event, BTimeout timeout=BTimeoutForever)

  Get an item from the queue.

Private Attributes

- int ofds [2]

  File descriptors for pipe.

7.31.1 Detailed Description

This class provides an interface for sending simple integer events via a pipe file descriptor.

7.31.2 Constructor & Destructor Documentation

7.31.2.1 BEventPipe::BEventPipe ( )

7.31.2.2 BEventPipe::∼BEventPipe ( )

7.31.3 Member Function Documentation

7.31.3.1 void BEventPipe::clear ( )

Clear events pending.

7.31.3.2 int BEventPipe::getFd ( )

7.31.3.3 BError BEventPipe::read (BEvent & *event*, BTimeout *timeout* = BTimeoutForever)

Get an item from the queue.

7.31.3.4 BUInt BEventPipe::readAvailable ( ) const

7.31.3.5 BError BEventPipe::write (const BEvent & *event*, BTimeout *timeout* = BTimeoutForever)

Append an item onto the queue.

7.31.3.6 BUInt BEventPipe::writeAvailable ( ) const

7.31.4 Member Data Documentation

7.31.4.1 int BEventPipe::ofds[2] `[private]`

File descriptors for pipe.

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

- /src/cern/tms/beam/libBeam/BEvent.h
- /src/cern/tms/beam/libBeam/BEvent.cpp

7.32 BFifo< Type > Class Template Reference

```
#include <BFifo.h>
```

Public Member Functions

- BFifo (BUInt size)
- ∼BFifo ()
- void clear ()
- BUInt size ()

  Returns fifo size.
- BError resize (BUInt size)

  Resize FIFO, clears it as well.
- BUInt writeAvailable ()

  How many items that can be written.

- BUInt writeAvailableChunk ()

  How many items that can be written in a chunk.
- BError write (const Type v)

  Write a single item.
- BError write (const Type ∗data, BUInt num)

  Write a set of items. Can only write a maximum of writeAvailableChunk() to save going beyond end of FIFO buffer.
- Type ∗ writeData ()

  Returns a pointer to the data.
- Type ∗ writeData (BUInt &num)

  Returns a pointer to the data and how many can be written in a chunk.
- void writeDone (BUInt num)

  Indicates when write is complete.
- void writeBackup (BUInt num)

  Backup, remove num items at end of fifo. Careful, make sure read is not already happening.
- BUInt readAvailable ()

  How many items are available to read.
- BUInt readAvailableChunk ()

  How many items are available to read in a chunk.
- Type read ()

  Read one item.
- BError read (Type ∗data, BUInt num)

  Read a set of items.
- Type readPos (BUInt pos)

  Read item at given offset from current read position.
- Type ∗ readData ()

  Returns a pointer to the data.
- Type ∗ readData (BUInt &num)

  Returns a pointer to the data and how many can be read in a chunk.
- void readDone (BUInt num)
- Type & operator[] (int pos)

  Direct access to read samples in buffer.

Protected Attributes

- BMutex olock
- BUInt osize

  The size of the FIFO.
- Type ∗ odata

  FIFO memory buffer.
- BUInt owritePos

  The write pointer.
- BUInt oreadPos

  The read pointer.

7.32.1 Constructor & Destructor Documentation

7.32.1.1 template<class Type> BFifo< Type >::BFifo (BUInt *size*)

7.32.1.2 template<class Type> BFifo< Type >::∼BFifo ( )

7.32.2 Member Function Documentation

7.32.2.1 template<class Type> void BFifo< Type >::clear ( )

7.32.2.2 template<class Type> Type& BFifo< Type >::operator[] (int *pos*)

Direct access to read samples in buffer.

7.32.2.3 template<class Type> Type BFifo< Type >::read ( )

Read one item.

7.32.2.4 template<class Type> BError BFifo< Type >::read (Type ∗ *data,* BUInt *num*)

Read a set of items.

7.32.2.5 template<class Type> BUInt BFifo< Type >::readAvailable ( )

How many items are available to read.

7.32.2.6 template<class Type> BUInt BFifo< Type >::readAvailableChunk ( )

How many items are available to read in a chunk.

7.32.2.7 template<class Type> Type∗ BFifo< Type >::readData ( )

Returns a pointer to the data.

7.32.2.8 template<class Type> Type∗ BFifo< Type >::readData (BUInt & *num*)

Returns a pointer to the data and how many can be read in a chunk.

7.32.2.9 template<class Type> void BFifo< Type >::readDone (BUInt *num*)

7.32.2.10 template<class Type> Type BFifo< Type >::readPos (BUInt *pos*)

Read item at given offset from current read position.

7.32.2.11 template<class Type> BError BFifo< Type >::resize (BUInt *size*)

Resize FIFO, clears it as well.

7.32.2.12 template<class Type> BUInt BFifo< Type >::size ( )

Returns fifo size.

7.32.2.13 template<class Type> BError BFifo< Type >::write (const Type *v*)

Write a single item.

7.32.2.14 template<class Type> BError BFifo< Type >::write (const Type ∗ *data,* BUInt *num*)

Write a set of items. Can only write a maximum of writeAvailableChunk() to save going beyond end of FIFO buffer.

7.32.2.15 template<class Type> BUInt BFifo< Type >::writeAvailable ( )

How many items that can be written.

7.32.2.16 template<class Type> BUInt BFifo< Type >::writeAvailableChunk ( )

How many items that can be written in a chunk.

7.32.2.17 template<class Type> void BFifo< Type >::writeBackup (BUInt *num*)

Backup, remove num items at end of fifo. Careful, make sure read is not already happening.

7.32.2.18 template<class Type> Type∗ BFifo< Type >::writeData ( )

Returns a pointer to the data.

7.32.2.19 template<class Type> Type∗ BFifo< Type >::writeData (BUInt & *num*)

Returns a pointer to the data and how many can be written in a chunk.

7.32.2.20 template<class Type> void BFifo< Type >::writeDone (BUInt *num*)

Indicates when write is complete.

7.32.3 Member Data Documentation

7.32.3.1 template<class Type> Type∗ BFifo< Type >::odata [protected]

FIFO memory buffer.

7.32.3.2 template<class Type> BMutex BFifo< Type >::olock [protected]

7.32.3.3 template<class Type> BUInt BFifo< Type >::oreadPos [protected]

The read pointer.

7.32.3.4 template<class Type> BUInt BFifo< Type >::osize [protected]

The size of the FIFO.

7.32.3.5 template<class Type> BUInt BFifo< Type >::owritePos [protected]

The write pointer.

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

- /src/cern/tms/beam/libBeam/BFifo.h

7.33 BFifoCirc< Type > Class Template Reference

This class implements a thread safe FIFO buffer.

```
#include <BFifoCirc.h>
```

Public Types

- enum { defaultSize = 1024 }

Public Member Functions

- BFifoCirc (uint32_t size=defaultSize)
- ∼BFifoCirc ()
- uint32_t size ()

  Return the buffers actual size.
- void clear ()

  Clear all of the data in the buffer.
- uint32_t writeAvailable ()

  Returns the space available to write.
- BError writeWaitAvailable (uint32_t numFifoSamples)

  Wait for the given number of samples.
- BError write (const Type ∗data, uint32_t numFifoSamples)

  Writes the data to the buffer. Blocks until complete.
- Type ∗ writeData ()

  Return a pointer to the current start of the buffer.
- void writeDone (uint32_t numFifoSamples)

  Update the write pointer.
- uint32_t readAvailable ()

  Returns the number of bytes of data available.
- BError readWaitAvailable (uint32_t numFifoSamples)

  Wait for given number of samples.
- BError read (Type ∗data, uint32_t numFifoSamples)
- Type ∗ readData ()

  Pointer to raw data.
- BError readDone (uint32_t numFifoSamples)

  Updates read pointer.
- Type & operator[] (int pos)

  Direct access to read samples in buffer.

Protected Member Functions

- BError mapCircularBuffer (uint32_t size)
- void unmapCircularBuffer ()

Protected Attributes

- BMutex olock
- uint32_t ovmSize
- uint32_t osize
- Type ∗ odata
- BFifoCircPos owritePos

  Current write position.
- BCondValue owriteNumFifoSamples

  The number of samples in the FIFO.
- BFifoCircPos oreadPos

  Current read position.

7.33.1 Detailed Description

template<class Type>class BFifoCirc< Type >

This class implements a thread safe FIFO buffer.

7.33.2 Member Enumeration Documentation

7.33.2.1 template<class Type > anonymous enum

Enumerator

defaultSize

7.33.3 Constructor & Destructor Documentation

7.33.3.1 template<class Type > BFifoCirc< Type >::BFifoCirc (uint32_t *size* = defaultSize)

7.33.3.2 template<class Type > BFifoCirc< Type >::∼BFifoCirc ( )

7.33.4 Member Function Documentation

7.33.4.1 template<class Type > void BFifoCirc< Type >::clear ( )

Clear all of the data in the buffer.

7.33.4.2 template<class Type > BError BFifoCirc< Type >::mapCircularBuffer (uint32_t *size*) [protected]

7.33.4.3 template<class Type > Type& BFifoCirc< Type >::operator[] (int *pos*)

Direct access to read samples in buffer.

7.33.4.4 template<class Type > BError BFifoCirc< Type >::read (Type ∗ *data*, uint32_t *numFifoSamples*)

7.33.4.5 template<class Type > uint32_t BFifoCirc< Type >::readAvailable ( )

Returns the number of bytes of data available.

7.33.4.6 template<class Type > Type∗ BFifoCirc< Type >::readData ( )

Pointer to raw data.

7.33.4.7 template<class Type > BError BFifoCirc< Type >::readDone (uint32_t *numFifoSamples*)

Updates read pointer.

7.33.4.8 template<class Type > BError BFifoCirc< Type >::readWaitAvailable (uint32_t *numFifoSamples*)

Wait for given number of samples.

7.33.4.9 template<class Type > uint32_t BFifoCirc< Type >::size ( )

Return the buffers actual size.

7.33.4.10 template<class Type > void BFifoCirc< Type >::unmapCircularBuffer ( ) `[protected]`

7.33.4.11 template<class Type > BError BFifoCirc< Type >::write (const Type ∗ *data,* uint32_t *numFifoSamples*)

Writes the data to the buffer. Blocks until complete.

7.33.4.12 template<class Type > uint32_t BFifoCirc< Type >::writeAvailable ( )

Returns the space available to write.

7.33.4.13 template<class Type > Type∗ BFifoCirc< Type >::writeData ( )

Return a pointer to the current start of the buffer.

7.33.4.14 template<class Type > void BFifoCirc< Type >::writeDone (uint32_t *numFifoSamples*)

Update the write pointer.

7.33.4.15 template<class Type > BError BFifoCirc< Type >::writeWaitAvailable (uint32_t *numFifoSamples*)

Wait for the given number of samples.

7.33.5 Member Data Documentation

7.33.5.1 template<class Type > Type∗ BFifoCirc< Type >::odata `[protected]`

7.33.5.2 template<class Type > BMutex BFifoCirc< Type >::olock `[protected]`

7.33.5.3 template<class Type > BFifoCircPos BFifoCirc< Type >::oreadPos `[protected]`

Current read position.

7.33.5.4 template<class Type > uint32_t BFifoCirc< Type >::osize [protected]

7.33.5.5 template<class Type > uint32_t BFifoCirc< Type >::ovmSize [protected]

7.33.5.6 template<class Type > BCondValue BFifoCirc< Type >::owriteNumFifoSamples [protected]

The number of samples in the FIFO.

7.33.5.7 template<class Type > BFifoCircPos BFifoCirc< Type >::owritePos [protected]

Current write position.

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

- /src/cern/tms/beam/libBeam/BFifoCirc.h

7.34 BFifoCircPos Class Reference

This class implements a pointer into the Fifo's circular buffer.

`#include <BFifoCirc.h>`

Public Member Functions

- BFifoCircPos (uint32_t size)
- void setSize (uint32_t size)
- void set (uint32_t pos)

    Sets the position.
- uint32_t pos ()

    The current position.
- void increment (uint32_t numFifoSamples)

    Increment the pointer by the given value.
- uint32_t difference (const BFifoCircPos &pos)

    Return the difference between the two pointers.
- operator int ()
- void operator+= (uint32_t numFifoSamples)
- int operator== (const BFifoCircPos &pos)
- int operator!= (const BFifoCircPos &pos)

Private Attributes

- uint32_t osize
- uint32_t opos

7.34.1 Detailed Description

This class implements a pointer into the Fifo's circular buffer.

7.34.2 Constructor & Destructor Documentation

7.34.2.1 BFifoCircPos::BFifoCircPos (uint32_t *size*)

7.34.3 Member Function Documentation

7.34.3.1 uint32_t BFifoCircPos::difference (const BFifoCircPos & *pos*)

Return the difference between the two pointers.

7.34.3.2 void BFifoCircPos::increment (uint32_t *numFifoSamples*)

Increment the pointer by the given value.

7.34.3.3 BFifoCircPos::operator int ( )

7.34.3.4 int BFifoCircPos::operator!= (const BFifoCircPos & *pos*)

7.34.3.5 void BFifoCircPos::operator+= (uint32_t *numFifoSamples*)

7.34.3.6 int BFifoCircPos::operator== (const BFifoCircPos & *pos*)

7.34.3.7 uint32_t BFifoCircPos::pos ( )

The current position.

7.34.3.8 void BFifoCircPos::set (uint32_t *pos*)

Sets the position.

7.34.3.9 void BFifoCircPos::setSize (uint32_t *size*)

7.34.4 Member Data Documentation

7.34.4.1 uint32_t BFifoCircPos::opos `[private]`

7.34.4.2 uint32_t BFifoCircPos::osize `[private]`

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

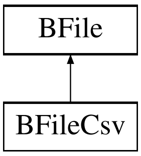
- /src/cern/tms/beam/libBeam/BFifoCirc.h
- /src/cern/tms/beam/libBeam/BFifoCirc.cpp

7.35 BFile Class Reference

File operations class.

`#include <BFile.h>`

Inheritance diagram for BFile:



Public Member Functions

- BFile ()
- BFile (const BFile &file)

    Create opened specified file.
- ∼BFile ()
- BError open (BString name, BString mode)

    Open file.
- BError open (FILE ∗file)

    Assign object to opened file handle.
- BError open (int fd, BString mode)

    Assign object to opened file descriptor.
- BError close ()

    Close file.
- int isOpen ()

    Returns 1 if the file is open.
- int isEnd ()

    Returns 1 if at the end of the file, 0 otherwise.
- FILE ∗ getFd ()

    File descriptor.
- BUInt64 length ()

    File size in bytes.
- int setVBuf (char ∗buf, int mode, size_t size)

    Set stream buffering options.
- int read (void ∗buf, int nbytes)

    Read from file.
- int readString (BString &str)

    Read string. (ref fgets)
- char ∗ fgets (char ∗buf, size_t size)
- int write (const void ∗buf, int nbytes)

    Write to file.
- int writeString (const BString &str)

    Write string to file.
- int seek (BUInt64 pos)

    Set seek position.
- BUInt64 position ()

    The files position.
- int printf (const char ∗fmt,...)

    Formated print into the file.
- BError truncate ()

    Truncate the file.
- BError flush ()

    Flush the file.
- BString fileName ()

    Return file name.
- BFile & operator= (const BFile &file)

Private Attributes

- FILE ∗ ofile
- BString ofileName
- BString omode

7.35.1 Detailed Description

File operations class.

7.35.2 Constructor & Destructor Documentation

7.35.2.1 BFile::BFile ( )

7.35.2.2 BFile::BFile (const BFile & *file*)

Create opened specified file.

7.35.2.3 BFile::∼BFile ( )

7.35.3 Member Function Documentation

7.35.3.1 BError BFile::close ( )

Close file.

7.35.3.2 char ∗ BFile::fgets (char ∗ *buf,* size_t *size*)

7.35.3.3 BString BFile::fileName ( )

Return file name.

7.35.3.4 BError BFile::flush ( )

Flush the file.

7.35.3.5 FILE ∗ BFile::getFd ( )

File descriptor.

7.35.3.6 int BFile::isEnd ( )

Returns 1 if at the end of the file, 0 otherwise.

7.35.3.7 int BFile::isOpen ( )

Returns 1 if the file is open.

7.35.3.8 BUInt64 BFile::length ( )

File size in bytes.

7.35.3.9 BError BFile::open (BString *name,* BString *mode*)

Open file.

7.35.3.10 BError BFile::open (FILE ∗ *file*)

Assign object to opened file handle.

7.35.3.11 BError BFile::open (int *fd,* BString *mode*)

Assign object to opened file descriptor.

7.35.3.12 BFile & BFile::operator= (const BFile & *file*)

7.35.3.13 BUInt64 BFile::position ( )

The files position.

7.35.3.14 int BFile::printf (const char ∗ *fmt,* *...*)

Formated print into the file.

7.35.3.15 int BFile::read (void ∗ *buf,* int *nbytes*)

Read from file.

7.35.3.16 int BFile::readString (BString & *str*)

Read string. (ref fgets)

7.35.3.17 int BFile::seek (BUInt64 *pos*)

Set seek position.

7.35.3.18 int BFile::setVBuf (char ∗ *buf,* int *mode,* size_t *size*)

Set stream buffering options.

7.35.3.19 BError BFile::truncate ( )

Truncate the file.

7.35.3.20 int BFile::write (const void ∗ *buf,* int *nbytes*)

Write to file.

7.35.3.21 int BFile::writeString (const BString & *str*)

Write string to file.

7.35.4 Member Data Documentation

7.35.4.1 FILE∗ BFile::ofile [private]

7.35.4.2 BString BFile::ofileName [private]

7.35.4.3 BString BFile::omode [private]

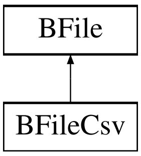
The documentation for this class was generated from the following files:

- /src/cern/tms/beam/libBeam/BFile.h
- /src/cern/tms/beam/libBeam/BFile.cpp

7.36 BFileCsv Class Reference

`#include <BFileCsv.h>`

Inheritance diagram for BFileCsv:



Public Member Functions

- BFileCsv (char separator= ';')
- BError readCsv (BStringList &csvList)
- BError writeCsv (BStringList &csvList)

Private Attributes

- char oseparator

7.36.1 Constructor & Destructor Documentation

7.36.1.1 BFileCsv::BFileCsv (char *separator* = ' ; ')

7.36.2 Member Function Documentation

7.36.2.1 BError BFileCsv::readCsv (BStringList & *csvList*)

7.36.2.2 BError BFileCsv::writeCsv (BStringList & *csvList*)

7.36.3 Member Data Documentation

7.36.3.1 char BFileCsv::oseparator [private]

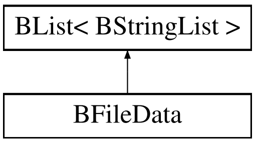
The documentation for this class was generated from the following files:

- /src/cern/tms/beam/libBeam/BFileCsv.h
- /src/cern/tms/beam/libBeam/BFileCsv.cpp

7.37 BFileData Class Reference

```
#include <BFileData.h>
```

Inheritance diagram for BFileData:



Public Member Functions

- BError open (BString filename)
- BError getNextId (int &id)
- BError find (int id, BStringList &csvList)
- BError write (int id, BStringList &csvList)
- BError del (int id)

Private Member Functions

- BError read ()
- BError write ()

Private Attributes

- BString ofilename

Additional Inherited Members

7.37.1 Member Function Documentation

7.37.1.1 BError BFileData::del (int *id*)

7.37.1.2 BError BFileData::find (int *id,* BStringList & *csvList*)

7.37.1.3 BError BFileData::getNextId (int & *id*)

7.37.1.4 BError BFileData::open (BString *filename*)

7.37.1.5 BError BFileData::read ( ) [private]

7.37.1.6 BError BFileData::write (int *id,* BStringList & *csvList*)

7.37.1.7 BError BFileData::write ( ) [private]

7.37.2 Member Data Documentation

7.37.2.1 BString BFileData::ofilename [private]

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

- /src/cern/tms/beam/libBeam/BFileData.h
- /src/cern/tms/beam/libBeam/BFileData.cpp

7.38 BIter Class Reference

Iterator for BList.

`#include <BList.h>`

Public Member Functions

- BIter (BNode ∗i=0)
- operator BNode ∗ ()
- int operator== (const BIter &i)
- int valid ()

Private Attributes

- BNode ∗ oi

7.38.1 Detailed Description

Iterator for BList.

7.38.2 Constructor & Destructor Documentation

7.38.2.1 BIter::BIter (BNode ∗ *i* = 0) `[inline]`

7.38.3 Member Function Documentation

7.38.3.1 BIter::operator BNode ∗ ( ) `[inline]`

7.38.3.2 int BIter::operator== (const BIter & *i*) `[inline]`

7.38.3.3 int BIter::valid ( ) `[inline]`

7.38.4 Member Data Documentation

7.38.4.1 BNode∗ BIter::oi `[private]`

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

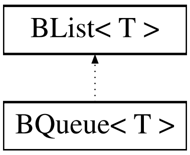
- /src/cern/tms/beam/libBeam/BList.h

7.39 BList< T > Class Template Reference

Template based list class.

`#include <BList.h>`

Inheritance diagram for BList< T >:



Classes

- class Node

Public Types

- typedef int(∗ SortFunc)(T &a, T &b)

  Prototype for sorting function.

Public Member Functions

- BList ()
- BList (const BList< T > &l)
- virtual ∼BList ()
- void start (BIter &i) const

  Iterator to start of list.
- BIter begin () const

  Iterator for start of list.
- BIter end () const

  Iterator for end of list.
- BIter end (BIter &i) const

  Iterator for end of list.
- void next (BIter &i) const

  Iterator for next item in list.
- void prev (BIter &i)

  Iterator for previous item in list.
- BIter goTo (int pos) const

  Iterator for pos item in list.
- int position (BIter i)

  Postition in list item with iterator i.
- unsigned int number () const

  Number of items in list.
- unsigned int size () const

  Number of items in list.
- int isEnd (BIter &i) const

  True if iterator refers to last item.
- T & front ()

  Get first item in list.
- T & rear ()

  Get last item in list.
- T & get (BIter i)

  Get item specified by iterator in list.
- const T & get (BIter i) const

  Get item specified by iterator in list.

- void append (const T &item)

  Append item to list.
- virtual void insert (BIter &i, const T &item)

  Insert item before item.
- void insertAfter (BIter &i, const T &item)

  Insert item after item.
- virtual void clear ()

  Clear the list.
- virtual void del (BIter &i)

  Delete specified item.
- void deleteLast ()

  Delete last item.
- void deleteFirst ()

  Delete fisrt item.
- void push (const T &i)

  Push item onto list.
- T pop ()

  Pop item from list deleteing item.
- void queueAdd (const T &i)

  Add item to end of list.
- T queueGet ()

  Get item from front of list deleteing item.
- void append (const BList< T > &l)

  Append list to list.
- int has (const T &i) const

  Checks if the item is in the list.
- void swap (BIter i1, BIter i2)

  Swap two items in list.
- void sort ()

  Sort list based on get(i) values.
- void sort (SortFunc func)

  Sort list based on Sort func.
- BList< T > & operator= (const BList< T > &l)
- T & operator[] (int i)
- const T & operator[] (int i) const
- T & operator[] (BIter i)
- const T & operator[] (const BIter &i) const
- BList< T > operator+ (const BList< T > &l) const

Protected Member Functions

- virtual Node * nodeGet (BIter i)
- virtual const Node * nodeGet (BIter i) const
- virtual Node * nodeCreate (const T &item)

Protected Attributes

- Node * onodes
- unsigned int olength

Private Member Functions

- virtual Node ∗ nodeCreate ()

7.39.1 Detailed Description

template<class T>class BList< T >

Template based list class.

7.39.2 Member Typedef Documentation

7.39.2.1 template<class T> typedef int(∗ BList< T >::SortFunc)(T &a, T &b)

Prototype for sorting function.

7.39.3 Constructor & Destructor Documentation

7.39.3.1 template<class T > BList< T >::BList ( )

7.39.3.2 template<class T> BList< T >::BList (const BList< T > & *l*)

7.39.3.3 template<class T > BList< T >::∼BList ( ) [virtual]

7.39.4 Member Function Documentation

7.39.4.1 template<class T> void BList< T >::append (const T & *item*)

Append item to list.

7.39.4.2 template<class T> void BList< T >::append (const BList< T > & *l*)

Append list to list.

7.39.4.3 template<class T > BIter BList< T >::begin ( ) const

Iterator for start of list.

7.39.4.4 template<class T > void BList< T >::clear ( ) [virtual]

Clear the list.

Reimplemented in BEntryFile, BDir, BEntryList, BDict< Type >, BQueue< T >, and BQueue< BoapMcPacket >.

7.39.4.5 template<class T > void BList< T >::del (BIter & *i*) [virtual]

Delete specified item.

Reimplemented in BEntryList, and BDict< Type >.

7.39.4.6 template<class T > void BList< T >::deleteFirst ( )

Delete fisrt item.

7.39.4.7 template<class T > void BList< T >::deleteLast ( )

Delete last item.

7.39.4.8 template<class T > BIter BList< T >::end ( ) const

Iterator for end of list.

7.39.4.9 template<class T > BIter BList< T >::end (BIter & *i*) const

Iterator for end of list.

7.39.4.10 template<class T > T & BList< T >::front ( )

Get first item in list.

7.39.4.11 template<class T > T & BList< T >::get (BIter *i*)

Get item specified by iterator in list.

7.39.4.12 template<class T > const T & BList< T >::get (BIter *i*) const

Get item specified by iterator in list.

7.39.4.13 template<class T > BIter BList< T >::goTo (int *pos*) const

Iterator for pos item in list.

7.39.4.14 template<class T> int BList< T >::has (const T & *i*) const

Checks if the item is in the list.

7.39.4.15 template<class T> void BList< T >::insert (BIter & *i*, const T & *item*) [virtual]

Insert item before item.

Reimplemented in BEntryList, and BDict< Type >.

7.39.4.16 template<class T> void BList< T >::insertAfter (BIter & *i*, const T & *item*)

Insert item after item.

7.39.4.17 template<class T > int BList< T >::isEnd (BIter & *i*) const

True if iterator refers to last item.

7.39.4.18 template< class T > void BList< T >::next (BIter & *i*) const

Iterator for next item in list.

7.39.4.19 template< class T> BList< T >::Node * BList< T >::nodeCreate (const T & *item*) [protected], [virtual]

7.39.4.20 template< class T> BList< T >::Node * BList< T >::nodeCreate ( ) [private], [virtual]

7.39.4.21 template< class T > BList< T >::Node * BList< T >::nodeGet (BIter *i*) [protected], [virtual]

7.39.4.22 template< class T > const BList< T >::Node * BList< T >::nodeGet (BIter *i*) const [protected], [virtual]

7.39.4.23 template< class T > unsigned int BList< T >::number ( ) const

Number of items in list.

7.39.4.24 template< class T> BList< T > BList< T >::operator+ (const BList< T > & *l*) const

7.39.4.25 template< class T> BList< T > & BList< T >::operator= (const BList< T > & *l*)

7.39.4.26 template< class T > T & BList< T >::operator[] (int *i*)

7.39.4.27 template< class T > const T & BList< T >::operator[] (int *i*) const

7.39.4.28 template< class T > T & BList< T >::operator[] (BIter *i*)

7.39.4.29 template< class T > const T & BList< T >::operator[] (const BIter & *i*) const

7.39.4.30 template< class T > T BList< T >::pop ( )

Pop item from list deleteing item.

7.39.4.31 template< class T > int BList< T >::position (BIter *i*)

Postition in list item with iterator i.

7.39.4.32 template< class T > void BList< T >::prev (BIter & *i*)

Iterator for previous item in list.

7.39.4.33 template< class T> void BList< T >::push (const T & *i*)

Push item onto list.

7.39.4.34 template< class T> void BList< T >::queueAdd (const T & *i*)

Add item to end of list.

7.39.4.35 template< class T > T BList< T >::queueGet ( )

Get item from front of list deleteing item.

7.39.4.36 template< class T > T & BList< T >::rear ( )

Get last item in list.

7.39.4.37 template< class T > unsigned int BList< T >::size ( ) const

Number of items in list.

7.39.4.38 template< class T > void BList< T >::sort ( )

Sort list based on get(i) values.

7.39.4.39 template< class T > void BList< T >::sort (SortFunc *func*)

Sort list based on Sort func.

7.39.4.40 template< class T > void BList< T >::start (BIter & *i*) const

Iterator to start of list.

7.39.4.41 template< class T > void BList< T >::swap (BIter *i1*, BIter *i2*)

Swap two items in list.

7.39.5 Member Data Documentation

7.39.5.1 template< class T> unsigned int BList< T >::olength `[protected]`

7.39.5.2 template< class T> Node∗ BList< T >::onodes `[protected]`

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

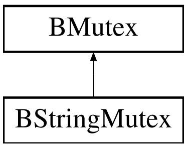
- /src/cern/tms/beam/libBeam/BList.h
- /src/cern/tms/beam/libBeam/BList_func.h

7.40 BMutex Class Reference

Mutex class.

`#include <BMutex.h>`

Inheritance diagram for BMutex:



Public Types

- enum Type { Normal, Recursive }

Public Member Functions

- BMutex (Type type=Normal)
- BMutex (const BMutex &mutex)
- ∼BMutex ()
- int lock ()

  Set lock, wait as necessary.
- int timedLock (int timeoutUs)

  Set lock, wait as necessary but timeout after given time.
- int unlock ()

  Unlock the lock.
- int tryLock ()

  Test the lock.
- BMutex & operator= (const BMutex &mutex)

Private Attributes

- pthread_mutex_t omutex

7.40.1 Detailed Description

Mutex class.

7.40.2 Member Enumeration Documentation

7.40.2.1 enum **BMutex::Type**

Enumerator

Normal

Recursive

7.40.3 Constructor & Destructor Documentation

7.40.3.1 BMutex::BMutex (**Type** *type* = **Normal**)

7.40.3.2 BMutex::BMutex (const **BMutex** & *mutex*)

7.40.3.3 BMutex::∼BMutex ( )

7.40.4 Member Function Documentation

7.40.4.1 int BMutex::lock ( )

Set lock, wait as necessary.

7.40.4.2 BMutex & BMutex::operator= (const BMutex & *mutex*)

7.40.4.3 int BMutex::timedLock (int *timeoutUs*)

Set lock, wait as necessary but timeout after given time.

7.40.4.4 int BMutex::tryLock ( )

Test the lock.

7.40.4.5 int BMutex::unlock ( )

Unlock the lock.

7.40.5 Member Data Documentation

7.40.5.1 pthread_mutex_t BMutex::omutex [private]

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

- /src/cern/tms/beam/libBeam/BMutex.h
- /src/cern/tms/beam/libBeam/BMutex.cpp

7.41 BMutexLock Class Reference

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

Public Member Functions

- BMutexLock (BMutex &lock, int doLock=0)
- ∼BMutexLock ()
- int lock ()
- int unlock ()

Private Attributes

- BMutex & olock

7.41.1 Constructor & Destructor Documentation

7.41.1.1 BMutexLock::BMutexLock (BMutex & *lock*, int *doLock* = 0) [inline]

7.41.1.2 BMutexLock::∼BMutexLock ( ) [inline]

7.41.2 Member Function Documentation

7.41.2.1 int BMutexLock::lock ( ) [inline]

7.41.2.2 int BMutexLock::unlock ( ) [inline]

7.41.3 Member Data Documentation

7.41.3.1 BMutex& BMutexLock::olock [private]

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

- /src/cern/tms/beam/libBeam/BMutex.h

7.42 BMysql Class Reference

`#include <BMysql.h>`

Public Member Functions

- BMysql ()
- ∼BMysql ()
- BError open (BString hostName, BString dataBase, BString userName, BString password)
- void close ()
- BError get (BString table, BString where, BDictString &fields)
- BError insert (BString table, BDictString fields, BUInt32 ∗id=0)
- BError update (BString table, BUInt32 id, BDictString fields)
- BError del (BString table, BUInt32 id)

  Delete record from table.
- BError flush ()

  Flush all data to disk.
- BString escapeString (BString str)

  Escapes special characters in the string.
- BError query (BString cmd, BList< BDictString > &result)
- MYSQL & db ()
- void setDebug (int debug)

Private Attributes

- MYSQL odb
- int oopened
- int odebug
- BMutex olock

7.42.1 Constructor & Destructor Documentation

7.42.1.1 BMysql::BMysql ( )

7.42.1.2 BMysql::∼BMysql ( )

7.42.2 Member Function Documentation

7.42.2.1 void BMysql::close ( )

7.42.2.2 MYSQL & BMysql::db ( )

7.42.2.3 BError BMysql::del (BString *table,* BUInt32 *id*)

Delete record from table.

7.42.2.4 BString BMysql::escapeString (BString *str*)

Escapes special characters in the string.

7.42.2.5 BError BMysql::flush ( )

Flush all data to disk.

7.42.2.6 BError BMysql::get (BString *table,* BString *where,* BDictString & *fields*)

7.42.2.7 BError BMysql::insert (BString *table,* BDictString *fields,* BUInt32 ∗ *id* = 0)

7.42.2.8 BError BMysql::open (BString *hostName,* BString *dataBase,* BString *userName,* BString *password*)

7.42.2.9 BError BMysql::query (BString *cmd,* BList< BDictString > & *result*)

7.42.2.10 void BMysql::setDebug (int *debug*)

7.42.2.11 BError BMysql::update (BString *table,* BUInt32 *id,* BDictString *fields*)

7.42.3 Member Data Documentation

7.42.3.1 MYSQL BMysql::odb `[private]`

7.42.3.2 int BMysql::odebug `[private]`

7.42.3.3 BMutex BMysql::olock `[private]`

7.42.3.4 int BMysql::oopened `[private]`

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

- /src/cern/tms/beam/libBeam/BMysql.h
- /src/cern/tms/beam/libBeam/BMysql.cpp

7.43 BNameValue< T > Class Template Reference

```
#include <BNameValue.h>
```

Public Member Functions

- BNameValue ()
- BNameValue (BString name, const T &value)
- BString getName ()
- T & getValue ()

Private Attributes

- BString oname
- T ovalue

7.43.1 Constructor & Destructor Documentation

7.43.1.1 template<class T > BNameValue< T >::BNameValue ( ) [inline]

7.43.1.2 template<class T > BNameValue< T >::BNameValue (BString *name*, const T & *value*) [inline]

7.43.2 Member Function Documentation

7.43.2.1 template<class T > BString BNameValue< T >::getName ( ) [inline]

7.43.2.2 template<class T > T& BNameValue< T >::getValue ( ) [inline]

7.43.3 Member Data Documentation

7.43.3.1 template<class T > BString BNameValue< T >::oname [private]

7.43.3.2 template<class T > T BNameValue< T >::ovalue [private]

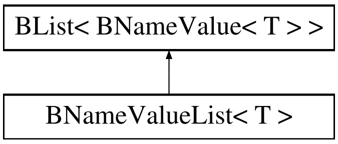
The documentation for this class was generated from the following file:

- /src/cern/tms/beam/libBeam/BNameValue.h

7.44 BNameValueList< T > Class Template Reference

```
#include <BNameValue.h>
```

Inheritance diagram for BNameValueList< T >:



Public Member Functions

- T ∗ find (BString name)
- BIter findPos (BString name)

Additional Inherited Members

7.44.1 Member Function Documentation

7.44.1.1 template<class T > T∗ BNameValueList< T >::find (BString *name*) [inline]

7.44.1.2 template<class T > BIter BNameValueList< T >::findPos (BString *name*) [inline]

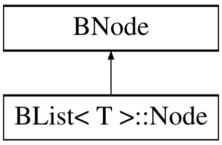
The documentation for this class was generated from the following file:

- /src/cern/tms/beam/libBeam/BNameValue.h

7.45 BNode Class Reference

```
#include <BList.h>
```

Inheritance diagram for BNode:



Public Member Functions

- BNode ()

Public Attributes

- BNode ∗ next
- BNode ∗ prev

7.45.1 Constructor & Destructor Documentation

7.45.1.1 BNode::BNode ( ) [inline]

7.45.2 Member Data Documentation

7.45.2.1 BNode∗ BNode::next

7.45.2.2 BNode∗ BNode::prev

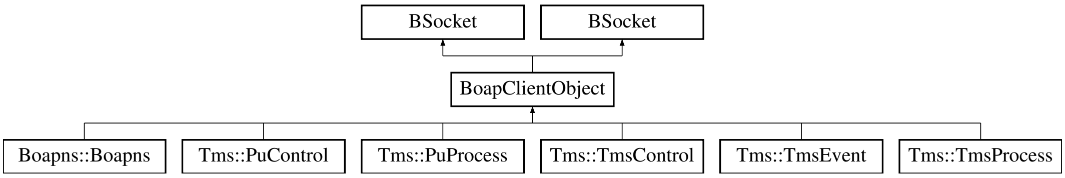
The documentation for this class was generated from the following file:

- /src/cern/tms/beam/libBeam/BList.h

7.46 BoapClientObject Class Reference

```
#include <BoapSimple.h>
```

Inheritance diagram for BoapClientObject:



Public Member Functions

- BoapClientObject (BString name="")
- virtual ∼BoapClientObject ()

- BError connectService (BString name)

    Connects to the named service.
- BError disconnectService ()

    Disconnects from the named service.
- BString getServiceName ()

    Get the name of the service.
- BError ping (BUInt32 &apiVersion)

    Pings the connection and finds the remotes version number.
- BError setConnectionPriority (BoapPriority priority)

    Sets the connection priority.
- void setMaxLength (BUInt32 maxLength)

    Sets the maximum packet length.
- void setTimeout (int timeout)

    Sets the timeout in micro seconds. -1 is wait indefinitely.
- BoapClientObject (BString name)
- BError connectService (BString name)

Protected Member Functions

- BError pingLocked (BUInt32 &apiVersion)
- BError checkApiVersion ()
- BError performCall (BoapPacket &tx, BoapPacket &rx)

    Performs a RPC call to the named service.
- BError performSend (BoapPacket &tx)

    Performs a send to the named service.
- BError performRecv (BoapPacket &rx)

    Performs a receive.
- virtual BError handleReconnect (BError err)

    Handle a reconnect performing autorisaztion if required.
- BError performSend (BoapPacket &tx)
- BError performRecv (BoapPacket &rx)
- BError performCall (BoapPacket &tx, BoapPacket &rx)

Protected Attributes

- BString oname
- BUInt32 oapiVersion
- BoapPriority opriority
- BoapService oservice
- int oconnected
- BUInt32 omaxLength
- BoapPacket otx
- BoapPacket orx
- BMutex olock
- int otimeout
- int oreconnect

    Handle an automatic reconnect on timeout.

Additional Inherited Members

7.46.1 Constructor & Destructor Documentation

7.46.1.1 BoapClientObject::BoapClientObject (**BString** *name* = "")

7.46.1.2 BoapClientObject::∼BoapClientObject ( ) [virtual]

7.46.1.3 BoapClientObject::BoapClientObject (**BString** *name*)

7.46.2 Member Function Documentation

7.46.2.1 **BError** BoapClientObject::checkApiVersion ( ) [protected]

7.46.2.2 **BError** BoapClientObject::connectService (**BString** *name*)

Connects to the named service.

7.46.2.3 **BError** BoapClientObject::connectService (**BString** *name*)

7.46.2.4 **BError** BoapClientObject::disconnectService ( )

Disconnects from the named service.

7.46.2.5 **BString** BoapClientObject::getServiceName ( )

Get the name of the service.

7.46.2.6 **BError** BoapClientObject::handleReconnect (**BError** *err*) [protected], [virtual]

Handle a reconnect performing autorisaztion if required.

7.46.2.7 **BError** BoapClientObject::performCall (**BoapPacket** & *tx*, **BoapPacket** & *rx*) [protected]

Performs a RPC call to the named service.

7.46.2.8 **BError** BoapClientObject::performCall (**BoapPacket** & *tx*, **BoapPacket** & *rx*) [protected]

7.46.2.9 **BError** BoapClientObject::performRecv (**BoapPacket** & *rx*) [protected]

Performs a receive.

7.46.2.10 **BError** BoapClientObject::performRecv (**BoapPacket** & *rx*) [protected]

7.46.2.11 **BError** BoapClientObject::performSend (**BoapPacket** & *tx*) [protected]

Performs a send to the named service.

7.46.2.12 **BError** BoapClientObject::performSend (**BoapPacket** & *tx*) [protected]

7.46.2.13 BError BoapClientObject::ping (BUInt32 & *apiVersion*)

Pings the connection and finds the remotes version number.

7.46.2.14 BError BoapClientObject::pingLocked (BUInt32 & *apiVersion*) [protected]

7.46.2.15 BError BoapClientObject::setConnectionPriority (BoapPriority *priority*)

Sets the connection priority.

7.46.2.16 void BoapClientObject::setMaxLength (BUInt32 *maxLength*)

Sets the maximum packet length.

7.46.2.17 void BoapClientObject::setTimeout (int *timeout*)

Sets the timeout in micro seconds. -1 is wait indefinitely.

7.46.3 Member Data Documentation

7.46.3.1 BUInt32 BoapClientObject::oapiVersion [protected]

7.46.3.2 int BoapClientObject::oconnected [protected]

7.46.3.3 BMutex BoapClientObject::olock [protected]

7.46.3.4 BUInt32 BoapClientObject::omaxLength [protected]

7.46.3.5 BString BoapClientObject::oname [protected]

7.46.3.6 BoapPriority BoapClientObject::opriority [protected]

7.46.3.7 int BoapClientObject::oreconnect [protected]

Handle an automatic reconnect on timeout.

7.46.3.8 BoapPacket BoapClientObject::orx [protected]

7.46.3.9 BoapService BoapClientObject::oservice [protected]

7.46.3.10 int BoapClientObject::otimeout [protected]

7.46.3.11 BoapPacket BoapClientObject::otx [protected]

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

- /src/cern/tms/beam/libBeam/Boap.h
- /src/cern/tms/beam/libBeam/BoapSimple.h
- /src/cern/tms/beam/libBeam/Boap.cpp
- /src/cern/tms/beam/libBeam/BoapSimple.cc

7.47 Boapns::BoapEntry Class Reference

```
#include <BoapnsD.h>
```

Public Member Functions

- BoapEntry (BString pname=BString(), BString phostName=BString(), BList< BString > paddressList=BList< BString >(), BUInt32 pport=BUInt32(), BUInt32 pservice=BUInt32())

Public Attributes

- BString name
- BString hostName
- BList< BString > addressList
- BUInt32 port
- BUInt32 service

7.47.1 Constructor & Destructor Documentation

7.47.1.1 Boapns::BoapEntry::BoapEntry (BString *pname* = BString (), BString *phostName* = BString (), BList< BString > *paddressList* = BList<BString >(), BUInt32 *pport* = BUInt32 (), BUInt32 *pservice* = BUInt32 ())

7.47.2 Member Data Documentation

7.47.2.1 BList<BString > Boapns::BoapEntry::addressList

7.47.2.2 BString Boapns::BoapEntry::hostName

7.47.2.3 BString Boapns::BoapEntry::name

7.47.2.4 BUInt32 Boapns::BoapEntry::port

7.47.2.5 BUInt32 Boapns::BoapEntry::service

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

- /src/cern/tms/beam/libBeam/BoapnsD.h
- /src/cern/tms/beam/libBeam/BoapnsD.cpp

7.48 BoapFuncEntry Class Reference

```
#include <BoapSimple.h>
```

Public Member Functions

- BoapFuncEntry (int cmd, BoapFunc func)
- BoapFuncEntry (int cmd, BoapFunc func)

Public Attributes

- BUInt32 ocmd
- BoapFunc ofunc
- UInt32 ocmd

7.48.1 Constructor & Destructor Documentation

7.48.1.1 BoapFuncEntry::BoapFuncEntry (int *cmd*, **BoapFunc** *func*)

7.48.1.2 BoapFuncEntry::BoapFuncEntry (int *cmd*, **BoapFunc** *func*)

7.48.2 Member Data Documentation

7.48.2.1 **UInt32** BoapFuncEntry::ocmd

7.48.2.2 **BUInt32** BoapFuncEntry::ocmd

7.48.2.3 **BoapFunc** BoapFuncEntry::ofunc

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

- /src/cern/tms/beam/libBeam/Boap.h
- /src/cern/tms/beam/libBeam/BoapSimple.h
- /src/cern/tms/beam/libBeam/Boap.cpp
- /src/cern/tms/beam/libBeam/BoapSimple.cc

7.49 BoapMcClientObject Class Reference

```
#include <BoapMc.h>
```

Public Member Functions

- BoapMcClientObject (BComms &comms)
- virtual ∼BoapMcClientObject ()
- void setAddress (BUInt8 addressTo, BUInt8 addressFrom)
- BUInt32 getApiVersion ()

  Returns the API version.

Protected Member Functions

- BError performCall ()

  Performs a RPC call to the named service.
- BError performSend ()

  Performs a send to the named service.
- BError performRecv ()

  Performs a receive.

Protected Attributes

- BUInt32 oapiVersion
- BComms & ocomms
- BUInt8 oaddressTo
- BUInt8 oaddressFrom
- BoapMcPacket opacket

7.49.1 Constructor & Destructor Documentation

7.49.1.1 BoapMcClientObject::BoapMcClientObject (BComms & *comms*)

7.49.1.2 BoapMcClientObject::∼BoapMcClientObject ( ) [virtual]

7.49.2 Member Function Documentation

7.49.2.1 BUInt32 BoapMcClientObject::getApiVersion ( )

Returns the API version.

7.49.2.2 BError BoapMcClientObject::performCall ( ) [protected]

Performs a RPC call to the named service.

7.49.2.3 BError BoapMcClientObject::performRecv ( ) [protected]

Performs a receive.

7.49.2.4 BError BoapMcClientObject::performSend ( ) [protected]

Performs a send to the named service.

7.49.2.5 void BoapMcClientObject::setAddress (BUInt8 *addressTo,* BUInt8 *addressFrom*)

7.49.3 Member Data Documentation

7.49.3.1 BUInt8 BoapMcClientObject::oaddressFrom [protected]

7.49.3.2 BUInt8 BoapMcClientObject::oaddressTo [protected]

7.49.3.3 BUInt32 BoapMcClientObject::oapiVersion [protected]

7.49.3.4 BComms& BoapMcClientObject::ocomms [protected]

7.49.3.5 BoapMcPacket BoapMcClientObject::opacket [protected]

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

- /src/cern/tms/beam/libBeam/BoapMc.h
- /src/cern/tms/beam/libBeam/BoapMc.cpp

7.50 BoapMcComms Class Reference

```
#include <BoapMc.h>
```

Public Member Functions

- BoapMcComms (Bool threaded=0, BUInt rxQueueSize=4)
- virtual ∼BoapMcComms ()
- void setCommsMode (Bool slave, BUInt txQueueSize)

  Sets slave mode.
- void setComms (BComms &comms)

  Sets the communications interface to use.
- void setComms (BComms ∗comms)

  Sets the communications interface to use.
- void setAddress (BUInt8 addressTo, BUInt8 addressFrom)

  Sets the to and from addresses.
- BUInt32 getApiVersion ()

  Returns the API version.
- BUInt32 setTimeout (BUInt32 timeoutUs)

  Sets the call timeout returning the current value.
- virtual BError processRx (BTimeout timeoutUs=BTimeoutForever)

  Process any RX packets queuing them as needed.
- virtual BError processRequests (BTimeout timeoutUs=BTimeoutForever)

  Check and process all requests.
- virtual BError processRequest (BTimeout timeoutUs=BTimeoutForever)

  Check and process any request.
- virtual BError processPacket (BoapMcPacket &rx, BoapMcPacket &tx)

  Process a recieved packet.

Protected Member Functions

- BError performCall ()

  Performs a RPC call to the remote side.
- BError performSend ()

  Performs a RPC send to the remote side.
- BError packetSend (BoapMcPacket &packet)

  Receives a packet.
- BError packetRecv (BoapMcPacket &packet)

  Receives a packet.

Protected Attributes

- Bool othreaded
- BMutex olockCall

  Lock for RPC calls. Only one at a time.
- BMutex olockTx

  Lock for TX.
- BComms ∗ ocomms
- BUInt32 oapiVersion
- Bool oslave

Set slave mode.

- BUInt32 otimeout

  The timeout in us for calls.
- BUInt8 oaddressTo
- BUInt8 oaddressFrom
- BoapMcPacket opacket

  Packet RX buffer.
- BoapMcPacket opacketTx

  Packet TX buffer for calls.
- BoapMcPacket opacketRx

  Packet RX buffer for calls.
- BSemaphore opacketRxSema

  Wait RX semaphore.
- BoapMcPacket opacketReqTx

  Packet TX buffer for requests.
- BoapMcPacket opacketReqRx

  Packet RX buffer for requests.
- BQueue< BoapMcPacket > opacketReqQueue

  Packet RX buffer queue for requests.
- BFifo< BoapMcPacket > opacketTxQueue

  Packet TX Queue.
- BSemaphoreCount opacketTxQueueWriteNum

  Packet TX Queue number.
- BSemaphore opacketTxSema

  Wait for TX semaphore.

7.50.1 Constructor & Destructor Documentation

7.50.1.1 BoapMcComms::BoapMcComms (Bool *threaded* = 0, BUInt *rxQueueSize* = 4)

7.50.1.2 BoapMcComms::~BoapMcComms ( ) [virtual]

7.50.2 Member Function Documentation

7.50.2.1 BUInt32 BoapMcComms::getApiVersion ( )

Returns the API version.

7.50.2.2 BError BoapMcComms::packetRecv (BoapMcPacket & *packet*) [protected]

Receives a packet.

7.50.2.3 BError BoapMcComms::packetSend (BoapMcPacket & *packet*) [protected]

Receives a packet.

7.50.2.4 BError BoapMcComms::performCall ( ) [protected]

Performs a RPC call to the remote side.

7.50.2.5 BError BoapMcComms::performSend ( ) [protected]

Performs a RPC send to the remote side.

7.50.2.6 BError BoapMcComms::processPacket (BoapMcPacket & *rx*, BoapMcPacket & *tx*) [virtual]

Process a recieved packet.

7.50.2.7 BError BoapMcComms::processRequest (BTimeout *timeoutUs* = BTimeoutForever) [virtual]

Check and process any request.

7.50.2.8 BError BoapMcComms::processRequests (BTimeout *timeoutUs* = BTimeoutForever) [virtual]

Check and process all requests.

7.50.2.9 BError BoapMcComms::processRx (BTimeout *timeoutUs* = BTimeoutForever) [virtual]

Process any RX packets queuing them as needed.

!!! This should wait on comms for timeoutUs !!!

7.50.2.10 void BoapMcComms::setAddress (BUInt8 *addressTo*, BUInt8 *addressFrom*)

Sets the to and from addresses.

7.50.2.11 void BoapMcComms::setComms (BComms & *comms*)

Sets the communications interface to use.

7.50.2.12 void BoapMcComms::setComms (BComms * *comms*)

Sets the communications interface to use.

7.50.2.13 void BoapMcComms::setCommsMode (Bool *slave*, BUInt *txQueueSize*)

Sets slave mode.

7.50.2.14 BUInt32 BoapMcComms::setTimeout (BUInt32 *timeoutUs*)

Sets the call timeout returning the current value.

7.50.3 Member Data Documentation

7.50.3.1 BUInt8 BoapMcComms::oaddressFrom [protected]

7.50.3.2 BUInt8 BoapMcComms::oaddressTo [protected]

7.50.3.3 BUInt32 BoapMcComms::oapiVersion [protected]

7.50.3.4 BComms∗ BoapMcComms::ocomms [protected]

7.50.3.5 BMutex BoapMcComms::olockCall [protected]

Lock for RPC calls. Only one at a time.

7.50.3.6 BMutex BoapMcComms::olockTx [protected]

Lock for TX.

7.50.3.7 BoapMcPacket BoapMcComms::opacket [protected]

Packet RX buffer.

7.50.3.8 BQueue<BoapMcPacket> BoapMcComms::opacketReqQueue [protected]

Packet RX buffer queue for requests.

7.50.3.9 BoapMcPacket BoapMcComms::opacketReqRx [protected]

Packet RX buffer for requests.

7.50.3.10 BoapMcPacket BoapMcComms::opacketReqTx [protected]

Packet TX buffer for requests.

7.50.3.11 BoapMcPacket BoapMcComms::opacketRx [protected]

Packet RX buffer for calls.

7.50.3.12 BSemaphore BoapMcComms::opacketRxSema [protected]

Wait RX semaphore.

7.50.3.13 BoapMcPacket BoapMcComms::opacketTx [protected]

Packet TX buffer for calls.

7.50.3.14 BFifo<BoapMcPacket> BoapMcComms::opacketTxQueue [protected]

Packet TX Queue.

7.50.3.15 BSemaphoreCount BoapMcComms::opacketTxQueueWriteNum [protected]

Packet TX Queue number.

7.50.3.16 BSemaphore BoapMcComms::opacketTxSema [protected]

Wait for TX semaphore.

7.50.3.17 Bool BoapMcComms::oslave [protected]

Set slave mode.

7.50.3.18 Bool BoapMcComms::othreaded [protected]

7.50.3.19 BUInt32 BoapMcComms::otimeout [protected]

The timeout in us for calls.

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

- /src/cern/tms/beam/libBeam/BoapMc.h
- /src/cern/tms/beam/libBeam/BoapMc.cpp

7.51 BoapMcPacket Class Reference

```
#include <BoapMc.h>
```

Public Attributes

- BoapMcPacketHead head
- char data [256-sizeof(BoapMcPacketHead)]

7.51.1 Member Data Documentation

7.51.1.1 char BoapMcPacket::data[256-sizeof(BoapMcPacketHead)]

7.51.1.2 BoapMcPacketHead BoapMcPacket::head

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

- /src/cern/tms/beam/libBeam/BoapMc.h

7.52 BoapMcPacketHead Struct Reference

```
#include <BoapMc.h>
```

Public Attributes

- BUInt8 length
- BUInt8 addressTo
- BUInt8 addressFrom
- BUInt8 cmd
- BUInt16 error
- BUInt16 checksum

7.52.1 Member Data Documentation

7.52.1.1 BUInt8 BoapMcPacketHead::addressFrom

7.52.1.2 BUInt8 BoapMcPacketHead::addressTo

7.52.1.3 BUInt16 BoapMcPacketHead::checksum

7.52.1.4 BUInt8 BoapMcPacketHead::cmd

7.52.1.5 BUInt16 BoapMcPacketHead::error

7.52.1.6 BUInt8 BoapMcPacketHead::length

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

- /src/cern/tms/beam/libBeam/BoapMc.h

7.53 BoapMcServiceObject Class Reference

`#include <BoapMc.h>`

Public Member Functions

- BoapMcServiceObject ()
- virtual ∼BoapMcServiceObject ()
- virtual BError process (BoapMcPacket &rx, BoapMcPacket &tx)
- virtual BError processEvent (BoapMcPacket &rx)

Protected Member Functions

- BError sendEvent (BoapMcPacket &tx)

Protected Attributes

- BUInt32 oapiVersion

7.53.1 Constructor & Destructor Documentation

7.53.1.1 BoapMcServiceObject::BoapMcServiceObject ( )

7.53.1.2 BoapMcServiceObject::∼BoapMcServiceObject ( ) `[virtual]`

7.53.2 Member Function Documentation

7.53.2.1 BError BoapMcServiceObject::process (BoapMcPacket & *rx*, BoapMcPacket & *tx*) `[virtual]`

7.53.2.2 BError BoapMcServiceObject::processEvent (BoapMcPacket & *rx*) `[virtual]`

7.53.2.3 BError BoapMcServiceObject::sendEvent (BoapMcPacket & *tx*) `[protected]`

7.53.3 Member Data Documentation

7.53.3.1 BUInt32 BoapMcServiceObject::oapiVersion [protected]

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

- /src/cern/tms/beam/libBeam/BoapMc.h
- /src/cern/tms/beam/libBeam/BoapMc.cpp

7.54 BoapMcSignalObject Class Reference

```
#include <BoapMc.h>
```

Public Member Functions

- BoapMcSignalObject (BComms &comms)

Protected Member Functions

- BError performSend (BoapMcPacket &tx)

Protected Attributes

- BComms & ocomms

7.54.1 Constructor & Destructor Documentation

7.54.1.1 BoapMcSignalObject::BoapMcSignalObject (BComms & *comms*)

7.54.2 Member Function Documentation

7.54.2.1 BError BoapMcSignalObject::performSend (BoapMcPacket & *tx*) [protected]

7.54.3 Member Data Documentation

7.54.3.1 BComms& BoapMcSignalObject::ocomms [protected]

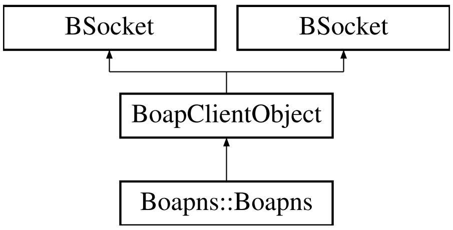
The documentation for this class was generated from the following files:

- /src/cern/tms/beam/libBeam/BoapMc.h
- /src/cern/tms/beam/libBeam/BoapMc.cpp

7.55 Boapns::Boapns Class Reference

```
#include <BoapnsC.h>
```

Inheritance diagram for Boapns::Boapns:



Public Member Functions

- Boapns (BString name="")
- BError getVersion (BString &version)
- BError getEntryList (BList< BoapEntry > &entryList)
- BError getEntry (BString name, BoapEntry &entry)
- BError addEntry (BoapEntry entry)
- BError delEntry (BString name)
- BError getNewName (BString &name)

Additional Inherited Members

7.55.1 Constructor & Destructor Documentation

7.55.1.1 Boapns::Boapns::Boapns (BString *name* = " ")

7.55.2 Member Function Documentation

7.55.2.1 BError Boapns::Boapns::addEntry (BoapEntry *entry*)

7.55.2.2 BError Boapns::Boapns::delEntry (BString *name*)

7.55.2.3 BError Boapns::Boapns::getEntry (BString *name,* BoapEntry & *entry*)

7.55.2.4 BError Boapns::Boapns::getEntryList (BList< BoapEntry > & *entryList*)

7.55.2.5 BError Boapns::Boapns::getNewName (BString & *name*)

7.55.2.6 BError Boapns::Boapns::getVersion (BString & *version*)

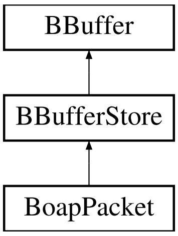
The documentation for this class was generated from the following files:

- /src/cern/tms/beam/libBeam/BoapnsC.h
- /src/cern/tms/beam/libBeam/BoapnsC.cpp

7.56 BoapPacket Class Reference

```
#include <BoapSimple.h>
```

Inheritance diagram for BoapPacket:



Public Member Functions

- BoapPacket ()
- ∼BoapPacket ()
- BUInt32 getCmd ()
- int peekHead (BoapPacketHead &head)
- int pushHead (BoapPacketHead &head)
- int popHead (BoapPacketHead &head)
- void updateHead ()
- BoapPacket ()
- ∼BoapPacket ()
- int resize (int size)
- BError setData (void ∗data, int nbytes)
- int nbytes ()
- char ∗ data ()
- int pushHead (BoapPacketHead &head)
- int push (Int8 v)
- int push (UInt8 v)
- int push (Int16 v)
- int push (UInt16 v)
- int push (Int32 v)
- int push (UInt32 v)
- int push (BString &v)
- int push (Double v)
- int push (BError &v)
- int push (UInt32 nBytes, const void ∗data)
- int popHead (BoapPacketHead &head)
- int pop (Int8 &v)
- int pop (UInt8 &v)
- int pop (Int16 &v)
- int pop (UInt16 &v)
- int pop (Int32 &v)
- int pop (UInt32 &v)
- int pop (BString &v)
- int pop (Double &v)
- int pop (BError &v)
- int pop (UInt32 nBytes, void ∗data)

Private Member Functions

- void updateLen ()

Private Attributes

- int osize
- int onbytes
- char ∗ odata
- int opos

Additional Inherited Members

7.56.1 Constructor & Destructor Documentation

7.56.1.1 BoapPacket::BoapPacket ( )

7.56.1.2 BoapPacket::∼BoapPacket ( )

7.56.1.3 BoapPacket::BoapPacket ( )

7.56.1.4 BoapPacket::∼BoapPacket ( )

7.56.2 Member Function Documentation

7.56.2.1 char ∗ BoapPacket::data ( )

7.56.2.2 BUInt32 BoapPacket::getCmd ( )

7.56.2.3 int BoapPacket::nbytes ( )

7.56.2.4 int BoapPacket::peekHead (BoapPacketHead & *head*)

7.56.2.5 int BoapPacket::pop (Int8 & *v*)

7.56.2.6 int BoapPacket::pop (UInt8 & *v*)

7.56.2.7 int BoapPacket::pop (Int16 & *v*)

7.56.2.8 int BoapPacket::pop (UInt16 & *v*)

7.56.2.9 int BoapPacket::pop (Int32 & *v*)

7.56.2.10 int BoapPacket::pop (UInt32 & *v*)

7.56.2.11 int BoapPacket::pop (BString & *v*)

7.56.2.12 int BoapPacket::pop (Double & *v*)

7.56.2.13 int BoapPacket::pop (BError & *v*)

7.56.2.14 int BoapPacket::pop (UInt32 *nBytes,* void ∗ *data*)

7.56.2.15 int BoapPacket::popHead (BoapPacketHead & *head*)

7.56.2.16 int BoapPacket::popHead (BoapPacketHead & *head*)

7.56.2.17 int BoapPacket::push (Int8 *v*)

7.56.2.18 int BoapPacket::push (UInt8 *v*)

7.56.2.19 int BoapPacket::push (Int16 *v*)

7.56.2.20 int BoapPacket::push (UInt16 *v*)

7.56.2.21 int BoapPacket::push (Int32 *v*)

7.56.2.22 int BoapPacket::push (UInt32 *v*)

7.56.2.23 int BoapPacket::push (BString & *v*)

7.56.2.24 int BoapPacket::push (Double *v*)

7.56.2.25 int BoapPacket::push (BError & *v*)

7.56.2.26 int BoapPacket::push (UInt32 *nBytes,* const void ∗ *data*)

7.56.2.27 int BoapPacket::pushHead (BoapPacketHead & *head*)

7.56.2.28 int BoapPacket::pushHead (BoapPacketHead & *head*)

7.56.2.29 int BoapPacket::resize (int *size*)

7.56.2.30 BError BoapPacket::setData (void ∗ *data,* int *nbytes*)

7.56.2.31 void BoapPacket::updateHead ( )

7.56.2.32 void BoapPacket::updateLen ( ) [private]

7.56.3 Member Data Documentation

7.56.3.1 char∗ BoapPacket::odata [private]

7.56.3.2 int BoapPacket::onbytes [private]

7.56.3.3 int BoapPacket::opos [private]

7.56.3.4 int BoapPacket::osize [private]

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

- /src/cern/tms/beam/libBeam/Boap.h
- /src/cern/tms/beam/libBeam/BoapSimple.h
- /src/cern/tms/beam/libBeam/Boap.cpp
- /src/cern/tms/beam/libBeam/BoapSimple.cc

7.57 BoapPacketHead Struct Reference

```
#include <BoapSimple.h>
```

Public Attributes

- BUInt32 type

- BUInt32 length
- BUInt32 service
- BUInt32 cmd
- UInt32 length
- BoapType type
- BoapService service
- UInt32 cmd
- UInt32 reserved [12]

7.57.1 Member Data Documentation

7.57.1.1 UInt32 BoapPacketHead::cmd

7.57.1.2 BUInt32 BoapPacketHead::cmd

7.57.1.3 UInt32 BoapPacketHead::length

7.57.1.4 BUInt32 BoapPacketHead::length

7.57.1.5 UInt32 BoapPacketHead::reserved[12]

7.57.1.6 BoapService BoapPacketHead::service

7.57.1.7 BUInt32 BoapPacketHead::service

7.57.1.8 BUInt32 BoapPacketHead::type

7.57.1.9 BoapType BoapPacketHead::type

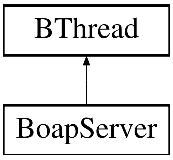
The documentation for this struct was generated from the following files:

- /src/cern/tms/beam/libBeam/Boap.h
- /src/cern/tms/beam/libBeam/BoapSimple.h

7.58 BoapServer Class Reference

```
#include <BoapSimple.h>
```

Inheritance diagram for BoapServer:



Public Types

- enum { NOTHREADS =0, THREADED =1 }

Public Member Functions

- BoapServer ()
- virtual ∼BoapServer ()
- virtual BError init (BString boapNsHost="", int port=0, int threaded=0, int isBoapns=0)
- virtual BError run (int inThread=0)
- virtual BError process (BoapServerConnection ∗conn, BoapPacket &rx, BoapPacket &tx)
- virtual BError processEvent (BoapPacket &rx)
- virtual BError addObject (BoapServiceObject ∗object)
- virtual BError sendEvent (BoapPacket &tx)
- virtual BError processEvent (int fd)
- virtual void clientGone (BoapServerConnection ∗client)
- BSocket & getSocket ()
- BSocket & getEventSocket ()
- BString getHostName ()
- int getConnectionsNumber ()
- virtual BoapServerConnection ∗ newConnection (int fd, BSocketAddressINET address)
- BoapServer ()
- BError init (int boapNs=0)
- BError run ()
- BError processEvent (BoapPacket &rx)
- BError addObject (BoapServiceObject ∗object)
- BError process (int fd)
- BError sendEvent (BoapPacket &tx)
- BSocket & getSocket ()
- BSocket & getEventSocket ()
- BError processEvent (int fd)
- BString getHostName ()

Public Attributes

- BUInt64 onumOperations

Private Member Functions

- void ∗ function ()

Private Attributes

- int othreaded
- int oisBoapns
- Boapns::Boapns ∗ oboapns
- BList< BoapServerConnection ∗ > oclients
- BEvent1Int oclientGoneEvent
- BList< BoapServiceEntry > oservices
- BPoll opoll
- BSocket onet
- BSocket onetEvent
- BSocketAddressINET onetEventAddress
- BString ohostName
- int oboapNs
- BoapPacket orx
- BoapPacket otx

7.58.1 Member Enumeration Documentation

7.58.1.1 anonymous enum

Enumerator

NOTHREADS

THREADED

7.58.2 Constructor & Destructor Documentation

7.58.2.1 BoapServer::BoapServer ( )

7.58.2.2 BoapServer::∼BoapServer ( ) [virtual]

7.58.2.3 BoapServer::BoapServer ( )

7.58.3 Member Function Documentation

7.58.3.1 BError BoapServer::addObject (BoapServiceObject ∗ *object*)

7.58.3.2 BError BoapServer::addObject (BoapServiceObject ∗ *object*) [virtual]

7.58.3.3 void BoapServer::clientGone (BoapServerConnection ∗ *client*) [virtual]

7.58.3.4 void ∗ BoapServer::function ( ) [private], [virtual]

Reimplemented from BThread.

7.58.3.5 int BoapServer::getConnectionsNumber ( )

7.58.3.6 BSocket& BoapServer::getEventSocket ( )

7.58.3.7 BSocket & BoapServer::getEventSocket ( )

7.58.3.8 BString BoapServer::getHostName ( )

7.58.3.9 BString BoapServer::getHostName ( )

7.58.3.10 BSocket& BoapServer::getSocket ( )

7.58.3.11 BSocket & BoapServer::getSocket ( )

7.58.3.12 BError BoapServer::init (int *boapNs* = 0)

7.58.3.13 BError BoapServer::init (BString *boapNsHost* = "", int *port* = 0, int *threaded* = 0, int *isBoapns* = 0) [virtual]

7.58.3.14 BoapServerConnection ∗ BoapServer::newConnection (int *fd*, BSocketAddressINET *address*) [virtual]

7.58.3.15 BError BoapServer::process (int *fd*)

7.58.3.16 BError BoapServer::process (BoapServerConnection ∗ *conn*, BoapPacket & *rx*, BoapPacket & *tx*) [virtual]

7.58.3.17 BError BoapServer::processEvent (BoapPacket & *rx*)

7.58.3.18 BError BoapServer::processEvent (int *fd*)

7.58.3.19 BError BoapServer::processEvent (BoapPacket & *rx*) [virtual]

7.58.3.20 BError BoapServer::processEvent (int *fd*) [virtual]

7.58.3.21 BError BoapServer::run ( )

7.58.3.22 BError BoapServer::run (int *inThread* = 0) [virtual]

7.58.3.23 BError BoapServer::sendEvent (BoapPacket & *tx*)

7.58.3.24 BError BoapServer::sendEvent (BoapPacket & *tx*) [virtual]

7.58.4 Member Data Documentation

7.58.4.1 int BoapServer::oboapNs [private]

7.58.4.2 Boapns::Boapns∗ BoapServer::oboapns [private]

7.58.4.3 BEvent1Int BoapServer::oclientGoneEvent [private]

7.58.4.4 BList<BoapServerConnection∗> BoapServer::oclients [private]

7.58.4.5 BString BoapServer::ohostName [private]

7.58.4.6 int BoapServer::oisBoapns [private]

7.58.4.7 BSocket BoapServer::onet [private]

7.58.4.8 BSocket BoapServer::onetEvent [private]

7.58.4.9 BSocketAddressINET BoapServer::onetEventAddress [private]

7.58.4.10 BUInt64 BoapServer::onumOperations

7.58.4.11 BPoll BoapServer::opoll [private]

7.58.4.12 BoapPacket BoapServer::orx [private]

7.58.4.13 BList< BoapServiceEntry > BoapServer::oservices [private]

7.58.4.14 int BoapServer::othreaded [private]

7.58.4.15 BoapPacket BoapServer::otx [private]

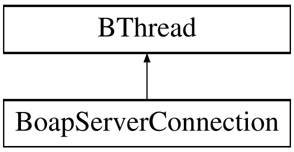
The documentation for this class was generated from the following files:

- /src/cern/tms/beam/libBeam/Boap.h
- /src/cern/tms/beam/libBeam/BoapSimple.h
- /src/cern/tms/beam/libBeam/Boap.cpp
- /src/cern/tms/beam/libBeam/BoapSimple.cc

7.59 BoapServerConnection Class Reference

```
#include <Boap.h>
```

Inheritance diagram for BoapServerConnection:



Public Member Functions

- BoapServerConnection (BoapServer &boapServer, int fd)
- virtual ∼BoapServerConnection ()
- virtual BError init ()

  Initialise connection.
- virtual BError process ()
- virtual BSocket & getSocket ()
- virtual void setMaxLength (BUInt32 maxLength)
- virtual BError getHead (BoapPacketHead &head)
- virtual BError validate ()

  Validate the connection.

Private Member Functions

- void ∗ function ()

Private Attributes

- BoapServer & oboapServer
- BSocket osocket
- BoapPacket orx
- BoapPacket otx
- BUInt32 omaxLength

7.59.1 Constructor & Destructor Documentation

7.59.1.1 BoapServerConnection::BoapServerConnection (BoapServer & *boapServer,* int *fd*)

7.59.1.2 BoapServerConnection::∼BoapServerConnection ( ) `[virtual]`

7.59.2 Member Function Documentation

7.59.2.1 void ∗ BoapServerConnection::function ( ) `[private], [virtual]`

Reimplemented from BThread.

7.59.2.2 BError BoapServerConnection::getHead (BoapPacketHead & *head*) [virtual]

7.59.2.3 BSocket & BoapServerConnection::getSocket ( ) [virtual]

7.59.2.4 BError BoapServerConnection::init ( ) [virtual]

Initialise connection.

7.59.2.5 BError BoapServerConnection::process ( ) [virtual]

7.59.2.6 void BoapServerConnection::setMaxLength (BUInt32 *maxLength*) [virtual]

7.59.2.7 BError BoapServerConnection::validate ( ) [virtual]

Validate the connection.

7.59.3 Member Data Documentation

7.59.3.1 BoapServer& BoapServerConnection::oboapServer [private]

7.59.3.2 BUInt32 BoapServerConnection::omaxLength [private]

7.59.3.3 BoapPacket BoapServerConnection::orx [private]

7.59.3.4 BSocket BoapServerConnection::osocket [private]

7.59.3.5 BoapPacket BoapServerConnection::otx [private]

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

- /src/cern/tms/beam/libBeam/Boap.h
- /src/cern/tms/beam/libBeam/Boap.cpp

7.60 BoapServiceEntry Class Reference

```
#include <BoapSimple.h>
```

Public Member Functions

- BoapServiceEntry (BoapService service=0, BoapServiceObject ∗object=0)
- BoapServiceEntry (BoapService service=0, BoapServiceObject ∗object=0)

Public Attributes

- BoapService oservice
- BoapServiceObject ∗ oobject

7.60.1 Constructor & Destructor Documentation

7.60.1.1 BoapServiceEntry::BoapServiceEntry (BoapService *service* = 0, BoapServiceObject ∗ *object* = 0) [inline]

7.60.1.2 BoapServiceEntry::BoapServiceEntry (BoapService *service* = 0, BoapServiceObject ∗ *object* = 0) [inline]

7.60.2 Member Data Documentation

7.60.2.1 BoapServiceObject ∗ BoapServiceEntry::oobject

7.60.2.2 BoapService BoapServiceEntry::oservice

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

- /src/cern/tms/beam/libBeam/Boap.h
- /src/cern/tms/beam/libBeam/BoapSimple.h

7.61 BoapServiceObject Class Reference

```
#include <BoapSimple.h>
```

Public Member Functions

- BoapServiceObject (BoapServer &server, BString name="")
- virtual ∼BoapServiceObject ()
- BError setName (BString name)
- BError sendEvent (BString signalName, BInt32 arg)
- virtual BError processEvent (BString objectName, BString name, BInt32 arg)
- BString name ()
- BError doPing (BoapServerConnection ∗conn, BoapPacket &rx, BoapPacket &tx)
- BError doConnectionPriority (BoapServerConnection ∗conn, BoapPacket &rx, BoapPacket &tx)
- BError process (BoapServerConnection ∗conn, BoapPacket &rx, BoapPacket &tx)
- virtual BError processEvent (BoapPacket &rx)
- BoapServiceObject (BoapServer &server, BString name)
- virtual ∼BoapServiceObject ()
- BError sendEvent (BString signalName, Int32 arg)
- virtual BError processEvent (BString objectName, BString name, Int32 arg)
- BString name ()
- BError process (BoapPacket &rx, BoapPacket &tx)
- virtual BError processEvent (BoapPacket &rx)

Protected Member Functions

- BError sendEvent (BoapPacket &tx)
- BError sendEvent (BoapPacket &tx)

Protected Attributes

- BoapServer & oserver
- BString oname
- BUInt32 oapiVersion
- BList< BoapFuncEntry > ofuncList

7.61.1 Constructor & Destructor Documentation

7.61.1.1 BoapServiceObject::BoapServiceObject (**BoapServer** & *server,* **BString** *name =* " ")

7.61.1.2 BoapServiceObject::∼BoapServiceObject ( ) [virtual]

7.61.1.3 BoapServiceObject::BoapServiceObject (**BoapServer** & *server,* **BString** *name*)

7.61.1.4 virtual BoapServiceObject::∼BoapServiceObject ( ) [virtual]

7.61.2 Member Function Documentation

7.61.2.1 **BError** BoapServiceObject::doConnectionPriority (**BoapServerConnection** ∗ *conn,* **BoapPacket** & *rx,* **BoapPacket** & *tx*)

7.61.2.2 **BError** BoapServiceObject::doPing (**BoapServerConnection** ∗ *conn,* **BoapPacket** & *rx,* **BoapPacket** & *tx*)

7.61.2.3 **BString** BoapServiceObject::name ( )

7.61.2.4 **BString** BoapServiceObject::name ( )

7.61.2.5 **BError** BoapServiceObject::process (**BoapPacket** & *rx,* **BoapPacket** & *tx*)

7.61.2.6 **BError** BoapServiceObject::process (**BoapServerConnection** ∗ *conn,* **BoapPacket** & *rx,* **BoapPacket** & *tx*)

7.61.2.7 virtual **BError** BoapServiceObject::processEvent (**BString** *objectName,* **BString** *name,* **Int32** *arg*) [virtual]

7.61.2.8 virtual **BError** BoapServiceObject::processEvent (**BoapPacket** & *rx*) [virtual]

7.61.2.9 **BError** BoapServiceObject::processEvent (**BString** *objectName,* **BString** *name,* **BInt32** *arg*) [virtual]

7.61.2.10 **BError** BoapServiceObject::processEvent (**BoapPacket** & *rx*) [virtual]

7.61.2.11 **BError** BoapServiceObject::sendEvent (**BString** *signalName,* **Int32** *arg*)

7.61.2.12 **BError** BoapServiceObject::sendEvent (**BoapPacket** & *tx*) [protected]

7.61.2.13 **BError** BoapServiceObject::sendEvent (**BString** *signalName,* **BInt32** *arg*)

7.61.2.14 **BError** BoapServiceObject::sendEvent (**BoapPacket** & *tx*) [protected]

7.61.2.15 **BError** BoapServiceObject::setName (**BString** *name*)

7.61.3 Member Data Documentation

7.61.3.1 **BUInt32** BoapServiceObject::oapiVersion [protected]

7.61.3.2 **BList**< **BoapFuncEntry** > BoapServiceObject::ofuncList [protected]

7.61.3.3 **BString** BoapServiceObject::oname [protected]

7.61.3.4 **BoapServer** & BoapServiceObject::oserver [protected]

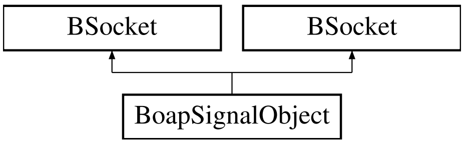
The documentation for this class was generated from the following files:

- /src/cern/tms/beam/libBeam/Boap.h
- /src/cern/tms/beam/libBeam/BoapSimple.h
- /src/cern/tms/beam/libBeam/Boap.cpp
- /src/cern/tms/beam/libBeam/BoapSimple.cc

7.62 BoapSignalObject Class Reference

```
#include <BoapSimple.h>
```

Inheritance diagram for BoapSignalObject:



Public Member Functions

- BoapSignalObject ()
- BoapSignalObject ()

Protected Member Functions

- BError performSend (BoapPacket &tx)
- BError performSend (BoapPacket &tx)

Protected Attributes

- BoapPacket otx
- BoapPacket orx

Additional Inherited Members

7.62.1 Constructor & Destructor Documentation

7.62.1.1 BoapSignalObject::BoapSignalObject ( )

7.62.1.2 BoapSignalObject::BoapSignalObject ( )

7.62.2 Member Function Documentation

7.62.2.1 BError BoapSignalObject::performSend (BoapPacket & *tx*) [protected]

7.62.2.2 BError BoapSignalObject::performSend (BoapPacket & *tx*) [protected]

7.62.3 Member Data Documentation

7.62.3.1 BoapPacket BoapSignalObject::orx [protected]

7.62.3.2 BoapPacket BoapSignalObject::otx `[protected]`

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

- /src/cern/tms/beam/libBeam/Boap.h
- /src/cern/tms/beam/libBeam/BoapSimple.h
- /src/cern/tms/beam/libBeam/Boap.cpp
- /src/cern/tms/beam/libBeam/BoapSimple.cc

7.63 BObj Class Reference

`#include <BObj.h>`

Public Member Functions

- BObj ()
- virtual ∼BObj ()
- virtual const char ∗ getType () const
- virtual const BObjMember ∗ getMembers () const
- virtual BError getMembers (BDictString &members)
- virtual BError getMember (BString name, BString &value)
- virtual BError setMembers (BDictString &members)
- virtual BError setMember (BString name, BString value)
- virtual void membersPrint () const

  Prints out members.
- virtual BString getDebugString ()

  Returns contents as a debug string.

7.63.1 Constructor & Destructor Documentation

7.63.1.1 BObj::BObj ( )

7.63.1.2 BObj::∼BObj ( ) `[virtual]`

7.63.2 Member Function Documentation

7.63.2.1 BString BObj::getDebugString ( ) `[virtual]`

Returns contents as a debug string.

7.63.2.2 BError BObj::getMember (BString *name,* BString & *value*) `[virtual]`

7.63.2.3 const BObjMember ∗ BObj::getMembers ( ) const `[virtual]`

7.63.2.4 BError BObj::getMembers (BDictString & *members*) `[virtual]`

7.63.2.5 const char ∗ BObj::getType ( ) const `[virtual]`

7.63.2.6 void BObj::membersPrint ( ) const `[virtual]`

Prints out members.

7.63.2.7 BError BObj::setMember (BString *name,* BString *value*) [virtual]

7.63.2.8 BError BObj::setMembers (BDictString & *members*) [virtual]

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

- /src/cern/tms/beam/libBeam/BObj.h
- /src/cern/tms/beam/libBeam/BObj.cpp

7.64 BObjMember Struct Reference

`#include <BTypes.h>`

Public Attributes

- BType type
- BTypeComp typeComp
- BUInt16 dataOffset
- BUInt16 size
- const char ∗ typeName
- const char ∗ name

7.64.1 Member Data Documentation

7.64.1.1 BUInt16 BObjMember::dataOffset

7.64.1.2 const char∗ BObjMember::name

7.64.1.3 BUInt16 BObjMember::size

7.64.1.4 BType BObjMember::type

7.64.1.5 BTypeComp BObjMember::typeComp

7.64.1.6 const char∗ BObjMember::typeName

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

- /src/cern/tms/beam/libBeam/BTypes.h

7.65 BPoll Class Reference

This class provides an interface for polling a number of file descriptors. It uses round robin polling.

`#include <BPoll.h>`

Public Types

- typedef struct pollfd PollFd

Public Member Functions

- BPoll ()
- ∼BPoll ()
- void append (int fd, int events=POLLIN|POLLERR|POLLHUP|POLLNVAL)

    Append a file descriptor to polling list.
- void delFd (int fd)

    Remove a file descriptor from polling list.
- BError doPoll (int &fd, int timeoutUs=-1)

    Perform polling operation.
- BError doPollEvents (int &fd, int &events, int timeoutUs=-1)

    Perform polling operation and return events.
- int getPollFdsNum ()
- PollFd ∗ getPollFds ()
- void clear ()

Private Member Functions

- int nextFd (int i)

Private Attributes

- int ofdsNum

    The number of FD's in list.
- PollFd ∗ ofds

    The list of poll fd's.
- int ofdsNext

    The next list entry for round robin polling.

7.65.1 Detailed Description

This class provides an interface for polling a number of file descriptors. It uses round robin polling.

7.65.2 Member Typedef Documentation

7.65.2.1 typedef struct pollfd **BPoll::PollFd**

7.65.3 Constructor & Destructor Documentation

7.65.3.1 BPoll::BPoll ( )

7.65.3.2 BPoll::∼BPoll ( )

7.65.4 Member Function Documentation

7.65.4.1 void BPoll::append (int *fd*, int *events* = `POLLIN|POLLERR|POLLHUP|POLLNVAL`)

Append a file descriptor to polling list.

7.65.4.2 void BPoll::clear ( )

7.65.4.3 void BPoll::delFd (int *fd*)

Remove a file descriptor from polling list.

7.65.4.4 BError BPoll::doPoll (int & *fd*, int *timeoutUs* = `-1`)

Perform polling operation.

7.65.4.5 BError BPoll::doPollEvents (int & *fd*, int & *events*, int *timeoutUs* = `-1`)

Perform polling operation and return events.

7.65.4.6 BPoll::PollFd ∗ BPoll::getPollFds ( )

7.65.4.7 int BPoll::getPollFdsNum ( )

7.65.4.8 int BPoll::nextFd (int *i*) `[private]`

7.65.5 Member Data Documentation

7.65.5.1 PollFd∗ BPoll::ofds `[private]`

The list of poll fd's.

7.65.5.2 int BPoll::ofdsNext `[private]`

The next list entry for round robin polling.

7.65.5.3 int BPoll::ofdsNum `[private]`

The number of FD's in list.

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

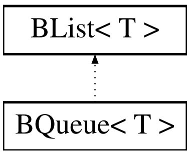
- /src/cern/tms/beam/libBeam/BPoll.h
- /src/cern/tms/beam/libBeam/BPoll.cpp

7.66 BQueue< T > Class Template Reference

Queue class.

`#include <BQueue.h>`

Inheritance diagram for BQueue< T >:



Public Member Functions

- BQueue (BUInt size)
- ∼BQueue ()
- void clear ()

  Clear the queue.
- BUInt writeAvailable () const
- BError write (const T &v, BTimeout timeout=BTimeoutForever)

  Append an item onto the queue.
- BUInt readAvailable () const
- BError read (T &v, BTimeout timeout=BTimeoutForever)

  Get an item from the queue.

Private Attributes

- BMutex olock
- BUInt osize
- BCondInt onumber

Additional Inherited Members

7.66.1 Detailed Description

template<class T>class BQueue< T >

Queue class.

7.66.2 Constructor & Destructor Documentation

7.66.2.1 template<class T > BQueue< T >::BQueue (BUInt *size*)

7.66.2.2 template<class T > BQueue< T >::∼BQueue ( )

7.66.3 Member Function Documentation

7.66.3.1 template<class T > void BQueue< T >::clear ( ) [virtual]

Clear the queue.

Reimplemented from BList< T >.

7.66.3.2 template<class T> BError BQueue< T >::read (T & *v*, BTimeout *timeout* = BTimeoutForever)

Get an item from the queue.

7.66.3.3 template<class T > BUInt BQueue< T >::readAvailable ( ) const

7.66.3.4 template<class T> BError BQueue< T >::write (const T & *v*, BTimeout *timeout* = BTimeoutForever)

Append an item onto the queue.

7.66.3.5 template< class T > BUInt BQueue< T >::writeAvailable ( ) const

7.66.4 Member Data Documentation

7.66.4.1 template<class T> BMutex BQueue< T >::olock [private]

7.66.4.2 template<class T> BCondInt BQueue< T >::onumber [private]

7.66.4.3 template<class T> BUInt BQueue< T >::osize [private]

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

- /src/cern/tms/beam/libBeam/BQueue.h

7.67 BRefData Class Reference

```
#include <BRefData.h>
```

Public Member Functions

- BRefData ()
- BRefData (int len)
- BRefData (const BRefData &refData)
- ∼BRefData ()
- BRefData ∗ copy ()

  Create a copy of this reference for writing, if necessary.
- BRefData ∗ addRef ()

  Increment the reference counter.
- int deleteRef ()

  Decrement the reference counter.
- char ∗ data ()

  Return the raw data pointer.
- int len ()

  Return the length in bytes.
- BRefData & operator= (const BRefData &refData)
- void setLen (int len)

  Set the length in bytes. Note should only be used if orefCount = 1.

Private Attributes

- BAtomicCount orefCount

  The reference count, how many users.
- int olen

  The actual length of data in oData.
- void ∗ odata

  Pointer to the data.

7.67.1 Detailed Description

Referenced data storage. This is Thread safe to a degree. The reference counting is protected. However, setLen() is not and should be protected at a higher level.

7.67.2 Constructor & Destructor Documentation

7.67.2.1 BRefData::BRefData ( )

7.67.2.2 BRefData::BRefData (int *len*)

7.67.2.3 BRefData::BRefData (const BRefData & *refData*)

7.67.2.4 BRefData::∼BRefData ( )

7.67.3 Member Function Documentation

7.67.3.1 BRefData ∗ BRefData::addRef ( )

Increment the reference counter.

7.67.3.2 BRefData ∗ BRefData::copy ( )

Create a copy of this reference for writing, if necessary.

7.67.3.3 char∗ BRefData::data ( ) [inline]

Return the raw data pointer.

7.67.3.4 int BRefData::deleteRef ( )

Decrement the reference counter.

7.67.3.5 int BRefData::len ( ) [inline]

Return the length in bytes.

7.67.3.6 BRefData & BRefData::operator= (const BRefData & *refData*)

7.67.3.7 void BRefData::setLen (int *len*)

Set the length in bytes. Note should only be used if orefCount = 1.

7.67.4 Member Data Documentation

7.67.4.1 void∗ BRefData::odata [private]

Pointer to the data.

7.67.4.2 int BRefData::olen [private]

The actual length of data in oData.

7.67.4.3 BAtomicCount BRefData::orefCount [private]

The reference count, how many users.

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

- /src/cern/tms/beam/libBeam/BRefData.h
- /src/cern/tms/beam/libBeam/BRefData.cpp

7.68 BRtc Class Reference

Realtime clock.

`#include <BRtc.h>`

Public Member Functions

- BRtc ()
- ∼BRtc ()
- BError init (int rate)

  Setup interrupt rate.
- void wait (int delayUs)

  Wait specified uS.

Private Attributes

- int ofd
- int orate

7.68.1 Detailed Description

Realtime clock.

7.68.2 Constructor & Destructor Documentation

7.68.2.1 BRtc::BRtc ( )

7.68.2.2 BRtc::∼BRtc ( )

7.68.3 Member Function Documentation

7.68.3.1 BError BRtc::init (int *rate*)

Setup interrupt rate.

7.68.3.2 void BRtc::wait (int *delayUs*)

Wait specified uS.

7.68.4 Member Data Documentation

7.68.4.1 int BRtc::ofd [private]

7.68.4.2 int BRtc::orate [private]

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

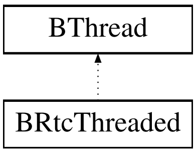
- /src/cern/tms/beam/libBeam/BRtc.h
- /src/cern/tms/beam/libBeam/BRtc.cpp

7.69 BRtcThreaded Class Reference

Threaded real time clock.

`#include <BRtc.h>`

Inheritance diagram for BRtcThreaded:



Public Member Functions

- BRtcThreaded ()
- ∼BRtcThreaded ()
- BError init (int rate)

  Setup interrupt rate.
- void wait (int delayUs)

  Wait specified uS.

Private Member Functions

- void ∗ function ()

Private Attributes

- BRtc ortc
- int orate
- BCond ocond

7.69.1 Detailed Description

Threaded real time clock.

7.69.2 Constructor & Destructor Documentation

7.69.2.1 BRtcThreaded::BRtcThreaded ( )

7.69.2.2 BRtcThreaded::∼BRtcThreaded ( )

7.69.3 Member Function Documentation

7.69.3.1 void ∗ BRtcThreaded::function ( ) [private], [virtual]

Reimplemented from BThread.

7.69.3.2 BError BRtcThreaded::init (int *rate*)

Setup interrupt rate.

7.69.3.3 void BRtcThreaded::wait (int *delayUs*)

Wait specified uS.

7.69.4 Member Data Documentation

7.69.4.1 BCond BRtcThreaded::ocond [private]

7.69.4.2 int BRtcThreaded::orate [private]

7.69.4.3 BRtc BRtcThreaded::ortc [private]

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

- /src/cern/tms/beam/libBeam/BRtc.h
- /src/cern/tms/beam/libBeam/BRtc.cpp

7.70 BRWLock Class Reference

thread read-write locks

`#include <BRWLock.h>`

Public Member Functions

- BRWLock ()
- BRWLock (const BRWLock &rwlock)
- ∼BRWLock ()
- int rdLock ()

  Set lock, wait if necessary.
- int tryRdLock ()

  Test the lock.
- int wrLock ()

  Set lock, wait if necessary.
- int tryWrLock ()

  Test the lock.

- int unlock ()

  Unlock the lock.
- BRWLock & operator= (const BRWLock &rwlock)

Private Attributes

- pthread_rwlock_t olock

7.70.1 Detailed Description

thread read-write locks

7.70.2 Constructor & Destructor Documentation

7.70.2.1 BRWLock::BRWLock ( )

7.70.2.2 BRWLock::BRWLock (const BRWLock & *rwlock*)

7.70.2.3 BRWLock::∼BRWLock ( )

7.70.3 Member Function Documentation

7.70.3.1 BRWLock & BRWLock::operator= (const BRWLock & *rwlock*)

7.70.3.2 int BRWLock::rdLock ( )

Set lock, wait if necessary.

7.70.3.3 int BRWLock::tryRdLock ( )

Test the lock.

7.70.3.4 int BRWLock::tryWrLock ( )

Test the lock.

7.70.3.5 int BRWLock::unlock ( )

Unlock the lock.

7.70.3.6 int BRWLock::wrLock ( )

Set lock, wait if necessary.

7.70.4 Member Data Documentation

7.70.4.1 pthread_rwlock_t BRWLock::olock `[private]`

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

- /src/cern/tms/beam/libBeam/BRWLock.h

- /src/cern/tms/beam/libBeam/BRWLock.cpp

7.71 BSema Class Reference

Sempahore class.

```
#include <BSema.h>
```

Public Member Functions

- BSema (int value=0)
- BSema (const BSema &sema)
- ∼BSema ()
- int post ()

  Post condition.
- int wait ()

  Wait for contition.
- int timedWait (int timeUs)

  Wait for condition with timeout.
- int tryWait ()

  Test for the condition.
- int getValue () const
- BSema & operator= (const BSema &sema)

Private Attributes

- sem_t osema

7.71.1 Detailed Description

Sempahore class.

7.71.2 Constructor & Destructor Documentation

7.71.2.1 BSema::BSema (int *value* = 0)

7.71.2.2 BSema::BSema (const BSema & *sema*)

7.71.2.3 BSema::∼BSema ( )

7.71.3 Member Function Documentation

7.71.3.1 int BSema::getValue ( ) const

7.71.3.2 BSema & BSema::operator= (const BSema & *sema*)

7.71.3.3 int BSema::post ( )

Post condition.

7.71.3.4 int BSema::timedWait (int *timeUs*)

Wait for condition with timeout.

7.71.3.5 int BSema::tryWait ( )

Test for the condition.

7.71.3.6 int BSema::wait ( )

Wait for contition.

7.71.4 Member Data Documentation

7.71.4.1 sem_t BSema::osema [private]

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

- /src/cern/tms/beam/libBeam/BSema.h
- /src/cern/tms/beam/libBeam/BSema.cpp

7.72 BSemaphore Class Reference

Semaphore class.

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

Public Member Functions

- BSemaphore ()
- BSemaphore (const BSemaphore &semaphore)
- ∼BSemaphore ()
- Bool wait (BTimeout timeoutUs=BTimeoutForever)

  Wait for the semaphore.
- void set ()

  Set the semaphore.
- int getValue () const
- BSemaphore & operator= (const BSemaphore &semaphore)

Private Attributes

- sem_t osema

7.72.1 Detailed Description

Semaphore class.

7.72.2 Constructor & Destructor Documentation

7.72.2.1 BSemaphore::BSemaphore ( )

7.72.2.2 BSemaphore::BSemaphore (const BSemaphore & *semaphore*)

7.72.2.3 BSemaphore::∼BSemaphore ( )

7.72.3 Member Function Documentation

7.72.3.1 int BSemaphore::getValue ( ) const

7.72.3.2 BSemaphore & BSemaphore::operator= (const BSemaphore & *semaphore*)

7.72.3.3 void BSemaphore::set ( )

Set the semaphore.

7.72.3.4 Bool BSemaphore::wait (BTimeout *timeoutUs* = BTimeoutForever)

Wait for the semaphore.

7.72.4 Member Data Documentation

7.72.4.1 sem_t BSemaphore::osema `[private]`

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

- /src/cern/tms/beam/libBeam/BSemaphore.h
- /src/cern/tms/beam/libBeam/BSemaphore.cpp

7.73 BSemaphoreCount Class Reference

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

Public Member Functions

- BSemaphoreCount ()
- BSemaphoreCount (const BSemaphoreCount &semaphore)
- ∼BSemaphoreCount ()
- void setValue (BUInt v)
- void add (int v=1)

  Set the semaphore.
- Bool wait (BUInt v=1, BTimeout timeoutUs=BTimeoutForever)

  Wait for the semaphore.
- Bool take (BUInt v=1, BTimeout timeoutUs=BTimeoutForever)

  Take for the semaphore.
- BUInt value ()
- BSemaphoreCount & operator= (const BSemaphoreCount &semaphore)

Private Attributes

- BMutex olock
- BSemaphore osema
- volatile BUInt ovalue

7.73.1 Constructor & Destructor Documentation

7.73.1.1 BSemaphoreCount::BSemaphoreCount ( )

7.73.1.2 BSemaphoreCount::BSemaphoreCount (const BSemaphoreCount & *semaphore*)

7.73.1.3 BSemaphoreCount::∼BSemaphoreCount ( )

7.73.2 Member Function Documentation

7.73.2.1 void BSemaphoreCount::add (int *v* = `1`)

Set the semaphore.

7.73.2.2 BSemaphoreCount & BSemaphoreCount::operator= (const BSemaphoreCount & *semaphore*)

7.73.2.3 void BSemaphoreCount::setValue (BUInt *v*)

7.73.2.4 Bool BSemaphoreCount::take (BUInt *v* = `1`, BTimeout *timeoutUs* = BTimeoutForever)

Take for the semaphore.

7.73.2.5 BUInt BSemaphoreCount::value ( )

7.73.2.6 Bool BSemaphoreCount::wait (BUInt *v* = `1`, BTimeout *timeoutUs* = BTimeoutForever)

Wait for the semaphore.

7.73.3 Member Data Documentation

7.73.3.1 BMutex BSemaphoreCount::olock `[private]`

7.73.3.2 BSemaphore BSemaphoreCount::osema `[private]`

7.73.3.3 volatile BUInt BSemaphoreCount::ovalue `[private]`

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

- /src/cern/tms/beam/libBeam/BSemaphore.h
- /src/cern/tms/beam/libBeam/BSemaphore.cpp

7.74 BSignal Class Reference

```
#include <SigGen.h>
```

Public Types

- enum { NumChannels = 9 }

Public Member Functions

- BSignal (int id=0, int numSamples=0, int numRepeat=0, int nextId=0)
- BSignal (const BSignal &sig)
- ∼BSignal ()
- BSignal & operator= (const BSignal &sig)

Public Attributes

- int id
- int numSamples
- int numRepeat
- int nextId
- Sample ∗ data [NumChannels]

7.74.1 Member Enumeration Documentation

7.74.1.1 anonymous enum

Enumerator

NumChannels

7.74.2 Constructor & Destructor Documentation

7.74.2.1 BSignal::BSignal (int *id* = 0, int *numSamples* = 0, int *numRepeat* = 0, int *nextId* = 0)

7.74.2.2 BSignal::BSignal (const BSignal & *sig*)

7.74.2.3 BSignal::∼BSignal ( )

7.74.3 Member Function Documentation

7.74.3.1 BSignal & BSignal::operator= (const BSignal & *sig*)

7.74.4 Member Data Documentation

7.74.4.1 Sample∗ BSignal::data[NumChannels]

7.74.4.2 int BSignal::id

7.74.4.3 int BSignal::nextId

7.74.4.4 int BSignal::numRepeat

7.74.4.5 int BSignal::numSamples

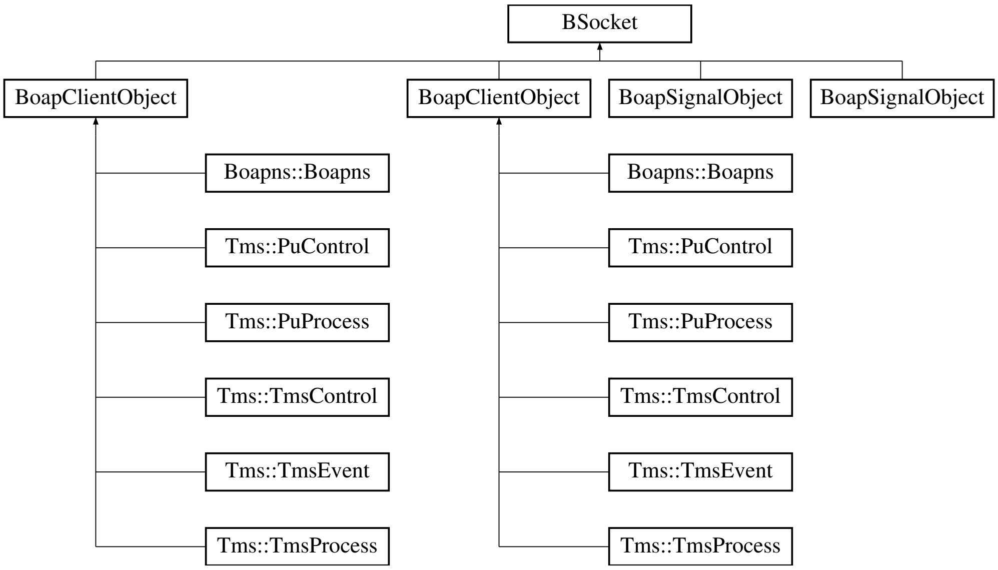
The documentation for this class was generated from the following files:

- SigGen.h
- SigGen.cpp

7.75 BSocket Class Reference

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

Inheritance diagram for BSocket:



Public Types

- enum NType { STREAM, DGRAM }
- enum Priority { PriorityLow, PriorityNormal, PriorityHigh }

Public Member Functions

- BSocket ()
- BSocket (int fd)
- BSocket (NType type)
- BSocket (int domain, int type, int protocol)
- ∼BSocket ()
- BError init (int domain, int type, int protocol)
- BError init (NType type)
- void setFd (int fd)
- int getFd ()
- BError bind (const BSocketAddress &add)
- BError connect (const BSocketAddress &add)
- BError shutdown (int how)
- BError close ()
- BError listen (int backlog=5)
- BError accept (int &fd)
- BError accept (int &fd, BSocketAddress &address)
- BError send (const void ∗buf, BSize nbytes, BSize &nbytesSent, int flags=0)
- BError sendTo (const BSocketAddress &address, const void ∗buf, BSize nbytes, BSize &nbytesSent, int flags=0)
- BError recv (void ∗buf, BSize maxbytes, BSize &nbytesRecv, int flags=0)
- BError recvFrom (BSocketAddress &address, void ∗buf, BSize maxbytes, BSize &nbytesRecv, int flags=0)
- BError recvWithTimeout (void ∗buf, BSize maxbytes, BSize &nbytesRecv, int timeout, int flags=0)

- BError recvFromWithTimeout (BSocketAddress &address, void ∗buf, BSize maxbytes, BSize &nbytesRecv, int timeout, int flags=0)
- BError setSockOpt (int level, int optname, void ∗optval, unsigned int optlen)
- BError getSockOpt (int level, int optname, void ∗optval, unsigned int ∗optlen)
- BError setReuseAddress (int on)
- BError setBroadCast (int on)
- BError setPriority (Priority priority)
- BError getMTU (uint32_t &mtu)
- BError getAddress (BSocketAddress &address)

Private Attributes

- int osocket

7.75.1 Member Enumeration Documentation

7.75.1.1 enum BSocket::NType

Enumerator

STREAM

DGRAM

7.75.1.2 enum BSocket::Priority

Enumerator

PriorityLow

PriorityNormal

PriorityHigh

7.75.2 Constructor & Destructor Documentation

7.75.2.1 BSocket::BSocket ( )

7.75.2.2 BSocket::BSocket (int *fd*)

7.75.2.3 BSocket::BSocket (NType *type*)

7.75.2.4 BSocket::BSocket (int *domain,* int *type,* int *protocol*)

7.75.2.5 BSocket::∼BSocket ( )

7.75.3 Member Function Documentation

7.75.3.1 BError BSocket::accept (int & *fd*)

7.75.3.2 BError BSocket::accept (int & *fd,* BSocketAddress & *address*)

7.75.3.3 BError BSocket::bind (const BSocketAddress & *add*)

7.75.3.4 BError BSocket::close ( )

7.75.3.5 BError BSocket::connect (const BSocketAddress & *add*)

7.75.3.6 BError BSocket::getAddress (BSocketAddress & *address*)

7.75.3.7 int BSocket::getFd ( )

7.75.3.8 BError BSocket::getMTU (uint32_t & *mtu*)

7.75.3.9 BError BSocket::getSockOpt (int *level,* int *optname,* void ∗ *optval,* unsigned int ∗ *optlen*)

7.75.3.10 BError BSocket::init (int *domain,* int *type,* int *protocol*)

7.75.3.11 BError BSocket::init (NType *type*)

7.75.3.12 BError BSocket::listen (int *backlog* = `5`)

7.75.3.13 BError BSocket::recv (void ∗ *buf,* BSize *maxbytes,* BSize & *nbytesRecv,* int *flags* = `0`)

7.75.3.14 BError BSocket::recvFrom (BSocketAddress & *address,* void ∗ *buf,* BSize *maxbytes,* BSize & *nbytesRecv,* int *flags* = `0`)

7.75.3.15 BError BSocket::recvFromWithTimeout (BSocketAddress & *address,* void ∗ *buf,* BSize *maxbytes,* BSize & *nbytesRecv,* int *timeout,* int *flags* = `0`)

7.75.3.16 BError BSocket::recvWithTimeout (void ∗ *buf,* BSize *maxbytes,* BSize & *nbytesRecv,* int *timeout,* int *flags* = `0`)

7.75.3.17 BError BSocket::send (const void ∗ *buf,* BSize *nbytes,* BSize & *nbytesSent,* int *flags* = `0`)

7.75.3.18 BError BSocket::sendTo (const BSocketAddress & *address,* const void ∗ *buf,* BSize *nbytes,* BSize & *nbytesSent,* int *flags* = `0`)

7.75.3.19 BError BSocket::setBroadCast (int *on*)

7.75.3.20 void BSocket::setFd (int *fd*)

7.75.3.21 BError BSocket::setPriority (Priority *priority*)

7.75.3.22 BError BSocket::setReuseAddress (int *on*)

7.75.3.23 BError BSocket::setSockOpt (int *level,* int *optname,* void ∗ *optval,* unsigned int *optlen*)

7.75.3.24 BError BSocket::shutdown (int *how*)

7.75.4 Member Data Documentation

7.75.4.1 int BSocket::osocket `[private]`

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

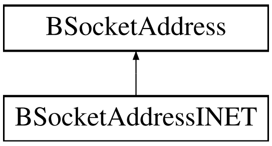
- /src/cern/tms/beam/libBeam/BSocket.h
- /src/cern/tms/beam/libBeam/BSocket.cpp

7.76 BSocketAddress Class Reference

Socket Address.

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

Inheritance diagram for BSocketAddress:



Public Types

- typedef struct sockaddr SockAddr

Public Member Functions

- BSocketAddress ()
- BSocketAddress (const BSocketAddress &add)
- BSocketAddress (SockAddr ∗address, int len)
- ∼BSocketAddress ()
- BError set (SockAddr ∗address, int len)
- const SockAddr ∗ raw () const
- int len () const
- BSocketAddress & operator= (const BSocketAddress &add)
- operator const SockAddr ∗ () const
- int operator== (const BSocketAddress &add) const
- int operator!= (const BSocketAddress &add) const

Private Attributes

- int olen
- SockAddr ∗ oaddress

7.76.1 Detailed Description

Socket Address.

7.76.2 Member Typedef Documentation

7.76.2.1 typedef struct sockaddr **BSocketAddress::SockAddr**

7.76.3 Constructor & Destructor Documentation

7.76.3.1 BSocketAddress::BSocketAddress ( )

7.76.3.2 BSocketAddress::BSocketAddress (const **BSocketAddress** & *add*)

7.76.3.3 BSocketAddress::BSocketAddress (**SockAddr** ∗ *address,* int *len*)

7.76.3.4 BSocketAddress::∼BSocketAddress ( )

7.76.4 Member Function Documentation

7.76.4.1 int BSocketAddress::len ( ) const

7.76.4.2 BSocketAddress::operator const SockAddr ∗ ( ) const [inline]

7.76.4.3 int BSocketAddress::operator!= (const BSocketAddress & *add*) const

7.76.4.4 BSocketAddress & BSocketAddress::operator= (const BSocketAddress & *add*)

7.76.4.5 int BSocketAddress::operator== (const BSocketAddress & *add*) const

7.76.4.6 const BSocketAddress::SockAddr ∗ BSocketAddress::raw ( ) const

7.76.4.7 BError BSocketAddress::set (SockAddr ∗ *address,* int *len*)

7.76.5 Member Data Documentation

7.76.5.1 SockAddr∗ BSocketAddress::oaddress [private]

7.76.5.2 int BSocketAddress::olen [private]

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

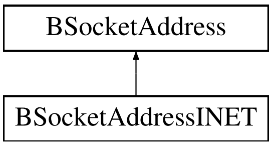
- /src/cern/tms/beam/libBeam/BSocket.h
- /src/cern/tms/beam/libBeam/BSocket.cpp

7.77 BSocketAddressINET Class Reference

IP aware socket address.

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

Inheritance diagram for BSocketAddressINET:



Public Types

- typedef struct sockaddr_in SockAddrIP

Public Member Functions

- BError set (BString hostName, uint32_t port)
- BError set (uint32_t address, uint32_t port)
- BError set (BString hostName, BString service, BString type)
- void setPort (uint32_t port)
- uint32_t address ()

  Returns socket ip address.
- uint32_t port ()

  Returns socket port.

- BString getString ()

  Return string version of address <ip>:<port>

Static Public Member Functions

- static BString getHostName ()

  Get this hosts network name.
- static BList< uint32_t > getIpAddresses ()

  Get a list of all the IP addresses of this host.
- static BList< BString > getIpAddressList ()

  Get a list of all the IP addresses of this host under hostname.
- static BList< BString > getIpAddressListAll ()

  Get a list of all the IP addresses of this host looking at physical interfaces.

7.77.1 Detailed Description

IP aware socket address.

7.77.2 Member Typedef Documentation

7.77.2.1 typedef struct sockaddr_in BSocketAddressINET::SockAddrIP

7.77.3 Member Function Documentation

7.77.3.1 uint32_t BSocketAddressINET::address ( )

Returns socket ip address.

7.77.3.2 BString BSocketAddressINET::getHostName ( ) [static]

Get this hosts network name.

7.77.3.3 BList< uint32_t > BSocketAddressINET::getIpAddresses ( ) [static]

Get a list of all the IP addresses of this host.

7.77.3.4 BList< BString > BSocketAddressINET::getIpAddressList ( ) [static]

Get a list of all the IP addresses of this host under hostname.

7.77.3.5 BList< BString > BSocketAddressINET::getIpAddressListAll ( ) [static]

Get a list of all the IP addresses of this host looking at physical interfaces.

7.77.3.6 BString BSocketAddressINET::getString ( )

Return string version of address <ip>:<port>

7.77.3.7 uint32_t BSocketAddressINET::port ( )

Returns socket port.

7.77.3.8 BError BSocketAddressINET::set (BString *hostName*, uint32_t *port*)

7.77.3.9 BError BSocketAddressINET::set (uint32_t *address*, uint32_t *port*)

7.77.3.10 BError BSocketAddressINET::set (BString *hostName*, BString *service*, BString *type*)

7.77.3.11 void BSocketAddressINET::setPort (uint32_t *port*)

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

- /src/cern/tms/beam/libBeam/BSocket.h
- /src/cern/tms/beam/libBeam/BSocket.cpp

7.78 BSpi Class Reference

BSpi class.

```
#include <BSpi.h>
```

Public Types

- enum Mode { Mode0 = 0, Mode1 = 1, Mode2 = 2, Mode3 = 3 }

Public Member Functions

- BSpi ()
- BError init (BString devName, BUInt speed=1000000, Mode mode=Mode1, Bool csActive=0)
- BError transact (BUInt8 dev, void ∗txBuf, int txLen, int pad, void ∗rxBuf, int rxLen)

Private Attributes

- BString odevName
- int odev

7.78.1 Detailed Description

BSpi class.

7.78.2 Member Enumeration Documentation

7.78.2.1 enum BSpi::Mode

Enumerator

Mode0

Mode1

Mode2

Mode3

7.78.3 Constructor & Destructor Documentation

7.78.3.1 BSpi::BSpi ( )

7.78.4 Member Function Documentation

7.78.4.1 BError BSpi::init (BString *devName*, BUInt *speed* = `1000000`, Mode *mode* = Mode1, Bool *csActive* = `0`)

7.78.4.2 BError BSpi::transact (BUInt8 *dev*, void ∗ *txBuf*, int *txLen*, int *pad*, void ∗ *rxBuf*, int *rxLen*)

7.78.5 Member Data Documentation

7.78.5.1 int BSpi::odev `[private]`

7.78.5.2 BString BSpi::odevName `[private]`

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

- /src/cern/tms/beam/libBeam/BSpi.h
- /src/cern/tms/beam/libBeam/BSpi.cpp

7.79 BString Class Reference

```
#include <BString.h>
```

Public Member Functions

- BString ()
- BString (const BString &string)
- BString (const char ∗str)
- BString (const char ∗str, unsigned int len)
- BString (char ch)
- BString (BInt v)
- BString (BUInt v)
- BString (BUInt64 v)
- BString (double v)
- ∼BString ()
- BString copy () const

  Return an independant copy.
- int len () const

  Length of string.
- const char ∗ retStr () const

  Ptr to char∗ representation.
- char ∗ retStrDup () const

  Ptr to newly malloc'd char∗.
- int retInt () const

  Return string as a int.
- unsigned int retUInt () const

  Return string as a int.
- double retDouble () const

  Return string as a double.
- int compare (const BString &string) const

Compare strings.

- int compareWild (const BString &string) const

  Compare string to string with wildcards.
- int compareWildExpression (const BString &string) const

  Compare string to space deliminated patterns.
- int compareRegex (const BString &pattern, int ignoreCase=0) const

  Compare strings.
- BString & truncate (int len)

  Truncate to length len.
- BString & pad (int len)

  Pad to length len.
- void clear ()

  Clear the string.
- BString & toUpper ()

  Convert to uppercase.
- BString & toLower ()

  Convert to lowercase.
- BString lowerFirst ()

  Return string with lowercase first character.
- void removeNL ()

  Remove if present NL from last char.
- BString justify (int leftMargin, int width)

  Justify the string to the given width.
- BString fixedLen (int length, int rightJustify=0)

  return string formated to fixed length
- BString firstLine ()

  Return first line.
- BString translateChar (char ch, BString replace=" ")

  Translate character converting them to the given string.
- BString reverse () const

  Reverse character order.
- BString subString (int start, int len) const

  Returns substring.
- int del (int start, int len)

  Delete substring.
- int insert (int start, BString str)

  Insert substring.
- int append (const BString &str)

  Append a string.
- BString add (const BString &str) const

  Add strings returning result.
- BString & printf (const char ∗fmt,...)

  Formated print into the string.
- int find (char ch) const

  Find ch in string searching forwards.
- int find (BString str) const

  Find string in string searching forwards.
- int findReverse (char ch) const

  Find ch in string searching backwards.
- BString csvEncode () const

  Encode a string for CSV.

- BString & csvDecode (const BString str)

    Decode a string from CSV.
- BString base64Encode () const

    Encode a string to base64.
- BError base64Decode (BString &str) const

    Decode a string from base64.
- BList< BString > getTokenList (BString separators)

    Break string into tokens.
- BList< BString > getTokenList (char separator)

    Break string into tokens.
- BString removeSeparators (BString separators)

    Remove any char from sepatators from string.
- BString pullToken (BString terminators)

    Pull token from start of string.
- BString pullSeparators (BString separators)

    Pull separators from start of string.
- BString pullWord ()

    Pull a word out of the head of the string.
- BString pullLine ()

    Pull a line out of the head of the string.
- BList< BString > split (char splitChar)

    Split string into an array based on the character separator.
- BString dirname ()
- BString basename ()
- BString extension ()
- BUInt32 hash () const
- char & get (int pos)
- const char & get (int pos) const
- BString & operator= (const BString &string)
- char & operator[] (int pos)
- int operator== (const BString &s) const
- int operator== (const char *s) const
- int operator> (const BString &s) const
- int operator> (const char *s) const
- int operator< (const BString &s) const
- int operator< (const char *s) const
- int operator>= (const BString &s) const
- int operator<= (const BString &s) const
- int operator!= (const BString &s) const
- int operator!= (const char *s) const
- BString operator+ (const BString &s) const
- BString operator+ (const char *s) const
- BString operator+= (const BString &s)
- BString operator+= (const char *s)
- BString operator+ (char ch) const
- BString operator+ (BInt i) const
- BString operator+ (BUInt i) const
- BString operator+ (BUInt64 i) const
- operator const char * () const
- BString field (int field) const
- char ** fields ()

Static Public Member Functions

- static BString convert (char ch)

  Converts char to string.
- static BString convert (BInt value)

  Converts int to string.
- static BString convert (BUInt value)

  Converts uint to string.
- static BString convert (double value, int eFormat=0)

  Converts double to string.
- static BString convert (BUInt64 value)

  Converts long long to string.
- static BString convertHex (BInt value)

  Converts int to string as hex value.
- static BString convertHex (BUInt value)

  Converts uint to string as hex value.

Protected Attributes

- BRefData ∗ ostr

Private Member Functions

- void init (const char ∗str)
- int inString (int pos) const
- int isSpace (char ch) const

7.79.1 Constructor & Destructor Documentation

7.79.1.1 BString::BString ( )

7.79.1.2 BString::BString (const **BString** & *string*)

7.79.1.3 BString::BString (const char ∗ *str*)

7.79.1.4 BString::BString (const char ∗ *str,* unsigned int *len*)

7.79.1.5 BString::BString (char *ch*)

7.79.1.6 BString::BString (**BInt** *v*)

7.79.1.7 BString::BString (**BUInt** *v*)

7.79.1.8 BString::BString (**BUInt64** *v*)

7.79.1.9 BString::BString (double *v*)

7.79.1.10 BString::∼BString ( )

7.79.2 Member Function Documentation

7.79.2.1 **BString** BString::add (const **BString** & *str*) const

Add strings returning result.

7.79.2.2 int BString::append (const BString & *str*)

Append a string.

7.79.2.3 BError BString::base64Decode (BString & *str*) const

Decode a string from base64.

7.79.2.4 BString BString::base64Encode ( ) const

Encode a string to base64.

7.79.2.5 BString BString::basename ( )

7.79.2.6 void BString::clear ( )

Clear the string.

7.79.2.7 int BString::compare (const BString & *string*) const

Compare strings.

7.79.2.8 int BString::compareRegex (const BString & *pattern*, int *ignoreCase* = 0) const

Compare strings.

7.79.2.9 int BString::compareWild (const BString & *string*) const

Compare string to string with wildcards.

7.79.2.10 int BString::compareWildExpression (const BString & *string*) const

Compare string to space deliminated patterns.

7.79.2.11 BString BString::convert (char *ch*) [static]

Converts char to string.

7.79.2.12 BString BString::convert (BInt *value*) [static]

Converts int to string.

7.79.2.13 BString BString::convert (BUInt *value*) [static]

Converts uint to string.

7.79.2.14 BString BString::convert (double *value*, int *eFormat* = 0) [static]

Converts double to string.

7.79.2.15 BString BString::convert (BUInt64 *value*) [static]

Converts long long to string.

7.79.2.16 BString BString::convertHex (BInt *value*) [static]

Converts int to string as hex value.

7.79.2.17 BString BString::convertHex (BUInt *value*) [static]

Converts uint to string as hex value.

7.79.2.18 BString BString::copy ( ) const

Return an independant copy.

7.79.2.19 BString & BString::csvDecode (const BString *str*)

Decode a string from CSV.

7.79.2.20 BString BString::csvEncode ( ) const

Encode a string for CSV.

7.79.2.21 int BString::del (int *start,* int *len*)

Delete substring.

7.79.2.22 BString BString::dirname ( )

7.79.2.23 BString BString::extension ( )

7.79.2.24 BString BString::field (int *field*) const

7.79.2.25 char ** BString::fields ( )

7.79.2.26 int BString::find (char *ch*) const

Find ch in string searching forwards.

7.79.2.27 int BString::find (BString *str*) const

Find string in string searching forwards.

7.79.2.28 int BString::findReverse (char *ch*) const

Find ch in string searching backwards.

7.79.2.29 BString BString::firstLine ( )

Return first line.

7.79.2.30 BString BString::fixedLen (int *length,* int *rightJustify =* 0)

return string formated to fixed length

7.79.2.31 char & BString::get (int *pos*)

7.79.2.32 const char & BString::get (int *pos*) const

7.79.2.33 BList< BString > BString::getTokenList (BString *separators*)

Break string into tokens.

7.79.2.34 BList< BString > BString::getTokenList (char *separator*)

Break string into tokens.

7.79.2.35 BUInt32 BString::hash ( ) const

7.79.2.36 void BString::init (const char ∗ *str*) [private]

7.79.2.37 int BString::insert (int *start,* BString *str*)

Insert substring.

7.79.2.38 int BString::inString (int *pos*) const [private]

7.79.2.39 int BString::isSpace (char *ch*) const [private]

7.79.2.40 BString BString::justify (int *leftMargin,* int *width*)

Justify the string to the given width.

7.79.2.41 int BString::len ( ) const

Length of string.

7.79.2.42 BString BString::lowerFirst ( )

Return string with lowercase first character.

7.79.2.43 BString::operator const char ∗ ( ) const [inline]

7.79.2.44 int BString::operator!= (const BString & *s*) const [inline]

7.79.2.45 int BString::operator!= (const char ∗ *s*) const [inline]

7.79.2.46 BString BString::operator+ (const BString & *s*) const [inline]

7.79.2.47 BString BString::operator+ (const char ∗ *s*) const [inline]

7.79.2.48 BString BString::operator+ (char *ch*) const [inline]

7.79.2.49 BString BString::operator+ (BInt *i*) const [inline]

7.79.2.50 BString BString::operator+ (BUInt *i*) const [inline]

7.79.2.51 BString BString::operator+ (BUInt64 *i*) const [inline]

7.79.2.52 BString BString::operator+= (const BString & *s*) [inline]

7.79.2.53 BString BString::operator+= (const char * *s*) [inline]

7.79.2.54 int BString::operator< (const BString & *s*) const [inline]

7.79.2.55 int BString::operator< (const char * *s*) const [inline]

7.79.2.56 int BString::operator<= (const BString & *s*) const [inline]

7.79.2.57 BString & BString::operator= (const BString & *string*)

7.79.2.58 int BString::operator== (const BString & *s*) const [inline]

7.79.2.59 int BString::operator== (const char * *s*) const [inline]

7.79.2.60 int BString::operator> (const BString & *s*) const [inline]

7.79.2.61 int BString::operator> (const char * *s*) const [inline]

7.79.2.62 int BString::operator>= (const BString & *s*) const [inline]

7.79.2.63 char & BString::operator[] (int *pos*)

7.79.2.64 BString & BString::pad (int *len*)

Pad to length len.

7.79.2.65 BString & BString::printf (const char * *fmt, ...*)

Formated print into the string.

7.79.2.66 BString BString::pullLine ( )

Pull a line out of the head of the string.

7.79.2.67 BString BString::pullSeparators (BString *separators*)

Pull separators from start of string.

7.79.2.68 BString BString::pullToken (BString *terminators*)

Pull token from start of string.

7.79.2.69 BString BString::pullWord ( )

Pull a word out of the head of the string.

7.79.2.70 void BString::removeNL ( )

Remove if present NL from last char.

7.79.2.71 BString BString::removeSeparators (BString *separators*)

Remove any char from sepatators from string.

7.79.2.72 double BString::retDouble ( ) const

Return string as a double.

7.79.2.73 int BString::retInt ( ) const

Return string as a int.

7.79.2.74 const char ∗ BString::retStr ( ) const

Ptr to char∗ representation.

7.79.2.75 char ∗ BString::retStrDup ( ) const

Ptr to newly malloc'd char∗.

7.79.2.76 unsigned int BString::retUInt ( ) const

Return string as a int.

7.79.2.77 BString BString::reverse ( ) const

Reverse character order.

7.79.2.78 BList< BString > BString::split (char *splitChar*)

Split string into an array based on the character separator.

7.79.2.79 BString BString::subString (int *start,* int *len*) const

Returns substring.

7.79.2.80 BString & BString::toLower ( )

Convert to lowercase.

7.79.2.81 BString & BString::toUpper ( )

Convert to uppercase.

7.79.2.82 BString BString::translateChar (char *ch*, BString *replace* = " ")

Translate character converting them to the given string.

7.79.2.83 BString & BString::truncate (int *len*)

Truncate to length len.

7.79.3 Member Data Documentation

7.79.3.1 BRefData∗ BString::ostr [protected]

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

- /src/cern/tms/beam/libBeam/BString.h
- /src/cern/tms/beam/libBeam/BString.cpp

7.80 BStringLocked Class Reference

```
#include <BStringLocked.h>
```

Public Member Functions

- BStringLocked ()
- BStringLocked (const BStringLocked &s)
- BStringLocked (const BString &s)
- int len () const

    Length of string.
- operator BString () const
- BStringLocked operator+ (const BStringLocked &s) const
- BStringLocked & operator= (const BStringLocked &s)

Private Attributes

- BStringMutex olock
- BString ostr

7.80.1 Constructor & Destructor Documentation

7.80.1.1 BStringLocked::BStringLocked ( ) [inline]

7.80.1.2 BStringLocked::BStringLocked (const BStringLocked & *s*) [inline]

7.80.1.3 BStringLocked::BStringLocked (const BString & *s*) [inline]

7.80.2 Member Function Documentation

7.80.2.1 int BStringLocked::len ( ) const [inline]

Length of string.

7.80.2.2 BStringLocked::operator BString ( ) const [inline]

7.80.2.3 BStringLocked BStringLocked::operator+ (const BStringLocked & *s*) const [inline]

7.80.2.4 BStringLocked& BStringLocked::operator= (const BStringLocked & *s*) [inline]

7.80.3 Member Data Documentation

7.80.3.1 BStringMutex BStringLocked::olock [mutable], [private]

7.80.3.2 BString BStringLocked::ostr [private]

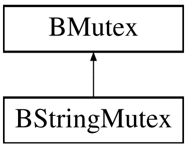
The documentation for this class was generated from the following file:

- /src/cern/tms/beam/libBeam/BStringLocked.h

7.81 BStringMutex Class Reference

`#include <BStringLocked.h>`

Inheritance diagram for BStringMutex:



Public Member Functions

- BStringMutex ()

Additional Inherited Members

7.81.1 Constructor & Destructor Documentation

7.81.1.1 BStringMutex::BStringMutex ( ) [inline]

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

- /src/cern/tms/beam/libBeam/BStringLocked.h

7.82 BTable Class Reference

`#include <BTable.h>`

Public Member Functions

- BTable ()
- ∼BTable ()
- void clear ()

- void setTitle (BArray< BString > title)
- void addRow (BArray< BString > data)
- void print ()

Private Member Functions

- void calculateWidths ()
- void printLine (BArray< BString > line, int comment=0)

Private Attributes

- BArray< BString > otitle
- BList< BArray< BString > > odata
- BArray< int > ocolumnWidths

7.82.1 Constructor & Destructor Documentation

7.82.1.1 BTable::BTable ( )

7.82.1.2 BTable::~BTable ( )

7.82.2 Member Function Documentation

7.82.2.1 void BTable::addRow (BArray< BString > *data*)

7.82.2.2 void BTable::calculateWidths ( ) [private]

7.82.2.3 void BTable::clear ( )

7.82.2.4 void BTable::print ( )

7.82.2.5 void BTable::printLine (BArray< BString > *line,* int *comment* = 0) [private]

7.82.2.6 void BTable::setTitle (BArray< BString > *title*)

7.82.3 Member Data Documentation

7.82.3.1 BArray<int> BTable::ocolumnWidths [private]

7.82.3.2 BList<BArray<BString> > BTable::odata [private]

7.82.3.3 BArray<BString> BTable::otitle [private]

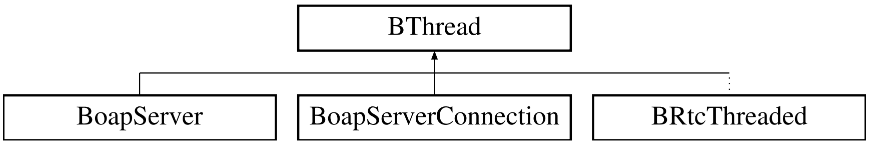
The documentation for this class was generated from the following files:

- /src/cern/tms/beam/libBeam/BTable.h
- /src/cern/tms/beam/libBeam/BTable.cpp

7.83 BThread Class Reference

```
#include <BThread.h>
```

Inheritance diagram for BThread:



Public Member Functions

- BThread ()
- virtual ∼BThread ()
- int setInitPriority (int policy, int priority)
- int setInitStackSize (size_t stackSize)
- int start ()
- void ∗ result ()
- int running ()
- int setPriority (int policy, int priority)
- int cancel ()
- void ∗ waitForCompletion ()
- pthread_t getThread ()
- virtual void ∗ function ()

Static Private Member Functions

- static void ∗ startFunc (void ∗)

Private Attributes

- pthread_t othread
- size_t ostackSize
- int opolicy
- int opriority
- int orunning
- void ∗ oresult

7.83.1 Constructor & Destructor Documentation

7.83.1.1 BThread::BThread ( )

7.83.1.2 BThread::∼BThread ( ) [virtual]

7.83.2 Member Function Documentation

7.83.2.1 int BThread::cancel ( )

7.83.2.2 void ∗ BThread::function ( ) [virtual]

Reimplemented in BoapServer, BoapServerConnection, and BRtcThreaded.

7.83.2.3 pthread_t BThread::getThread ( )

7.83.2.4 void ∗ BThread::result ( )

7.83.2.5 int BThread::running ( )

7.83.2.6 int BThread::setInitPriority (int *policy,* int *priority*)

7.83.2.7 int BThread::setInitStackSize (size_t *stackSize*)

7.83.2.8 int BThread::setPriority (int *policy,* int *priority*)

7.83.2.9 int BThread::start ( )

7.83.2.10 void ∗ BThread::startFunc (void ∗ *arg*) [static], [private]

7.83.2.11 void ∗ BThread::waitForCompletion ( )

7.83.3 Member Data Documentation

7.83.3.1 int BThread::opolicy [private]

7.83.3.2 int BThread::opriority [private]

7.83.3.3 void∗ BThread::oresult [private]

7.83.3.4 int BThread::orunning [private]

7.83.3.5 size_t BThread::ostackSize [private]

7.83.3.6 pthread_t BThread::othread [private]

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

- /src/cern/tms/beam/libBeam/BThread.h
- /src/cern/tms/beam/libBeam/BThread.cpp

7.84 BTime Class Reference

`#include <BTime.h>`

Public Member Functions

- BTime (BUInt32 t=0)
- void set (BUInt32 seconds)

  Set the date and time.
- void set (BUInt year, BUInt month, BUInt day, BUInt hour=0, BUInt minute=0, BUInt second=0)

  Set the date and time.
- void setYearDay (BUInt year, BUInt yearDay, BUInt hour=0, BUInt minute=0, BUInt second=0)

  Set the date and time.
- void getDate (BUInt &year, BUInt &month, BUInt &day) const

  Retun the date information.
- void getTime (BUInt &hour, BUInt &minute, BUInt &second) const

Return the time information.

- BUInt32 getSeconds () const

  Return the number of seconds.
- int isSet () const

  Check if set.
- int isLeapYear ()

  Returns if a leap year.
- void addSeconds (int seconds)

  Add the given number of seconds.
- BString getString (BString format="isoT") const

  Gets the date/time in string format.
- BError setString (const BString dateTime)

  Sets the date/time from string format.
- int operator== (const BTime &time) const
- int operator!= (const BTime &time) const
- int operator> (const BTime &time) const
- int operator>= (const BTime &time) const
- int operator< (const BTime &time) const
- int operator<= (const BTime &time) const
- BTime operator+ (int seconds) const
- BTime & operator+= (int seconds)

Private Attributes

- BUInt32 otime

  Time in seconds since 1970. range 1970-01-02 to 2106-02-07.

7.84.1 Constructor & Destructor Documentation

7.84.1.1 BTime::BTime (BUInt32 *t* = `0`)

7.84.2 Member Function Documentation

7.84.2.1 void BTime::addSeconds (int *seconds*)

Add the given number of seconds.

7.84.2.2 void BTime::getDate (BUInt & *year,* BUInt & *month,* BUInt & *day*) const

Retun the date information.

7.84.2.3 BUInt32 BTime::getSeconds ( ) const

Return the number of seconds.

7.84.2.4 BString BTime::getString (BString *format* = `"isoT"`) const

Gets the date/time in string format.

7.84.2.5 void BTime::getTime (BUInt & *hour,* BUInt & *minute,* BUInt & *second*) const

Return the time information.

7.84.2.6 int BTime::isLeapYear ( )

Returns if a leap year.

7.84.2.7 int BTime::isSet ( ) const [inline]

Check if set.

7.84.2.8 int BTime::operator!= (const BTime & *time*) const [inline]

7.84.2.9 BTime BTime::operator+ (int *seconds*) const [inline]

7.84.2.10 BTime& BTime::operator+= (int *seconds*) [inline]

7.84.2.11 int BTime::operator< (const BTime & *time*) const [inline]

7.84.2.12 int BTime::operator<= (const BTime & *time*) const [inline]

7.84.2.13 int BTime::operator== (const BTime & *time*) const [inline]

7.84.2.14 int BTime::operator> (const BTime & *time*) const [inline]

7.84.2.15 int BTime::operator>= (const BTime & *time*) const [inline]

7.84.2.16 void BTime::set (BUInt32 *seconds*)

Set the date and time.

7.84.2.17 void BTime::set (BUInt *year,* BUInt *month,* BUInt *day,* BUInt *hour* = 0, BUInt *minute* = 0, BUInt *second* = 0)

Set the date and time.

7.84.2.18 BError BTime::setString (const BString *dateTime*)

Sets the date/time from string format.

7.84.2.19 void BTime::setYearDay (BUInt *year,* BUInt *yearDay,* BUInt *hour* = 0, BUInt *minute* = 0, BUInt *second* = 0)

Set the date and time.

7.84.3 Member Data Documentation

7.84.3.1 BUInt32 BTime::otime [private]

Time in seconds since 1970. range 1970-01-02 to 2106-02-07.

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

- /src/cern/tms/beam/libBeam/BTime.h

- /src/cern/tms/beam/libBeam/BTime.cpp

7.85 BTimer Class Reference

Stopwatch style timer.

```
#include <BTimer.h>
```

Public Member Functions

- BTimer ()
- ∼BTimer ()
- void start ()

  Start timer.
- void stop ()

  Stop timer.
- void clear ()

  Clear timer.
- double getElapsedTime ()

  Returns the elapsed time from the last start.
- void add (BTimer &timer)

  Add two timers.
- double average ()

  Average time is duration between start() and stop() / number of stops.
- double peak ()

  Peak time.

Static Private Member Functions

- static double getTime ()

Private Attributes

- BMutex olock
- unsigned int onum
- double ostartTime
- double oendTime
- double oaverage
- double opeak

7.85.1 Detailed Description

Stopwatch style timer.

7.85.2 Constructor & Destructor Documentation

7.85.2.1 BTimer::BTimer ( )

7.85.2.2 BTimer::∼BTimer ( )

7.85.3 Member Function Documentation

7.85.3.1 void BTimer::add (BTimer & *timer*)

Add two timers.

7.85.3.2 double BTimer::average ( )

Average time is duration between start() and stop() / number of stops.

7.85.3.3 void BTimer::clear ( )

Clear timer.

7.85.3.4 double BTimer::getElapsedTime ( )

Returns the elapsed time from the last start.

7.85.3.5 double BTimer::getTime ( ) [static], [private]

7.85.3.6 double BTimer::peak ( )

Peak time.

7.85.3.7 void BTimer::start ( )

Start timer.

7.85.3.8 void BTimer::stop ( )

Stop timer.

7.85.4 Member Data Documentation

7.85.4.1 double BTimer::oaverage [private]

7.85.4.2 double BTimer::oendTime [private]

7.85.4.3 BMutex BTimer::olock [private]

7.85.4.4 unsigned int BTimer::onum [private]

7.85.4.5 double BTimer::opeak [private]

7.85.4.6 double BTimer::ostartTime [private]

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

- /src/cern/tms/beam/libBeam/BTimer.h
- /src/cern/tms/beam/libBeam/BTimer.cpp

7.86 BTimeStamp Class Reference

```
#include <BTimeStamp.h>
```

Public Member Functions

- BTimeStamp ()
- BTimeStamp (int year, int month=1, int day=1, int hour=0, int minute=0, int second=0, int microsecond=0)
- BTimeStamp (const BString str)
- ∼BTimeStamp ()
- void clear ()

  Clear the date/time.
- void setFirst ()

  Set the first date available.
- void setLast ()

  Set the last date available.
- void set (time_t time, int microSeconds)

  Set time using Unix time (seconds from 1970-01-01)
- void set (int year=0, int month=1, int day=1, int hour=0, int minute=0, int second=0, int microsecond=0)
- void set (const BTimeStampMs &timeStamp)

  Set the timeStamp to given MS time stamp.
- void setYDay (int year=0, int yday=0, int hour=0, int minute=0, int second=0, int microsecond=0)
- void setTime (int hour=0, int minute=0, int second=0, int microsecond=0)
- void setNow ()

  Set the timeStamp to now.
- int year () const
- int yday () const
- int month () const
- int day () const
- int hour () const
- int minute () const
- int second () const
- int microSecond () const
- void getDate (int &year, int &mon, int &day) const
- BString getString (BString separator="T") const

  Get the time as an ISO date/time string.
- BError setString (const BString dateTime)

  Set the time from an ISO date/time.
- BString getStringNoMs (BString separator="T") const

  Get the time as an ISO date/time string without microseconds.
- BString getStringFormatted (BString format) const

  Gets the time in a string form as per the format. Format syntax as per strftime()
- void addMilliSeconds (int milliSeconds)

  Add the given number of milli seconds. This should be less that a year.

- void addMicroSeconds (int64_t microSeconds)

  Add the given number of micro seconds. This should be less that a year.
- void addSeconds (int seconds)

  Add the given number of seconds. This should be less that a year.
- uint32_t getYearSeconds () const

  Get number of seconds within the year.
- uint64_t getYearMicroSeconds () const

  Get number of micro seconds within the year.
- int isSet () const
- int compare (const BTimeStamp &timeStamp) const

  Compare two dates.
- operator BString () const
- BTimeStamp & operator= (const BTimeStampMs &timeStamp)
- int operator== (const BTimeStamp &timeStamp) const
- int operator!= (const BTimeStamp &timeStamp) const
- int operator> (const BTimeStamp &timeStamp) const
- int operator>= (const BTimeStamp &timeStamp) const
- int operator< (const BTimeStamp &timeStamp) const
- int operator<= (const BTimeStamp &timeStamp) const

Static Public Member Functions

- static int isLeap (int year)
- static BInt64 difference (BTimeStamp t2, BTimeStamp t1)

Public Attributes

- uint16_t oyear

  Year (0 .. 65535)
- uint16_t oyday

  Day in year (0 .. 365)
- uint8_t ohour

  Hour (0 .. 23)
- uint8_t ominute

  Minute (0 .. 59)
- uint8_t osecond

  Second (0 .. 59)
- uint8_t ospare

  Padding.
- uint32_t omicroSecond

  MicroSecond (0 .. 999999)

7.86.1 Constructor & Destructor Documentation

7.86.1.1 BTimeStamp::BTimeStamp ( )

7.86.1.2 BTimeStamp::BTimeStamp (int *year*, int *month* = 1, int *day* = 1, int *hour* = 0, int *minute* = 0, int *second* = 0, int *microsecond* = 0)

7.86.1.3 BTimeStamp::BTimeStamp (const BString *str*)

7.86.1.4 BTimeStamp::∼BTimeStamp ( )

7.86.2 Member Function Documentation

7.86.2.1 void BTimeStamp::addMicroSeconds (int64_t *microSeconds*)

Add the given number of micro seconds. This should be less that a year.

7.86.2.2 void BTimeStamp::addMilliSeconds (int *milliSeconds*)

Add the given number of milli seconds. This should be less that a year.

7.86.2.3 void BTimeStamp::addSeconds (int *seconds*)

Add the given number of seconds. This should be less that a year.

7.86.2.4 void BTimeStamp::clear ( )

Clear the date/time.

7.86.2.5 int BTimeStamp::compare (const BTimeStamp & *timeStamp*) const

Compare two dates.

7.86.2.6 int BTimeStamp::day ( ) const

7.86.2.7 BInt64 BTimeStamp::difference (BTimeStamp *t2,* BTimeStamp *t1*) [static]

7.86.2.8 void BTimeStamp::getDate (int & *year,* int & *mon,* int & *day*) const

7.86.2.9 BString BTimeStamp::getString (BString *separator* = "T") const

Get the time as an ISO date/time string.

7.86.2.10 BString BTimeStamp::getStringFormatted (BString *format*) const

Gets the time in a string form as per the format. Format syntax as per strftime()

7.86.2.11 BString BTimeStamp::getStringNoMs (BString *separator* = "T") const

Get the time as an ISO date/time string without microseconds.

7.86.2.12 uint64_t BTimeStamp::getYearMicroSeconds ( ) const

Get number of micro seconds within the year.

7.86.2.13 uint32_t BTimeStamp::getYearSeconds ( ) const

Get number of seconds within the year.

7.86.2.14 int BTimeStamp::hour ( ) const

7.86.2.15 int BTimeStamp::isLeap (int *year*) [static]

7.86.2.16 int BTimeStamp::isSet ( ) const [inline]

7.86.2.17 int BTimeStamp::microSecond ( ) const

7.86.2.18 int BTimeStamp::minute ( ) const

7.86.2.19 int BTimeStamp::month ( ) const

7.86.2.20 BTimeStamp::operator BString ( ) const [inline]

7.86.2.21 int BTimeStamp::operator!= (const BTimeStamp & *timeStamp*) const [inline]

7.86.2.22 int BTimeStamp::operator< (const BTimeStamp & *timeStamp*) const [inline]

7.86.2.23 int BTimeStamp::operator<= (const BTimeStamp & *timeStamp*) const [inline]

7.86.2.24 BTimeStamp& BTimeStamp::operator= (const BTimeStampMs & *timeStamp*) [inline]

7.86.2.25 int BTimeStamp::operator== (const BTimeStamp & *timeStamp*) const [inline]

7.86.2.26 int BTimeStamp::operator> (const BTimeStamp & *timeStamp*) const [inline]

7.86.2.27 int BTimeStamp::operator>= (const BTimeStamp & *timeStamp*) const [inline]

7.86.2.28 int BTimeStamp::second ( ) const

7.86.2.29 void BTimeStamp::set (time_t *time*, int *microSeconds*)

Set time using Unix time (seconds from 1970-01-01)

7.86.2.30 void BTimeStamp::set (int *year* = 0, int *month* = 1, int *day* = 1, int *hour* = 0, int *minute* = 0, int *second* = 0, int *microsecond* = 0)

7.86.2.31 void BTimeStamp::set (const BTimeStampMs & *timeStamp*)

Set the timeStamp to given MS time stamp.

7.86.2.32 void BTimeStamp::setFirst ( )

Set the first date available.

7.86.2.33 void BTimeStamp::setLast ( )

Set the last date available.

7.86.2.34 void BTimeStamp::setNow ( )

Set the timeStamp to now.

7.86.2.35 BError BTimeStamp::setString (const BString *dateTime*)

Set the time from an ISO date/time.

7.86.2.36 void BTimeStamp::setTime (int *hour* = 0, int *minute* = 0, int *second* = 0, int *microsecond* = 0)

7.86.2.37 void BTimeStamp::setYDay (int *year* = 0, int *yday* = 0, int *hour* = 0, int *minute* = 0, int *second* = 0, int *microsecond* = 0)

7.86.2.38 int BTimeStamp::yday ( ) const

7.86.2.39 int BTimeStamp::year ( ) const

7.86.3 Member Data Documentation

7.86.3.1 uint8_t BTimeStamp::ohour

Hour (0 .. 23)

7.86.3.2 uint32_t BTimeStamp::omicroSecond

MicroSecond (0 .. 999999)

7.86.3.3 uint8_t BTimeStamp::ominute

Minute (0 .. 59)

7.86.3.4 uint8_t BTimeStamp::osecond

Second (0 .. 59)

7.86.3.5 uint8_t BTimeStamp::ospare

Padding.

7.86.3.6 uint16_t BTimeStamp::oyday

Day in year (0 .. 365)

7.86.3.7 uint16_t BTimeStamp::oyear

Year (0 .. 65535)

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

- /src/cern/tms/beam/libBeam/BTimeStamp.h
- /src/cern/tms/beam/libBeam/BTimeStamp.cpp

7.87 BTimeStampMs Class Reference

```
#include <BTimeStampMs.h>
```

Public Member Functions

- BTimeStampMs (BString str="")
- ∼BTimeStampMs ()
- void clear ()

  Clear the date/time.
- void setNow ()

  Set the timeStamp to now.
- BTimeStampMs & addMilliSeconds (int milliSeconds)

  Add the given number of milli seconds. This should be less that a year.
- BTimeStampMs & subMilliSeconds (int milliSeconds)

  Add the given number of milli seconds. This should be less that a year.
- BTimeStampMs & addSeconds (int seconds)

  Add the given number of seconds. This should be less that a year.
- BTimeStampMs & subSeconds (int seconds)

  Subtract the given number of seconds. This should be less that a year.
- uint32_t getYearSeconds ()

  Get number of seconds within the year.
- uint64_t getYearMilliSeconds ()

  Get number of seconds within the year.
- BString getString (BString separator="T")

  Get the time as an ISO date/time string.
- BString getStringNoMs (BString separator="T")

  Get the time as an ISO date/time string with no ms.
- BError setString (BString dateTime)

  Set the time from an ISO date/time.
- BString getDurationString (BString separator="T")

  Get the time as an ISO date/time string but with month's and days starting from 0.
- BString getDurationStringNoMs (BString separator="T")

  Get the time as an ISO date/time string but with month's and days starting from 0 with no ms.
- BError setDurationString (BString dateTime)

  Set the time from an ISO date/time string but with month's and days starting from 0.
- BString getStringRaw ()
- void getDate (int &year, int &mon, int &day)

  Get the year, month and day.
- int compare (const BTimeStampMs &timeStamp)

  Compare two dates.
- int operator> (const BTimeStampMs &timeStamp)
- int operator>= (const BTimeStampMs &timeStamp)
- int operator< (const BTimeStampMs &timeStamp)
- int operator<= (const BTimeStampMs &timeStamp)

Static Public Member Functions

- static int isLeap (int year)
- static BUInt64 difference (BTimeStampMs t2, BTimeStampMs t1)

Public Attributes

- uint16_t year

  Year (2000 .. 3000)
- uint16_t yday

  Day in year (0 .. 365)
- uint16_t hour

  Hour (0 .. 23)
- uint16_t minute

  Minute (0 .. 59)
- uint16_t second

  Second (0 .. 59)
- uint16_t milliSecond

  MilliSecond (0 .. 999)
- int32_t sampleNumber

  The sample number this time refers to.

7.87.1 Constructor & Destructor Documentation

7.87.1.1 BTimeStampMs::BTimeStampMs (**BString** *str* = "")

7.87.1.2 BTimeStampMs::∼BTimeStampMs ( )

7.87.2 Member Function Documentation

7.87.2.1 **BTimeStampMs** & BTimeStampMs::addMilliSeconds (int *milliSeconds*)

Add the given number of milli seconds. This should be less that a year.

7.87.2.2 **BTimeStampMs** & BTimeStampMs::addSeconds (int *seconds*)

Add the given number of seconds. This should be less that a year.

7.87.2.3 void BTimeStampMs::clear ( )

Clear the date/time.

7.87.2.4 int BTimeStampMs::compare (const **BTimeStampMs** & *timeStamp*)

Compare two dates.

7.87.2.5 **BUInt64** BTimeStampMs::difference (**BTimeStampMs** *t2,* **BTimeStampMs** *t1*) [static]

7.87.2.6 void BTimeStampMs::getDate (int & *year,* int & *mon,* int & *day*)

Get the year, month and day.

7.87.2.7 **BString** BTimeStampMs::getDurationString (**BString** *separator* = "T")

Get the time as an ISO date/time string but with month's and days starting from 0.

7.87.2.8 BString BTimeStampMs::getDurationStringNoMs (BString *separator* = "T")

Get the time as an ISO date/time string but with month's and days starting from 0 with no ms.

7.87.2.9 BString BTimeStampMs::getString (BString *separator* = "T")

Get the time as an ISO date/time string.

7.87.2.10 BString BTimeStampMs::getStringNoMs (BString *separator* = "T")

Get the time as an ISO date/time string with no ms.

7.87.2.11 BString BTimeStampMs::getStringRaw ( )

7.87.2.12 uint64_t BTimeStampMs::getYearMilliSeconds ( )

Get number of seconds within the year.

7.87.2.13 uint32_t BTimeStampMs::getYearSeconds ( )

Get number of seconds within the year.

7.87.2.14 int BTimeStampMs::isLeap (int *year*) [static]

7.87.2.15 int BTimeStampMs::operator< (const BTimeStampMs & *timeStamp*) [inline]

7.87.2.16 int BTimeStampMs::operator<= (const BTimeStampMs & *timeStamp*) [inline]

7.87.2.17 int BTimeStampMs::operator> (const BTimeStampMs & *timeStamp*) [inline]

7.87.2.18 int BTimeStampMs::operator>= (const BTimeStampMs & *timeStamp*) [inline]

7.87.2.19 BError BTimeStampMs::setDurationString (BString *dateTime*)

Set the time from an ISO date/time string but with month's and days starting from 0.

7.87.2.20 void BTimeStampMs::setNow ( )

Set the timeStamp to now.

7.87.2.21 BError BTimeStampMs::setString (BString *dateTime*)

Set the time from an ISO date/time.

7.87.2.22 BTimeStampMs & BTimeStampMs::subMilliSeconds (int *milliSeconds*)

Add the given number of milli seconds. This should be less that a year.

7.87.2.23 BTimeStampMs & BTimeStampMs::subSeconds (int *seconds*)

Subtract the given number of seconds. This should be less that a year.

7.87.3 Member Data Documentation

7.87.3.1 uint16_t BTimeStampMs::hour

Hour (0 .. 23)

7.87.3.2 uint16_t BTimeStampMs::milliSecond

MilliSecond (0 .. 999)

7.87.3.3 uint16_t BTimeStampMs::minute

Minute (0 .. 59)

7.87.3.4 int32_t BTimeStampMs::sampleNumber

The sample number this time refers to.

7.87.3.5 uint16_t BTimeStampMs::second

Second (0 .. 59)

7.87.3.6 uint16_t BTimeStampMs::yday

Day in year (0 .. 365)

7.87.3.7 uint16_t BTimeStampMs::year

Year (2000 .. 3000)

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

- /src/cern/tms/beam/libBeam/BTimeStampMs.h
- /src/cern/tms/beam/libBeam/BTimeStampMs.cpp

7.88 BUrl Class Reference

Basic access to a Url.

`#include <BUrl.h>`

Public Member Functions

- BUrl ()
- ∼BUrl ()
- BError readString (BString url, BString &str)

  Reads URL.

Static Private Member Functions

- static size_t writeData (void ∗data, size_t size, size_t elSize, void ∗stream)

Private Attributes

- BString ores

Static Private Attributes

- static int oinit

7.88.1 Detailed Description

Basic access to a Url.

7.88.2 Constructor & Destructor Documentation

7.88.2.1 BUrl::BUrl ( )

7.88.2.2 BUrl::∼BUrl ( )

7.88.3 Member Function Documentation

7.88.3.1 BError BUrl::readString (BString *url*, BString & *str*)

Reads URL.

7.88.3.2 size_t BUrl::writeData (void ∗ *data*, size_t *size*, size_t *elSize*, void ∗ *stream*) `[static]`, `[private]`

7.88.4 Member Data Documentation

7.88.4.1 int BUrl::oinit `[static]`, `[private]`

7.88.4.2 BString BUrl::ores `[private]`

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

- /src/cern/tms/beam/libBeam/BUrl.h
- /src/cern/tms/beam/libBeam/BUrl.cpp

7.89 Tms::ConfigInfo Class Reference

This class describes the configuration of the TMS.

```
#include <TmsD.h>
```

Public Member Functions

- ConfigInfo (BArray< PuChannel > ppuReferences=BArray< PuChannel >())

Public Attributes

- BArray< PuChannel > puReferences

  The logical to physical Pick-Up table. Each PuReference includes a Module Controller identifier, a Physical Pick-Up number and a Physical Channel.

7.89.1 Detailed Description

This class describes the configuration of the TMS.

7.89.2 Constructor & Destructor Documentation

7.89.2.1 Tms::ConfigInfo::ConfigInfo (BArray< PuChannel > *ppuReferences* = BArray<PuChannel >())

7.89.3 Member Data Documentation

7.89.3.1 BArray<PuChannel > Tms::ConfigInfo::puReferences

The logical to physical Pick-Up table. Each PuReference includes a Module Controller identifier, a Physical Pick-Up number and a Physical Channel.

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

- TmsD.h
- TmsD.cc

7.90 Tms::CycleInformation Class Reference

```
#include <TmsD.h>
```

Public Member Functions

- CycleInformation (BUInt32 pcycleNumber=BUInt32(), BString pcycleType=BString(), BList< CycleInformationPeriod > pperiods=BList< CycleInformationPeriod >())

Public Attributes

- BUInt32 cycleNumber

  The PS Cycle number.
- BString cycleType

  The Cycle Type Name.
- BList< CycleInformationPeriod > periods

  The list of cycle periods.

7.90.1 Constructor & Destructor Documentation

7.90.1.1 Tms::CycleInformation::CycleInformation (BUInt32 *pcycleNumber* = BUInt32 (), BString *pcycleType* = BString (), BList< CycleInformationPeriod > *pperiods* = BList<CycleInformationPeriod >())

7.90.2 Member Data Documentation

7.90.2.1 BUInt32 Tms::CycleInformation::cycleNumber

The PS Cycle number.

7.90.2.2 BString Tms::CycleInformation::cycleType

The Cycle Type Name.

7.90.2.3 BList< CycleInformationPeriod > Tms::CycleInformation::periods

The list of cycle periods.

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

- TmsD.h
- TmsD.cc

7.91 Tms::CycleInformationPeriod Class Reference

Cycle information.

```
#include <TmsD.h>
```

Public Member Functions

- CycleInformationPeriod (BUInt32 pcyclePeriod=BUInt32(), BUInt32 pstartTime=BUInt32(), BUInt32 pendTime=BUInt32(), BUInt32 pharmonic=BUInt32(), BUInt32 pnumBunches=BUInt32(), BUInt32 pbunchMask=BUInt32(), BUInt32 pnumValues=BUInt32())

Public Attributes

- BUInt32 cyclePeriod

  The Cycle Period.
- BUInt32 startTime

  The start time in ms.
- BUInt32 endTime

  The end time in ms.
- BUInt32 harmonic

  The Machines harmonic number.
- BUInt32 numBunches

  The number of bunches.
- BUInt32 bunchMask

  Bitmask defining which buckets the bunches are captured from. Bit 0 is bucket 1, bit 1 is bucket 2 etc.
- BUInt32 numValues

  The total number of raw data values available.

7.91.1 Detailed Description

Cycle information.

7.91.2 Constructor & Destructor Documentation

7.91.2.1 Tms::CycleInformationPeriod::CycleInformationPeriod (BUInt32 *pcyclePeriod* = BUInt32 ( ), BUInt32 *pstartTime* = BUInt32 ( ), BUInt32 *pendTime* = BUInt32 ( ), BUInt32 *pharmonic* = BUInt32 ( ), BUInt32 *pnumBunches* = BUInt32 ( ), BUInt32 *pbunchMask* = BUInt32 ( ), BUInt32 *pnumValues* = BUInt32 ( ))

7.91.3 Member Data Documentation

7.91.3.1 BUInt32 Tms::CycleInformationPeriod::bunchMask

Bitmask defining which buckets the bunches are captured from. Bit 0 is bucket 1, bit 1 is bucket 2 etc.

7.91.3.2 BUInt32 Tms::CycleInformationPeriod::cyclePeriod

The Cycle Period.

7.91.3.3 BUInt32 Tms::CycleInformationPeriod::endTime

The end time in ms.

7.91.3.4 BUInt32 Tms::CycleInformationPeriod::harmonic

The Machines harmonic number.

7.91.3.5 BUInt32 Tms::CycleInformationPeriod::numBunches

The number of bunches.

7.91.3.6 BUInt32 Tms::CycleInformationPeriod::numValues

The total number of raw data values available.

7.91.3.7 BUInt32 Tms::CycleInformationPeriod::startTime

The start time in ms.

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

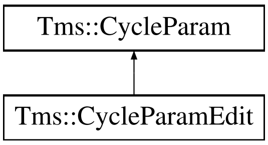
- TmsD.h
- TmsD.cc

7.92 Tms::CycleParam Class Reference

This class defines the parameters for a PS processing cycle.

```
#include <TmsD.h>
```

Inheritance diagram for Tms::CycleParam:



Public Member Functions

- CycleParam (BString pcycleType=BString(), BString pname=BString(), BString pinfo=BString(), BUInt32 pring=BUInt32(), BUInt32 pchannel=BUInt32(), BUInt32 pllCycleStartFrequency=BUInt32(), BUInt32 pllInitialFrequency=BUInt32(), BUInt32 pllInitialFrequencyDelay=BUInt32(), BUInt32 pllFrefGain=BUInt32(), BUInt32 pllGain=BUInt32(), BUInt32 pllDdsMinimum=BUInt32(), BUInt32 pllDdsMaximum=BUInt32(), BUInt32 pstateDelay=BUInt32(), BArray< BInt32 > pfrefPhaseDelay=BArray< BInt32 >(), BArray< PuStateTable > pstateTable=BArray< PuStateTable >(), BArray< BString > psettings=BArray< BString >())

Public Attributes

- BString cycleType

  The Cycle Type Name of this parameter set, normally the BEAM type the set of parameters is designed to measure.
- BString name

  The name of this set of parameters.
- BString info

  Information on this parameter set.
- BUInt32 ring

  The ring this configuration is for, 0 defines all rings.
- BUInt32 channel

  The channel number this configuration is for, 0 defines all channels.
- BUInt32 pllCycleStartFrequency

  This defines the initial PLL frequency. This is loaded on START_CYCLE.
- BUInt32 pllInitialFrequency

  This defines the initial PLL frequency. This is loaded after the delay given in pllInitialFrequencyDelay.
- BUInt32 pllInitialFrequencyDelay

  This defines the delay in milliseconds from START_CYCLE when the pllInitialFrequency is loaded.
- BUInt32 pllFrefGain

  The gain the FREF signal. This is a value in the range +-8191. A normal value would be around 4096.
- BUInt32 pllGain

  The gain of the PLL feedback system. This is the gain reduction of the PLL in terms of right bit shifts. A bit shift of 7 is about unity gain.
- BUInt32 pllDdsMinimum

  PLL DDS minimum frequency. If this and pllDdsMaximum is set to 0, this feature is disabled.
- BUInt32 pllDdsMaximum

  PLL DDS maximum frequency. If this and pllDdsMinimum is set to 0, this feature is disabled.
- BUInt32 stateDelay

  The delay in FREF periods when in the delay state.
- BArray< BInt32 > frefPhaseDelay

  The phase delay parameters for the Fref timing signal for each of the Pick-Up channels. This is set based on the position of the Pick-Up's in the PS ring. Its value is based of Fref / 512.
- BArray< PuStateTable > stateTable

  The array of State Table entries for the processing run.
- BArray< BString > settings

  A string array defining the settings for the states. Used for CycleParam editors.

7.92.1 Detailed Description

This class defines the parameters for a PS processing cycle.

7.92.2 Constructor & Destructor Documentation

7.92.2.1 Tms::CycleParam::CycleParam (BString *pcycleType* = BString ( ), BString *pname* = BString ( ), BString *pinfo* = BString ( ), BUInt32 *pring* = BUInt32 ( ), BUInt32 *pchannel* = BUInt32 ( ), BUInt32 *ppllCycleStartFrequency* = BUInt32 ( ), BUInt32 *ppllInitialFrequency* = BUInt32 ( ), BUInt32 *ppllInitialFrequencyDelay* = BUInt32 ( ), BUInt32 *ppllFrefGain* = BUInt32 ( ), BUInt32 *ppllGain* = BUInt32 ( ), BUInt32 *ppllDdsMinimum* = BUInt32 ( ), BUInt32 *ppllDdsMaximum* = BUInt32 ( ), BUInt32 *pstateDelay* = BUInt32 ( ), BArray< BInt32 > *pfrefPhaseDelay* = BArray<BInt32 > ( ), BArray< PuStateTable > *pstateTable* = BArray<PuStateTable > ( ), BArray< BString > *psettings* = BArray<BString > ( ))

7.92.3 Member Data Documentation

7.92.3.1 BUInt32 Tms::CycleParam::channel

The channel number this configuration is for, 0 defines all channels.

7.92.3.2 BString Tms::CycleParam::cycleType

The Cycle Type Name of this parameter set, normally the BEAM type the set of parameters is designed to measure.

7.92.3.3 BArray< BInt32 > Tms::CycleParam::frefPhaseDelay

The phase delay parameters for the Fref timing signal for each of the Pick-Up channels. This is set based on the position of the Pick-Up's in the PS ring. Its value is based of Fref / 512.

7.92.3.4 BString Tms::CycleParam::info

Information on this parameter set.

7.92.3.5 BString Tms::CycleParam::name

The name of this set of parameters.

7.92.3.6 BUInt32 Tms::CycleParam::pllCycleStartFrequency

This defines the initial PLL frequency. This is loaded on START_CYCLE.

7.92.3.7 BUInt32 Tms::CycleParam::pllDdsMaximum

PLL DDS maximum frequency. If this and pllDdsMinimum is set to 0, this feature is disabled.

7.92.3.8 BUInt32 Tms::CycleParam::pllDdsMinimum

PLL DDS minimum frequency. If this and pllDdsMaximum is set to 0, this feature is disabled.

7.92.3.9 BUInt32 Tms::CycleParam::pllFrefGain

The gain the FREF signal. This is a value in the range +-8191. A normal value would be around 4096.

7.92.3.10 BUInt32 Tms::CycleParam::pllGain

The gain of the PLL feedback system. This is the gain reduction of the PLL in terms of right bit shifts. A bit shift of 7 is about unity gain.

7.92.3.11 BUInt32 Tms::CycleParam::pllInitialFrequency

This defines the initial PLL frequency. This is loaded after the delay given in pllInitialFrequencyDelay.

7.92.3.12 BUInt32 Tms::CycleParam::pllInitialFrequencyDelay

This defines the delay in milliseconds from START_CYCLE when the pllInitialFrequency is loaded.

7.92.3.13 BUInt32 Tms::CycleParam::ring

The ring this configuration is for, 0 defines all rings.

7.92.3.14 BArray< BString > Tms::CycleParam::settings

A string array defining the settings for the states. Used for CycleParam editors.

7.92.3.15 BUInt32 Tms::CycleParam::stateDelay

The delay in FREF periods when in the delay state.

7.92.3.16 BArray< PuStateTable > Tms::CycleParam::stateTable

The array of State Table entries for the processing run.

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

- TmsD.h
- TmsD.cc

7.93 Tms::CycleParamDb Class Reference

Internal CycleParameter management class.

`#include <TmsLib.h>`

Public Member Functions

- CycleParamDb (BString baseDir=".")
- BError getCycleTypes (BList< BString > &typeList)

  Get the list of CycleParameter types in the directory.
- BError getFileNames (BList< BString > &fileList)

  Get a list of all of the CycleParameter file names.
- BError getCycleParams (BString fileName, Tms::CycleParam &param)

  Get the CycleParameter from the given file name.
- BError setCycleParams (Tms::CycleParam param)

  Set the CycleParameters. Writes to the appropriate file name.
- BError deleteCycleParams (BString cycleType, BUInt32 ring, BUInt32 puChannel)

  Deletes all CycleParameter definition files for the given ring/channel.
- BError readCycleParams (BString fileName, Tms::CycleParam &param)

  Reads a set of CycleParameters from a file.
- BError writeCycleParams (BString fileName, Tms::CycleParam param)

  Writes a set of CycleParameters to a file.

Private Attributes

- BString obaseDir

7.93.1 Detailed Description

Internal CycleParameter management class.

7.93.2 Constructor & Destructor Documentation

7.93.2.1 Tms::CycleParamDb::CycleParamDb (BString *baseDir* = " . ")

7.93.3 Member Function Documentation

7.93.3.1 BError Tms::CycleParamDb::deleteCycleParams (BString *cycleType*, BUInt32 *ring*, BUInt32 *puChannel*)

Deletes all CycleParameter definition files for the given ring/channel.

7.93.3.2 BError Tms::CycleParamDb::getCycleParams (BString *fileName*, Tms::CycleParam & *param*)

Get the CycleParameter from the given file name.

7.93.3.3 BError Tms::CycleParamDb::getCycleTypes (BList< BString > & *typeList*)

Get the list of CycleParameter types in the directory.

7.93.3.4 BError Tms::CycleParamDb::getFileNames (BList< BString > & *fileList*)

Get a list of all of the CycleParameter file names.

7.93.3.5 BError Tms::CycleParamDb::readCycleParams (BString *fileName*, Tms::CycleParam & *param*)

Reads a set of CycleParameters from a file.

7.93.3.6 BError Tms::CycleParamDb::setCycleParams (Tms::CycleParam *param*)

Set the CycleParameters. Writes to the appropriate file name.

7.93.3.7 BError Tms::CycleParamDb::writeCycleParams (BString *fileName*, Tms::CycleParam *param*)

Writes a set of CycleParameters to a file.

7.93.4 Member Data Documentation

7.93.4.1 BString Tms::CycleParamDb::obaseDir `[private]`

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

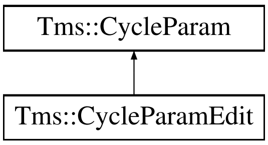
- TmsLib.h
- TmsLib.cc

7.94 Tms::CycleParamEdit Class Reference

Cycle Parameter management class.

```
#include <TmsCycleParam.h>
```

Inheritance diagram for Tms::CycleParamEdit:



Public Member Functions

- CycleParamEdit ()
- CycleParamEdit (const CycleParam &param)
- void clear ()
- BString getString ()

  Gets the Cycle Parameters in a string format for writing to a file or display.
- BError setString (BString str)

  Sets the Cycle Parameters from a string. For reading from a file.
- BError readFromFile (BString fileName)

  Reads the Cycle Parameters from a file.
- BError writeToFile (BString fileName)

  Writes the Cycle Parameters to a file.
- BError setStates (BList< CycleParamState > cycleStates)

  Sets the Cycle Parameter states given the state information list.
- BError getStates (BList< CycleParamState > &cycleStates)

  Returns the state information list describing the Cycle Parameter states. These may not be present.

Static Public Member Functions

- static void getDefaultState (CycleParamState &state)

  Get the default settings for a state.
- static void getdefaultPickupPositions (BArray< BInt32 > &pos)

  Get the default pickup positions.

Private Member Functions

- double value (int numSamples, int harmonic, double phase, int sample)
- int bunch (int numSamples, int harmonic, double phase, int sample)
- BError generateState (CycleParamState stateParam)

Additional Inherited Members

7.94.1 Detailed Description

Cycle Parameter management class.

7.94.2 Constructor & Destructor Documentation

7.94.2.1 Tms::CycleParamEdit::CycleParamEdit ( )

7.94.2.2 Tms::CycleParamEdit::CycleParamEdit (const CycleParam & *param*)

7.94.3 Member Function Documentation

7.94.3.1 int Tms::CycleParamEdit::bunch (int *numSamples,* int *harmonic,* double *phase,* int *sample*) [private]

7.94.3.2 void Tms::CycleParamEdit::clear ( )

7.94.3.3 BError Tms::CycleParamEdit::generateState (CycleParamState *stateParam*) [private]

This function will generate the phase tables for a given state. It is passed the parameters for the LO1 reference and the LO2 reference. If lo?Harmonic is 1, then FREF is generated.

7.94.3.4 void Tms::CycleParamEdit::getdefaultPickupPositions (BArray< BInt32 > & *pos*) [static]

Get the default pickup positions.

Calculates the base pickup phase values for the PS ring.

7.94.3.5 void Tms::CycleParamEdit::getDefaultState (CycleParamState & *state*) [static]

Get the default settings for a state.

7.94.3.6 BError Tms::CycleParamEdit::getStates (BList< CycleParamState > & *cycleStates*)

Returns the state information list describing the Cycle Parameter states. These may not be present.

7.94.3.7 BString Tms::CycleParamEdit::getString ( )

Gets the Cycle Parameters in a string format for writing to a file or display.

7.94.3.8 BError Tms::CycleParamEdit::readFromFile (BString *fileName*)

Reads the Cycle Parameters from a file.

7.94.3.9 BError Tms::CycleParamEdit::setStates (BList< CycleParamState > *cycleStates*)

Sets the Cycle Parameter states given the state information list.

7.94.3.10 BError Tms::CycleParamEdit::setString (BString *str*)

Sets the Cycle Parameters from a string. For reading from a file.

7.94.3.11 double Tms::CycleParamEdit::value (int *numSamples,* int *harmonic,* double *phase,* int *sample*) [private]

7.94.3.12 BError Tms::CycleParamEdit::writeToFile (BString *fileName*)

Writes the Cycle Parameters to a file.

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

- TmsCycleParam.h
- TmsCycleParam.cc

7.95 Tms::CycleParamItem Class Reference

```
#include <TmsD.h>
```

Public Member Functions

- CycleParamItem (BString pcycleType=BString(), BUInt32 pring=BUInt32(), BUInt32 pchannel=BUInt32())

Public Attributes

- BString cycleType

  The Cycle Type Name of this parameter set, normally the BEAM type the set of parameters is designed to measure.
- BUInt32 ring

  The ring number this configuration is for, 0 defines all rings.
- BUInt32 channel

  The channel number this configuration is for, 0 defines all channels.

7.95.1 Constructor & Destructor Documentation

7.95.1.1 Tms::CycleParamItem::CycleParamItem (BString *pcycleType* = BString (), BUInt32 *pring* = BUInt32 (), BUInt32 *pchannel* = BUInt32 ())

7.95.2 Member Data Documentation

7.95.2.1 BUInt32 Tms::CycleParamItem::channel

The channel number this configuration is for, 0 defines all channels.

7.95.2.2 BString Tms::CycleParamItem::cycleType

The Cycle Type Name of this parameter set, normally the BEAM type the set of parameters is designed to measure.

7.95.2.3 BUInt32 Tms::CycleParamItem::ring

The ring number this configuration is for, 0 defines all rings.

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

- TmsD.h
- TmsD.cc

7.96 Tms::CycleParamState Class Reference

```
#include <TmsCycleParam.h>
```

Public Member Functions

- CycleParamState ()
- void clear ()

    Clear the entry.
- void setNext (int nextNum, BUInt32 nextPeriod, bool f1RefSigma, bool f1LoMsb, bool f2RefSigma, bool f2LoMsb, bool pllF2, bool acquire)
- BString getString ()

    Returns the CycleParamState in string form.
- BError setString (BString str)

    Sets the CycleParamState from a string.

Public Attributes

- BUInt32 num

    The state number.
- BUInt32 period

    The cycle period.
- TmsState state

    The TMS State.
- BUInt32 bunchMask

    The set of bunches to capture bit mask.
- BUInt32 mean1Mask

    The set of bunches to pass through meanFilter1.
- BUInt32 mean2Mask

    The set of bunches to pass through meanFilter2.
- BUInt32 lo1Harmonic

    The LO harmonic number used in this state.
- double lo1Phase

    The phase offset of the LO as a fraction of FREF (+-1.0)
- BUInt32 lo2Harmonic

    The LO harmonic number used in this state.
- double lo2Phase

    The phase offset of the LO as a fraction of FREF (+-1.0)
- double gateWidth

    The gate pulse width as a fraction of LO (0 - 1.0)
- double gatePhase

    The gate phase offset as a fraction of LO (0 - 1.0)
- double blrWidth

    The gate pulse width as a fraction of LO (0 - 1.0)
- double blrPhase

    The gate phase offset as a fraction of LO (0 - 1.0)

7.96.1 Constructor & Destructor Documentation

7.96.1.1 Tms::CycleParamState::CycleParamState ( )

7.96.2 Member Function Documentation

7.96.2.1 void Tms::CycleParamState::clear ( )

Clear the entry.

7.96.2.2 BString Tms::CycleParamState::getString ( )

Returns the CycleParamState in string form.

7.96.2.3 void Tms::CycleParamState::setNext (int *nextNum*, BUInt32 *nextPeriod*, bool *f1RefSigma*, bool *f1LoMsb*, bool *f2RefSigma*, bool *f2LoMsb*, bool *pllF2*, bool *acquire*)

7.96.2.4 BError Tms::CycleParamState::setString (BString *str*)

Sets the CycleParamState from a string.

7.96.3 Member Data Documentation

7.96.3.1 double Tms::CycleParamState::blrPhase

The gate phase offset as a fraction of LO (0 - 1.0)

7.96.3.2 double Tms::CycleParamState::blrWidth

The gate pulse width as a fraction of LO (0 - 1.0)

7.96.3.3 BUInt32 Tms::CycleParamState::bunchMask

The set of bunches to capture bit mask.

7.96.3.4 double Tms::CycleParamState::gatePhase

The gate phase offset as a fraction of LO (0 - 1.0)

7.96.3.5 double Tms::CycleParamState::gateWidth

The gate pulse width as a fraction of LO (0 - 1.0)

7.96.3.6 BUInt32 Tms::CycleParamState::lo1Harmonic

The LO harmonic number used in this state.

7.96.3.7 double Tms::CycleParamState::lo1Phase

The phase offset of the LO as a fraction of FREF (+-1.0)

7.96.3.8 BUInt32 Tms::CycleParamState::lo2Harmonic

The LO harmonic number used in this state.

7.96.3.9 double Tms::CycleParamState::lo2Phase

The phase offset of the LO as a fraction of FREF (+-1.0)

7.96.3.10 BUInt32 Tms::CycleParamState::mean1Mask

The set of bunches to pass through meanFilter1.

7.96.3.11 BUInt32 Tms::CycleParamState::mean2Mask

The set of bunches to pass through meanFilter2.

7.96.3.12 BUInt32 Tms::CycleParamState::num

The state number.

7.96.3.13 BUInt32 Tms::CycleParamState::period

The cycle period.

7.96.3.14 TmsState Tms::CycleParamState::state

The TMS State.

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

- TmsCycleParam.h
- TmsCycleParam.cc

7.97 Tms::CycleTypeInformation Class Reference

```
#include <TmsD.h>
```

Public Member Functions

- CycleTypeInformation (BString pcycleType=BString(), BString pinfo=BString(), BList< CycleTypeInformationPeriod > pperiods=BList< CycleTypeInformationPeriod >())

Public Attributes

- BString cycleType

  The Cycle Type Name.
- BString info

  Information string on this cycle type.
- BList< CycleTypeInformationPeriod > periods

  The list of cycle periods.

7.97.1 Constructor & Destructor Documentation

7.97.1.1 Tms::CycleTypeInformation::CycleTypeInformation (BString *pcycleType* = BString(), BString *pinfo* = BString(), BList< CycleTypeInformationPeriod > *pperiods* = BList<CycleTypeInformationPeriod >())

7.97.2 Member Data Documentation

7.97.2.1 BString Tms::CycleTypeInformation::cycleType

The Cycle Type Name.

7.97.2.2 BString Tms::CycleTypeInformation::info

Information string on this cycle type.

7.97.2.3 BList< CycleTypeInformationPeriod > Tms::CycleTypeInformation::periods

The list of cycle periods.

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

- TmsD.h
- TmsD.cc

7.98 Tms::CycleTypeInformationPeriod Class Reference

Cycle Type information.

```
#include <TmsD.h>
```

Public Member Functions

- CycleTypeInformationPeriod (BUInt32 pcyclePeriod=BUInt32(), BUInt32 pharmonic=BUInt32(), BUInt32 pnumBunches=BUInt32(), BUInt32 pbunchMask=BUInt32())

Public Attributes

- BUInt32 cyclePeriod

  The Cycle Period.
- BUInt32 harmonic

  The Machines harmonic number.
- BUInt32 numBunches

  The number of bunches.
- BUInt32 bunchMask

  Bitmask defining which buckets the bunches are captured from. Bit 0 is bucket 1, bit 1 is bucket 2 etc.

7.98.1 Detailed Description

Cycle Type information.

7.98.2 Constructor & Destructor Documentation

7.98.2.1 Tms::CycleTypeInformationPeriod::CycleTypeInformationPeriod (BUInt32 *pcyclePeriod* = BUInt32 (), BUInt32 *pharmonic* = BUInt32 (), BUInt32 *pnumBunches* = BUInt32 (), BUInt32 *pbunchMask* = BUInt32 ())

7.98.3 Member Data Documentation

7.98.3.1 BUInt32 Tms::CycleTypeInformationPeriod::bunchMask

Bitmask defining which buckets the bunches are captured from. Bit 0 is bucket 1, bit 1 is bucket 2 etc.

7.98.3.2 BUInt32 Tms::CycleTypeInformationPeriod::cyclePeriod

The Cycle Period.

7.98.3.3 BUInt32 Tms::CycleTypeInformationPeriod::harmonic

The Machines harmonic number.

7.98.3.4 BUInt32 Tms::CycleTypeInformationPeriod::numBunches

The number of bunches.

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

- TmsD.h
- TmsD.cc

7.99 Tms::Data Class Reference

This class stores the raw data.

```
#include <TmsD.h>
```

Public Member Functions

- Data (BUInt32 pnumValues=BUInt32(), BUInt32 pdataType=BUInt32(), BUInt32 pnumBunches=BUInt32(), BUInt32 pnumChannels=BUInt32(), BArray< DataValue > pdataValues=BArray< DataValue >(), BArray< BError > perrors=BArray< BError >())

Public Attributes

- BUInt32 numValues

  The total number of data samples.
- BUInt32 dataType

  The type of data in the data block.
- BUInt32 numBunches

  The number of bunches.
- BUInt32 numChannels

  The number of channels.
- BArray< DataValue > dataValues

  The data.
- BArray< BError > errors

  Individual errors for each channel within dataValues.

7.99.1 Detailed Description

This class stores the raw data.

7.99.2 Constructor & Destructor Documentation

7.99.2.1 Tms::Data::Data (BUInt32 *pnumValues* = BUInt32 ( ) , BUInt32 *pdataType* = BUInt32 ( ) , BUInt32 *pnumBunches* = BUInt32 ( ) , BUInt32 *pnumChannels* = BUInt32 ( ) , BArray< DataValue > *pdataValues* = BArray<DataValue > ( ) , BArray< BError > *perrors* = BArray<BError > ( ))

7.99.3 Member Data Documentation

7.99.3.1 BUInt32 Tms::Data::dataType

The type of data in the data block.

7.99.3.2 BArray<DataValue > Tms::Data::dataValues

The data.

7.99.3.3 BArray<BError > Tms::Data::errors

Individual errors for each channel within dataValues.

7.99.3.4 BUInt32 Tms::Data::numBunches

The number of bunches.

7.99.3.5 BUInt32 Tms::Data::numChannels

The number of channels.

7.99.3.6 BUInt32 Tms::Data::numValues

The total number of data samples.

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

- TmsD.h
- TmsD.cc

7.100 Tms::DataInfo Class Reference

This class defines the data to be acquired and/or fetched.

```
#include <TmsD.h>
```

Public Member Functions

- DataInfo (BUInt32 pcycleNumber=BUInt32(), BUInt32 pchannel=BUInt32(), BUInt32 pcyclePeriod=BUInt32(), BUInt32 pstartTime=BUInt32(), BUInt32 porbitNumber=BUInt32(), BUInt32 pbunchNumber=BUInt32(), BUInt32 pfunction=BUInt32(), BUInt32 pargument=BUInt32(), BUInt32 pnumValues=BUInt32(), BInt32 pbeyondPeriod=BInt32())

Public Attributes

- BUInt32 cycleNumber

  The PS Cycle number.
- BUInt32 channel

  The pick-up channel number.
- BUInt32 cyclePeriod

  The cycle period the data is from.
- BUInt32 startTime

  The start time in milli-seconds in the cycle period (starting from 0)
- BUInt32 orbitNumber

  The starting orbit number (starting from 0)
- BUInt32 bunchNumber

  The Bunch number (starting from 1 (0 is all bunches))
- BUInt32 function

  The data processing function to perform or performed. (0 normal data)
- BUInt32 argument

  The Argument to the data processing function.
- BUInt32 numValues

  The total number of data points to return.
- BInt32 beyondPeriod

  If set allows reads of data beyond the end of the period.

7.100.1 Detailed Description

This class defines the data to be acquired and/or fetched.

7.100.2 Constructor & Destructor Documentation

7.100.2.1 Tms::DataInfo::DataInfo (**BUInt32** *pcycleNumber* = **BUInt32** ( ), **BUInt32** *pchannel* = **BUInt32** ( ), **BUInt32** *pcyclePeriod* = **BUInt32** ( ), **BUInt32** *pstartTime* = **BUInt32** ( ), **BUInt32** *porbitNumber* = **BUInt32** ( ), **BUInt32** *pbunchNumber* = **BUInt32** ( ), **BUInt32** *pfunction* = **BUInt32** ( ), **BUInt32** *pargument* = **BUInt32** ( ), **BUInt32** *pnumValues* = **BUInt32** ( ), **BInt32** *pbeyondPeriod* = **BInt32** ( ))

7.100.3 Member Data Documentation

7.100.3.1 BUInt32 Tms::DataInfo::argument

The Argument to the data processing function.

7.100.3.2 BInt32 Tms::DataInfo::beyondPeriod

If set allows reads of data beyond the end of the period.

7.100.3.3 BUInt32 Tms::DataInfo::bunchNumber

The Bunch number (starting from 1 (0 is all bunches))

7.100.3.4 BUInt32 Tms::DataInfo::channel

The pick-up channel number.

7.100.3.5 BUInt32 Tms::DataInfo::cycleNumber

The PS Cycle number.

7.100.3.6 BUInt32 Tms::DataInfo::cyclePeriod

The cycle period the data is from.

7.100.3.7 BUInt32 Tms::DataInfo::function

The data processing function to perform or performed. (0 normal data)

7.100.3.8 BUInt32 Tms::DataInfo::numValues

The total number of data points to return.

7.100.3.9 BUInt32 Tms::DataInfo::orbitNumber

The starting orbit number (starting from 0)

7.100.3.10 BUInt32 Tms::DataInfo::startTime

The start time in milli-seconds in the cycle period (starting from 0)

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

- TmsD.h
- TmsD.cc

7.101 Tms::DataValue Class Reference

This is the definition of a single data value.

```
#include <TmsD.h>
```

Public Member Functions

- DataValue (BInt16 psigma=BInt16(), BInt16 pdeltaX=BInt16(), BInt16 pdeltaY=BInt16(), BInt16 ptime=BInt16())

Public Attributes

- BInt16 sigma

  The Sigma value.
- BInt16 deltaX

  The DeltaX value.
- BInt16 deltaY

  The DeltaY value.
- BInt16 time

  The Time in ms this sample was processed.

7.101.1 Detailed Description

This is the definition of a single data value.

7.101.2 Constructor & Destructor Documentation

7.101.2.1 Tms::DataValue::DataValue (BInt16 *psigma* = BInt16 ( ), BInt16 *pdeltaX* = BInt16 ( ), BInt16 *pdeltaY* = BInt16 ( ), BInt16 *ptime* = BInt16 ( ))

7.101.3 Member Data Documentation

7.101.3.1 BInt16 Tms::DataValue::deltaX

The DeltaX value.

7.101.3.2 BInt16 Tms::DataValue::deltaY

The DeltaY value.

7.101.3.3 BInt16 Tms::DataValue::sigma

The Sigma value.

7.101.3.4 BInt16 Tms::DataValue::time

The Time in ms this sample was processed.

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

- TmsD.h
- TmsD.cc

7.102 Tms::NameValue Class Reference

```
#include <TmsD.h>
```

Public Member Functions

- NameValue (BString pname=BString(), BString pvalue=BString())

Public Attributes

- BString name

  The Name of the value.
- BString value

  The actual value in string form.

7.102.1 Constructor & Destructor Documentation

7.102.1.1 Tms::NameValue::NameValue (BString *pname* = BString (), BString *pvalue* = BString ())

7.102.2 Member Data Documentation

7.102.2.1 BString Tms::NameValue::name

The Name of the value.

7.102.2.2 BString Tms::NameValue::value

The actual value in string form.

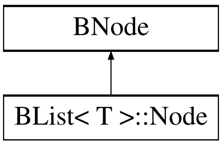
The documentation for this class was generated from the following files:

- TmsD.h
- TmsD.cc

7.103 BList< T >::Node Class Reference

```
#include <BList.h>
```

Inheritance diagram for BList< T >::Node:



Public Member Functions

- Node (const T &i)

Public Attributes

- T item

7.103.1 Constructor & Destructor Documentation

7.103.1.1 template<class T> BList< T >::Node::Node (const T & *i*) [inline]

7.103.2 Member Data Documentation

7.103.2.1 template<class T> T BList< T >::Node::item

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

- /src/cern/tms/beam/libBeam/BList.h

7.104 Tms::PuChannel Class Reference

This class stores a Physical Pick-Up channel id.

```
#include <TmsD.h>
```

Public Member Functions

- PuChannel (BUInt8 pmoduleNum=BUInt8(), BUInt8 ppupeNum=BUInt8(), BUInt8 ppupeChan=BUInt8())

Public Attributes

- BUInt8 moduleNum

  The Module number.
- BUInt8 pupeNum

  The PUPE number.
- BUInt8 pupeChan

  The PUPE channel.

7.104.1 Detailed Description

This class stores a Physical Pick-Up channel id.

7.104.2 Constructor & Destructor Documentation

7.104.2.1 Tms::PuChannel::PuChannel (**BUInt8** *pmoduleNum* = **BUInt8** ( ) , **BUInt8** *ppupeNum* = **BUInt8** ( ) , **BUInt8** *ppupeChan* = **BUInt8** ( ))

7.104.3 Member Data Documentation

7.104.3.1 BUInt8 Tms::PuChannel::moduleNum

The Module number.

7.104.3.2 BUInt8 Tms::PuChannel::pupeChan

The PUPE channel.

7.104.3.3 BUInt8 Tms::PuChannel::pupeNum

The PUPE number.

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

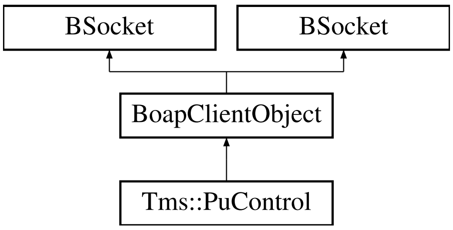
- TmsD.h
- TmsD.cc

7.105 Tms::PuControl Class Reference

This class defines the parameters for a test data capture.

`#include <TmsC.h>`

Inheritance diagram for Tms::PuControl:



Public Member Functions

- PuControl (BString name="")
- BError getVersion (BString &version)

  Gets the software version.
- BError init ()

  Initialises the system including loading all of the PUPE engines firmware. The call will return an error object indicating success or an error condition as appropriate.
- BError setProcessPriority (BUInt32 priority)

  Sets the priority of the process servicing this service.
- BError configure (BUInt32 ring, ConfigInfo configInfo)

  Configure the system for use. This includes mapping the individual physical PickUp channels to logical pickup channels.
- BError setControlInfo (CycleParam params)

  Sets the control information for the cycle type given and subsequent cycles. The parameters for the processing cycle are passed, this includes the Phase and State table information. The call will return an error object indicating success or an error.
- BError setNextCycle (BUInt32 cycleNumber, BString cycleType)

  Sets the cycle number and type for the next processing cycle. The call will return an error object indicating success or an error condition as appropriate. This should be called at least 10ms before the next CYCLE_START event.
- BError test (BList< BError > &errors)

  Performs a basic test of the system returning a list of errors. The call will return an error object indicating success or an error condition as appropriate.
- BError getStatus (BList< NameValue > &statusList)

  Returns the current status of the system. This information includes the number of Pick-Up's present and their individual status.
- BError getStatistics (BList< NameValue > &statsList)

  Returns a list of the statistic values as name/value pairs. The call will return an error object indicating success or an error condition as appropriate.
- BError getMasterPuChannel (PuChannel &puChannel)

  Returns the master PU Channel for timing.
- BError setTestMode (PuChannel puChannel, BUInt32 testOutput, BUInt32 timingDisableMask)

  The signal source for the digital test output connector. 0: None, 1: FrefLocal. The timingDisableMask bit mask defines which of the timing inputs should be disabled. If a timing input is disabled it can be still operated by software command.
- BError setTimingSignals (PuChannel puChannel, BUInt32 timingSignals)

  This function sets the given timing signals to the values as defined in the timingSignals bit array.
- BError captureDiagnostics (PuChannel puChannel, TestCaptureInfo captureInfo, BArray< BUInt64 > &data)

This function will capture test data.

- BError setTestData (PuChannel puChannel, BInt32 on, BArray< BUInt32 > data)

  This function will set a PU channel to sample data from memory rather than the ADC's.
- BError setPupeConfig (PuChannel puPhysChannel, PupeConfig pupeConfig)

  Sets special PUPE configuration for test purposes.
- BError getPupeConfig (PuChannel puPhysChannel, PupeConfig &pupeConfig)

  Gets special PUPE configuration for test purposes.

Additional Inherited Members

7.105.1 Detailed Description

This class defines the parameters for a test data capture.

This class stores a Physical Pick-Up channel id This class stores the status of an individual Pick-Up This class describes the configuration of the TMS This class defines the data to be acquired and/or fetched This is the definition of a single data value This class stores the raw data This class defines the Pick-Up state table This class defines the parameters for a PS processing cycleCycle information Cycle Type information This interface provides functions to control, test and get statistics from an individual pick-up

7.105.2 Constructor & Destructor Documentation

7.105.2.1 Tms::PuControl::PuControl (**BString** *name =* " ")

7.105.3 Member Function Documentation

7.105.3.1 BError Tms::PuControl::captureDiagnostics (**PuChannel** *puChannel,* **TestCaptureInfo** *captureInfo,* **BArray< BUInt64 >** & *data*)

This function will capture test data.

7.105.3.2 BError Tms::PuControl::configure (**BUInt32** *ring,* **ConfigInfo** *configInfo*)

Configure the system for use. This includes mapping the individual physical PickUp channels to logical pickup channels.

7.105.3.3 BError Tms::PuControl::getMasterPuChannel (**PuChannel** & *puChannel*)

Returns the master PU Channel for timing.

7.105.3.4 BError Tms::PuControl::getPupeConfig (**PuChannel** *puPhysChannel,* **PupeConfig** & *pupeConfig*)

Gets special PUPE configuration for test purposes.

7.105.3.5 BError Tms::PuControl::getStatistics (**BList< NameValue >** & *statsList*)

Returns a list of the statistic values as name/value pairs. The call will return an error object indicating success or an error condition as appropriate.

7.105.3.6 BError Tms::PuControl::getStatus (BList< NameValue > & *statusList*)

Returns the current status of the system. This information includes the number of Pick-Up's present and their individual status.

7.105.3.7 BError Tms::PuControl::getVersion (BString & *version*)

Gets the software version.

Parameters

version	A string variable filled in with the version number string.

7.105.3.8 BError Tms::PuControl::init ( )

Initialises the system including loading all of the PUPE engines firmware. The call will return an error object indicating success or an error condition as appropriate.

7.105.3.9 BError Tms::PuControl::setControlInfo (CycleParam *params*)

Sets the control information for the cycle type given and subsequent cycles. The parameters for the processing cycle are passed, this includes the Phase and State table information. The call will return an error object indicating success or an error.

7.105.3.10 BError Tms::PuControl::setNextCycle (BUInt32 *cycleNumber,* BString *cycleType*)

Sets the cycle number and type for the next processing cycle. The call will return an error object indicating success or an error condition as appropriate. This should be called at least 10ms before the next CYCLE_START event.

7.105.3.11 BError Tms::PuControl::setProcessPriority (BUInt32 *priority*)

Sets the priority of the process servicing this service.

7.105.3.12 BError Tms::PuControl::setPupeConfig (PuChannel *puPhysChannel,* PupeConfig *pupeConfig*)

Sets special PUPE configuration for test purposes.

7.105.3.13 BError Tms::PuControl::setTestData (PuChannel *puChannel,* BInt32 *on,* BArray< BUInt32 > *data*)

This function will set a PU channel to sample data from memory rather than the ADC's.

7.105.3.14 BError Tms::PuControl::setTestMode (PuChannel *puChannel,* BUInt32 *testOutput,* BUInt32 *timingDisableMask*)

The signal source for the digital test output connector. 0: None, 1: FrefLocal. The timingDisableMask bit mask defines which of the timing inputs should be disabled. If a timing input is disabled it can be still operated by software command.

7.105.3.15 BError Tms::PuControl::setTimingSignals (PuChannel *puChannel,* BUInt32 *timingSignals*)

This function sets the given timing signals to the values as defined in the timingSignals bit array.

7.105.3.16 BError Tms::PuControl::test (BList< BError > & *errors*)

Performs a basic test of the system returning a list of errors. The call will return an error object indicating success or an error condition as appropriate.

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

- TmsC.h
- TmsC.cc
- tmsFunctions.dox

7.106 Tms::PupeConfig Class Reference

```
#include <TmsD.h>
```

Public Member Functions

- PupeConfig (BUInt32 pinternalTimingMask=BUInt32(), BInt32 pdoubleInjection=BInt32(), BInt32 padcSysclkSync=BInt32(), BInt32 pdisableBlr=BInt32())

Public Attributes

- BUInt32 internalTimingMask

  Use internal, software/hardware generated, timing signals for the given signals.
- BInt32 doubleInjection

  Simulate double injection.
- BInt32 adcSysclkSync

  Sets the ADC clock to be synchronised with the SYSCLK timing clock.
- BInt32 disableBlr

  Disable the BLR algorithm.

7.106.1 Constructor & Destructor Documentation

7.106.1.1 Tms::PupeConfig::PupeConfig (BUInt32 *pinternalTimingMask* = BUInt32 (), BInt32 *pdoubleInjection* = BInt32 (), BInt32 *padcSysclkSync* = BInt32 (), BInt32 *pdisableBlr* = BInt32 ())

7.106.2 Member Data Documentation

7.106.2.1 BInt32 Tms::PupeConfig::adcSysclkSync

Sets the ADC clock to be synchronised with the SYSCLK timing clock.

7.106.2.2 BInt32 Tms::PupeConfig::disableBlr

Disable the BLR algorithm.

7.106.2.3 BInt32 Tms::PupeConfig::doubleInjection

Simulate double injection.

7.106.2.4 BUInt32 Tms::PupeConfig::internalTimingMask

Use internal, software/hardware generated, timing signals for the given signals.

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

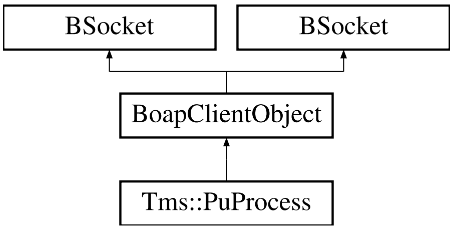
- TmsD.h
- TmsD.cc

7.107 Tms::PuProcess Class Reference

This interface provides functions to configure and capture data from individual pick-up.

`#include <TmsC.h>`

Inheritance diagram for Tms::PuProcess:



Public Member Functions

- PuProcess (BString name="")
- BError getVersion (BString &version)

    Gets the software version.
- BError getCycleInformation (BUInt32 cycleNumber, CycleInformation &cycleInformation)

    Gets information on given cycle number.
- BError getStatus (PuChannel puChannel, PuStatus &puStatus)
- BError getData (PuChannel puChannel, DataInfo dataInfo, Data &data, BUInt32 &orbitNumber)

    This function returns a set of data from the data present in the data cache or directly from the Pick-Up processing engines. The DataInfo object describes the data required. The call will return the required data along with an error object indicating success or an error condition as appropriate. The call will block until data is ready.
- BError addEventServer (BString name)

    Adds an event server.
- BError requestData (PuChannel puChannel, DataInfo dataInfo)

    This adds a request for some data. The DataInfo object defines the data required. This request can be made at any time. If the data is present in cache the data will be available immediately, if not the system will await the data from a subsequent processing cycle. When the data is available a "data" event will be sent to the client. Not that it is not necessary to use requestData. The client can call getData() directly although this call will block until the data is actually ready.

Additional Inherited Members

7.107.1 Detailed Description

This interface provides functions to configure and capture data from individual pick-up.

7.107.2 Constructor & Destructor Documentation

7.107.2.1 Tms::PuProcess::PuProcess (BString *name* = "")

7.107.3 Member Function Documentation

7.107.3.1 BError Tms::PuProcess::addEventServer (BString *name*)

Adds an event server.

7.107.3.2 BError Tms::PuProcess::getCycleInformation (BUInt32 *cycleNumber,* CycleInformation & *cycleInformation*)

Gets information on given cycle number.

7.107.3.3 BError Tms::PuProcess::getData (PuChannel *puChannel,* DataInfo *dataInfo,* Data & *data,* BUInt32 & *orbitNumber*)

This function returns a set of data from the data present in the data cache or directly from the Pick-Up processing engines. The DataInfo object describes the data required. The call will return the required data along with an error object indicating success or an error condition as appropriate. The call will block until data is ready.

7.107.3.4 BError Tms::PuProcess::getStatus (PuChannel *puChannel,* PuStatus & *puStatus*)

7.107.3.5 BError Tms::PuProcess::getVersion (BString & *version*)

Gets the software version.

7.107.3.6 BError Tms::PuProcess::requestData (PuChannel *puChannel,* DataInfo *dataInfo*)

This adds a request for some data. The DataInfo object defines the data required. This request can be made at any time. If the data is present in cache the data will be available immediately, if not the system will await the data from a subsequent processing cycle. When the data is available a "data" event will be sent to the client. Not that it is not necessary to use requestData. The client can call getData() directly although this call will block until the data is actually ready.

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

- TmsC.h
- TmsC.cc

7.108 Tms::PuStateTable Class Reference

This class defines the Pick-Up state table.

```
#include <TmsD.h>
```

Public Member Functions

- PuStateTable (BUInt32 pperiod=BUInt32(), BUInt32 pstate=BUInt32(), BUInt32 pharmonic=BUInt32(), BUInt32 pnumBunches=BUInt32(), BUInt32 pbunchMask=BUInt32(), BArray< BUInt8 > pphaseTable=BArray< BUInt8 >())

Public Attributes

- BUInt32 period

  The Cycle period this state is used for.
- BUInt32 state

  The State table entry.
- BUInt32 harmonic

  The harmonic number for this state.
- BUInt32 numBunches

  The number of bunches to capture.
- BUInt32 bunchMask

  Bitmask defining which buckets the bunches are captured from. Bit 0 is bucket 1, bit 1 is bucket 2 etc.
- BArray< BUInt8 > phaseTable

  The Phase table for this state.

7.108.1 Detailed Description

This class defines the Pick-Up state table.

7.108.2 Constructor & Destructor Documentation

7.108.2.1 Tms::PuStateTable::PuStateTable (BUInt32 *pperiod* = BUInt32 ( ) , BUInt32 *pstate* = BUInt32 ( ) , BUInt32 *pharmonic* = BUInt32 ( ) , BUInt32 *pnumBunches* = BUInt32 ( ) , BUInt32 *pbunchMask* = BUInt32 ( ) , BArray< BUInt8 > *pphaseTable* = BArray<BUInt8 > ( ))

7.108.3 Member Data Documentation

7.108.3.1 BUInt32 Tms::PuStateTable::bunchMask

Bitmask defining which buckets the bunches are captured from. Bit 0 is bucket 1, bit 1 is bucket 2 etc.

7.108.3.2 BUInt32 Tms::PuStateTable::harmonic

The harmonic number for this state.

7.108.3.3 BUInt32 Tms::PuStateTable::numBunches

The number of bunches to capture.

7.108.3.4 BUInt32 Tms::PuStateTable::period

The Cycle period this state is used for.

7.108.3.5 BArray<BUInt8 > Tms::PuStateTable::phaseTable

The Phase table for this state.

7.108.3.6 BUInt32 Tms::PuStateTable::state

The State table entry.

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

- TmsD.h
- TmsD.cc

7.109 Tms::PuStatus Class Reference

This class stores the status of an individual Pick-Up.

```
#include <TmsD.h>
```

Public Member Functions

- PuStatus (BInt32 prunning=BInt32(), BError perror=BError())

Public Attributes

- BInt32 running

  The Pick-Up is currently running.
- BError error

  The Pick-Up's current error status.

7.109.1 Detailed Description

This class stores the status of an individual Pick-Up.

7.109.2 Constructor & Destructor Documentation

7.109.2.1 Tms::PuStatus::PuStatus (BInt32 *prunning* = BInt32 () , BError *perror* = BError ())

7.109.3 Member Data Documentation

7.109.3.1 BError Tms::PuStatus::error

The Pick-Up's current error status.

7.109.3.2 BInt32 Tms::PuStatus::running

The Pick-Up is currently running.

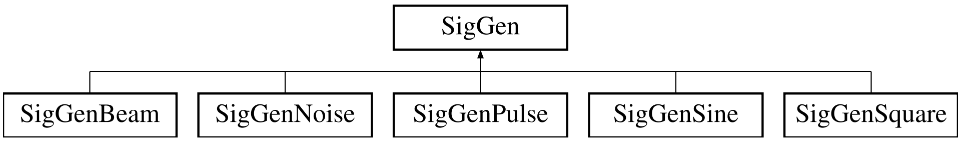
The documentation for this class was generated from the following files:

- TmsD.h
- TmsD.cc

7.110 SigGen Class Reference

```
#include <SigGen.h>
```

Inheritance diagram for SigGen:



Public Member Functions

- SigGen ()
- virtual ∼SigGen ()
- BError config (double sampleRate)
- virtual BError generate (Sample ∗data, int numSamples)

Protected Attributes

- double osampleRate
- unsigned long long ox

7.110.1 Constructor & Destructor Documentation

7.110.1.1 SigGen::SigGen ( )

7.110.1.2 SigGen::∼SigGen ( ) [virtual]

7.110.2 Member Function Documentation

7.110.2.1 BError SigGen::config (double *sampleRate*)

7.110.2.2 BError SigGen::generate (Sample ∗ *data,* int *numSamples*) [virtual]

Reimplemented in SigGenBeam, SigGenPulse, SigGenNoise, SigGenSquare, and SigGenSine.

7.110.3 Member Data Documentation

7.110.3.1 double SigGen::osampleRate [protected]

7.110.3.2 unsigned long long SigGen::ox [protected]

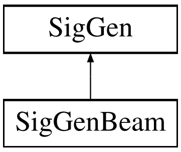
The documentation for this class was generated from the following files:

- SigGen.h
- SigGen.cpp

7.111 SigGenBeam Class Reference

```
#include <SigGen.h>
```

Inheritance diagram for SigGenBeam:



Public Member Functions

- SigGenBeam ()
- virtual ∼SigGenBeam ()
- BError config (double sampleRate, double fref, int harmonic, int bunchSet, double reduce, int blr, double amplitude)
- BError generate (Sample ∗data, int numSamples)
- BError generateIntegrated (Sample ∗data, int numSamples)

Public Attributes

- int oharmonic
- int obunchSet
- double oreduce
- int oblr
- double oamplitude
- double ofref

Additional Inherited Members

7.111.1 Constructor & Destructor Documentation

7.111.1.1 SigGenBeam::SigGenBeam ( )

7.111.1.2 SigGenBeam::∼SigGenBeam ( ) [virtual]

7.111.2 Member Function Documentation

7.111.2.1 BError SigGenBeam::config (double *sampleRate,* double *fref,* int *harmonic,* int *bunchSet,* double *reduce,* int *blr,* double *amplitude*)

7.111.2.2 BError SigGenBeam::generate (Sample ∗ *data,* int *numSamples*) [virtual]

Reimplemented from SigGen.

7.111.2.3 BError SigGenBeam::generateIntegrated (Sample ∗ *data,* int *numSamples*)

7.111.3 Member Data Documentation

7.111.3.1 double SigGenBeam::oamplitude

7.111.3.2 int SigGenBeam::oblr

7.111.3.3 int SigGenBeam::obunchSet

7.111.3.4 double SigGenBeam::ofref

7.111.3.5 int SigGenBeam::oharmonic

7.111.3.6 double SigGenBeam::oreduce

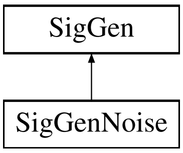
The documentation for this class was generated from the following files:

- SigGen.h
- SigGen.cpp

7.112 SigGenNoise Class Reference

```
#include <SigGen.h>
```

Inheritance diagram for SigGenNoise:



Public Member Functions

- SigGenNoise ()
- virtual ∼SigGenNoise ()
- BError config (double sampleRate, double amplitude=1.0)
- BError generate (Sample ∗data, int numSamples)

Public Attributes

- double oamplitude

Additional Inherited Members

7.112.1 Constructor & Destructor Documentation

7.112.1.1 SigGenNoise::SigGenNoise ( )

7.112.1.2 SigGenNoise::∼SigGenNoise ( ) `[virtual]`

7.112.2 Member Function Documentation

7.112.2.1 BError SigGenNoise::config (double *sampleRate,* double *amplitude* = `1.0`)

7.112.2.2 BError SigGenNoise::generate (Sample ∗ *data,* int *numSamples*) `[virtual]`

Reimplemented from SigGen.

7.112.3 Member Data Documentation

7.112.3.1 double SigGenNoise::oamplitude

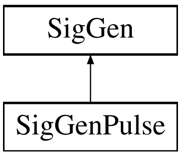
The documentation for this class was generated from the following files:

- SigGen.h
- SigGen.cpp

7.113 SigGenPulse Class Reference

```
#include <SigGen.h>
```

Inheritance diagram for SigGenPulse:



Public Member Functions

- SigGenPulse ()
- virtual ∼SigGenPulse ()
- BError config (double sampleRate, double freq, double amplitude, double onTime, double startTime=0.0)
- BError generate (Sample ∗data, int numSamples)

Public Attributes

- double ofreq
- double oamplitude
- double oonTime
- double ostartTime

Additional Inherited Members

7.113.1 Constructor & Destructor Documentation

7.113.1.1 SigGenPulse::SigGenPulse ( )

7.113.1.2 SigGenPulse::∼SigGenPulse ( ) [virtual]

7.113.2 Member Function Documentation

7.113.2.1 BError SigGenPulse::config (double *sampleRate,* double *freq,* double *amplitude,* double *onTime,* double *startTime* = 0.0)

7.113.2.2 BError SigGenPulse::generate (Sample ∗ *data,* int *numSamples*) [virtual]

Reimplemented from SigGen.

7.113.3 Member Data Documentation

7.113.3.1 double SigGenPulse::oamplitude

7.113.3.2 double SigGenPulse::ofreq

7.113.3.3 double SigGenPulse::oonTime

7.113.3.4 double SigGenPulse::ostartTime

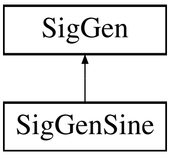
The documentation for this class was generated from the following files:

- SigGen.h
- SigGen.cpp

7.114 SigGenSine Class Reference

```
#include <SigGen.h>
```

Inheritance diagram for SigGenSine:



Public Member Functions

- SigGenSine ()
- virtual ∼SigGenSine ()
- BError config (double sampleRate, double freq, double amplitude=1.0)
- BError generate (Sample ∗data, int numSamples)

Public Attributes

- double ofreq
- double oamplitude

Additional Inherited Members

7.114.1 Constructor & Destructor Documentation

7.114.1.1 SigGenSine::SigGenSine ( )

7.114.1.2 SigGenSine::∼SigGenSine ( ) [virtual]

7.114.2 Member Function Documentation

7.114.2.1 BError SigGenSine::config (double *sampleRate*, double *freq*, double *amplitude* = 1.0)

7.114.2.2 BError SigGenSine::generate (Sample ∗ *data,* int *numSamples*) `[virtual]`

Reimplemented from SigGen.

7.114.3 Member Data Documentation

7.114.3.1 double SigGenSine::oamplitude

7.114.3.2 double SigGenSine::ofreq

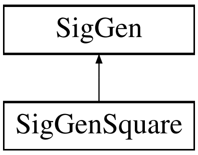
The documentation for this class was generated from the following files:

- SigGen.h
- SigGen.cpp

7.115 SigGenSquare Class Reference

`#include <SigGen.h>`

Inheritance diagram for SigGenSquare:



Public Member Functions

- SigGenSquare ()
- virtual ∼SigGenSquare ()
- BError config (double sampleRate, double freq, double amplitude=1.0, double offset=0.0)
- BError generate (Sample ∗data, int numSamples)

Public Attributes

- double ofreq
- double oamplitude
- double ooffset

Additional Inherited Members

7.115.1 Constructor & Destructor Documentation

7.115.1.1 SigGenSquare::SigGenSquare ( )

7.115.1.2 SigGenSquare::∼SigGenSquare ( ) `[virtual]`

7.115.2 Member Function Documentation

7.115.2.1 BError SigGenSquare::config (double *sampleRate,* double *freq,* double *amplitude =* `1.0`*,* double *offset =* `0.0`)

7.115.2.2 BError SigGenSquare::generate (Sample ∗ *data,* int *numSamples*) [virtual]

Reimplemented from SigGen.

7.115.3 Member Data Documentation

7.115.3.1 double SigGenSquare::oamplitude

7.115.3.2 double SigGenSquare::ofreq

7.115.3.3 double SigGenSquare::ooffset

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

- SigGen.h
- SigGen.cpp

7.116 Tms::Simulation Class Reference

```
#include <TmsD.h>
```

Public Member Functions

- Simulation (BInt32 ptiming=BInt32(), BInt32 pdoubleInjection=BInt32(), BInt32 pdata=BInt32(), BInt32 psetNextCycle=BInt32(), BString pcycleType=BString())

Public Attributes

- BInt32 timing

  Simulate timing signals.
- BInt32 doubleInjection

  Simulate double injection.
- BInt32 data

  Simulate FREF and Analogue Sigma, DeltaX and DeltaY data.
- BInt32 setNextCycle

  Simulate setNextCycle.
- BString cycleType

  The cycle type to set.

7.116.1 Constructor & Destructor Documentation

7.116.1.1 Tms::Simulation::Simulation (BInt32 *ptiming* = BInt32 ( ) , BInt32 *pdoubleInjection* = BInt32 ( ) , BInt32 *pdata* = BInt32 ( ) , BInt32 *psetNextCycle* = BInt32 ( ) , BString *pcycleType* = BString ( ))

7.116.2 Member Data Documentation

7.116.2.1 BString Tms::Simulation::cycleType

The cycle type to set.

7.116.2.2 BInt32 Tms::Simulation::data

Simulate FREF and Analogue Sigma, DeltaX and DeltaY data.

7.116.2.3 BInt32 Tms::Simulation::doubleInjection

Simulate double injection.

7.116.2.4 BInt32 Tms::Simulation::setNextCycle

Simulate setNextCycle.

7.116.2.5 BInt32 Tms::Simulation::timing

Simulate timing signals.

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

- TmsD.h
- TmsD.cc

7.117 Tms::TestCaptureInfo Class Reference

This class defines the parameters for a test data capture.

```
#include <TmsD.h>
```

Public Member Functions

- TestCaptureInfo (BUInt32 psource=BUInt32(), BUInt32 pclock=BUInt32(), BUInt32 pstartTime=BUInt32(), BUInt32 ppostTriggerDelay=BUInt32(), BUInt32 ptriggerMask=BUInt32(), BInt32 ptriggerAnd=BInt32(), BInt32 ptriggerStore=BInt32(), BInt32 ptriggerSourceData=BInt32(), BInt32 ptriggerStateEnable=BInt32(), BUInt32 ptriggerState=BUInt32())

Public Attributes

- BUInt32 source

  The source data (0 - 3)
- BUInt32 clock

  The Clock source.
- BUInt32 startTime

  The start time in ms from CYCLE_START before trigger is activated.
- BUInt32 postTriggerDelay

  The delay, in clock cycles, after the trigger before capture starts.
- BUInt32 triggerMask

  The Trigger bit mask. This is the bit mask of the 8 timing signals.
- BInt32 triggerAnd

  The Trigger function is an AND rather than an OR.
- BInt32 triggerStore

  Store the trigger in the upper 8 data bits.
- BInt32 triggerSourceData

Use lower 32bits of data as trigger source rather than timing signals.

- BInt32 triggerStateEnable

  Trigger in state enable.

- BUInt32 triggerState

  Trigger state number.

7.117.1 Detailed Description

This class defines the parameters for a test data capture.

7.117.2 Constructor & Destructor Documentation

7.117.2.1 Tms::TestCaptureInfo::TestCaptureInfo (BUInt32 *psource* = BUInt32 ( ), BUInt32 *pclock* = BUInt32 ( ), BUInt32 *pstartTime* = BUInt32 ( ), BUInt32 *ppostTriggerDelay* = BUInt32 ( ), BUInt32 *ptriggerMask* = BUInt32 ( ), BInt32 *ptriggerAnd* = BInt32 ( ), BInt32 *ptriggerStore* = BInt32 ( ), BInt32 *ptriggerSourceData* = BInt32 ( ), BInt32 *ptriggerStateEnable* = BInt32 ( ), BUInt32 *ptriggerState* = BUInt32 ( ))

7.117.3 Member Data Documentation

7.117.3.1 BUInt32 Tms::TestCaptureInfo::clock

The Clock source.

7.117.3.2 BUInt32 Tms::TestCaptureInfo::postTriggerDelay

The delay, in clock cycles, after the trigger before capture starts.

7.117.3.3 BUInt32 Tms::TestCaptureInfo::source

The source data (0 - 3)

7.117.3.4 BUInt32 Tms::TestCaptureInfo::startTime

The start time in ms from CYCLE_START before trigger is activated.

7.117.3.5 BInt32 Tms::TestCaptureInfo::triggerAnd

The Trigger function is an AND rather than an OR.

7.117.3.6 BUInt32 Tms::TestCaptureInfo::triggerMask

The Trigger bit mask. This is the bit mask of the 8 timing signals.

7.117.3.7 BInt32 Tms::TestCaptureInfo::triggerSourceData

Use lower 32bits of data as trigger source rather than timing signals.

7.117.3.8 BUInt32 Tms::TestCaptureInfo::triggerState

Trigger state number.

7.117.3.9 BInt32 Tms::TestCaptureInfo::triggerStateEnable

Trigger in state enable.

7.117.3.10 BInt32 Tms::TestCaptureInfo::triggerStore

Store the trigger in the upper 8 data bits.

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

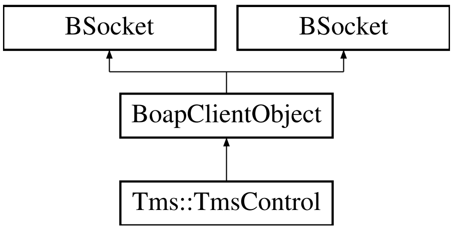
- TmsD.h
- TmsD.cc

7.118 Tms::TmsControl Class Reference

This interface provides functions to control, test and get statistics from the TMS as a whole.

`#include <TmsC.h>`

Inheritance diagram for Tms::TmsControl:



Public Member Functions

- TmsControl (BString name="")
- BError getVersion (BString &version)

  Gets the software version.
- BError setProcessPriority (BUInt32 priority)

  Sets the priority of the process servicing this service.
- BError init ()

  Initialises the system including resetting all of the PUPE engines firmware. The call will return an error object indicating success or an error condition as appropriate.
- BError configure (ConfigInfo configInfo)

  Configure the system for use. This includes mapping the individual physical PickUp channels to logical pickup channels.
- BError getConfiguration (ConfigInfo &configInfo)

  Get the current configuration.
- BError setControlInfo (CycleParam params)

  Sets the control information for the cycle type given. The parameters for the processing cycle are passed, this includes the Phase and State table information. The call will return an error object indicating success or an error.
- BError getControlInfo (BString cycleType, BUInt32 ring, BUInt32 puChannel, CycleParam &params)

  Gets the control information for the cycle type and puChannel number given. The call will return an error object indicating success or an error.
- BError delControlInfo (BString cycleType, BUInt32 ring, BUInt32 puChannel)

  Deletes the control information for the cycle type and puChannel number given. The call will return an error object indicating success or an error.

- BError getControlList (BList< CycleParamItem > &itemList)

    Gets the list of Cycle Parameters present in the system.
- BError setNextCycle (BUInt32 cycleNumber, BString cycleType)

    Sets the cycle number and type for the next processing cycle. The call will return an error object indicating success or an error condition as appropriate. This should be called at least 10ms before the next CYCLE_START event.
- BError test (BList< BError > &errors)

    Performs a basic test of the system returning a list of errors. The call will return an error object indicating success or an error condition as appropriate.
- BError getStatus (BList< NameValue > &statusList)

    Returns the current status of the system. This information includes the number of Pick-Up's present and their individual status.
- BError getStatistics (BList< NameValue > &statsList)

    Returns a list of the statistic values as name/value pairs. The call will return an error object indicating success or an error condition as appropriate.
- BError getPuChannel (BUInt32 puChannel, PuChannel &puPhysChannel)

    Returns the physical module/Pupe/Channel number given a logical PickUp id. This can be used so that the individual PickUps test functions can be accessed etc.
- BError setSimulation (Simulation simulation)

    Sets overall simulation modes.
- BError getSimulation (Simulation &simulation)

    Gets current simulation modes.
- BError setTestMode (PuChannel puPhysChannel, BUInt32 testOutput, BUInt32 timingDisableMask)

    The signal source for the digital test output connector. 0: None, 1: FrefLocal. The timingDisableMask bit mask defines which of the timing inputs should be disabled. If a timing input is disabled it can be still operated by software command.
- BError setTimingSignals (PuChannel puPhysChannel, BUInt32 timingSignals)

    This function sets the given timing signals to the values as defined in the timingSignals bit array.
- BError captureDiagnostics (PuChannel puPhysChannel, TestCaptureInfo captureInfo, BArray< BUInt64 > &data)

    This function will capture the diagnostics.
- BError setTestData (PuChannel puPhysChannel, BInt32 on, BArray< BUInt32 > data)

    This function will set a PU channel to sample data from memory rather than the ADC's.
- BError setPupeConfig (PuChannel puPhysChannel, PupeConfig pupeConfig)

    Sets special PUPE configuration for test purposes.
- BError getPupeConfig (PuChannel puPhysChannel, PupeConfig &pupeConfig)

    Gets special PUPE configuration for test purposes.
- BError puServerStarted (BUInt32 number)

    A TmsPuServer has started.

Additional Inherited Members

7.118.1 Detailed Description

This interface provides functions to control, test and get statistics from the TMS as a whole.

7.118.2 Constructor & Destructor Documentation

7.118.2.1 Tms::TmsControl::TmsControl (BString *name* = "")

7.118.3 Member Function Documentation

7.118.3.1 BError Tms::TmsControl::captureDiagnostics (PuChannel *puPhysChannel,* TestCaptureInfo *captureInfo,* BArray< BUInt64 > & *data*)

This function will capture the diagnostics.

7.118.3.2 BError Tms::TmsControl::configure (ConfigInfo *configInfo*)

Configure the system for use. This includes mapping the individual physical PickUp channels to logical pickup channels.

Parameters

configInfo	The channel mapping table.

This function configures the logical to physical channel mapping table.

7.118.3.3 BError Tms::TmsControl::delControlInfo (BString *cycleType,* BUInt32 *ring,* BUInt32 *puChannel*)

Deletes the control information for the cycle type and puChannel number given. The call will return an error object indicating success or an error.

Parameters

cycleType	This string defines the cycle type to delete from the database.
puChannel	This defines the specific channel to delete the information for. 0 means all channels.

This function will delete a set of Cycle parameters from the TMS's Cycle parameter database.

7.118.3.4 BError Tms::TmsControl::getConfiguration (ConfigInfo & *configInfo*)

Get the current configuration.

Parameters

configInfo	The channel mapping table that is filled in with the current curent channel mapping.

This function reads the current logical to physical channel mapping table.

7.118.3.5 BError Tms::TmsControl::getControlInfo (BString *cycleType,* BUInt32 *ring,* BUInt32 *puChannel,* CycleParam & *params*)

Gets the control information for the cycle type and puChannel number given. The call will return an error object indicating success or an error.

Parameters

cycleType	This string defines the cycle type for which to get the information.
puChannel	This defines the channel to get the information for. 0 means all channels.
params	The resuting cycle parameters are placed in this object.

This function reads back the set of Cycle parameters for the given cycle type and channel number. Normall the same cycle parameters are used for all PUPE engines. In this case setting the puChannel to 0 reads the Cycle Parameters that are being used on all channels. If a specific channel has other parameters the puChannel variable can be set to the appropriate channel number to get its particular settings.

7.118.3.6 BError Tms::TmsControl::getControlList (BList< CycleParamItem > & *itemList*)

Gets the list of Cycle Parameters present in the system.

Parameters

itemList	The list of CycleType information is returned.

This function will return a list of entries describing the Cycle Paramter sets present in the TMS database.

7.118.3.7 BError Tms::TmsControl::getPuChannel (BUInt32 *puChannel,* PuChannel & *puPhysChannel*)

Returns the physical module/Pupe/Channel number given a logical PickUp id. This can be used so that the individual PickUps test functions can be accessed etc.

Parameters

puChannel	The logical channel number.
puPhysChannel	The physical channel identifier is returned in this variable.

This function is given a logical pick-up channel number. It will return the physical module, pupe number and pupe channle that has been allocated to this channel.

7.118.3.8 BError Tms::TmsControl::getPupeConfig (PuChannel *puPhysChannel,* PupeConfig & *pupeConfig*)

Gets special PUPE configuration for test purposes.

Parameters

puPhysChannel	The physical channel identifier.
pupeConfig	The returned configuration parameters.

This function returns the current configuration of the given channel.

7.118.3.9 BError Tms::TmsControl::getSimulation (Simulation & *simulation*)

Gets current simulation modes.

7.118.3.10 BError Tms::TmsControl::getStatistics (BList< NameValue > & *statsList*)

Returns a list of the statistic values as name/value pairs. The call will return an error object indicating success or an error condition as appropriate.

Parameters

statsList	The statistics list is placed in this list object.

This function gets the statistics values from the TMS system. It returns a list of name/value pairs.

7.118.3.11 BError Tms::TmsControl::getStatus (BList< NameValue > & *statusList*)

Returns the current status of the system. This information includes the number of Pick-Up's present and their individual status.

Parameters

statusList	The list of status items is placed in this list object.

This function gets the status of the TMS system. It returns a list of name/value pairs.

7.118.3.12 BError Tms::TmsControl::getVersion (BString & *version*)

Gets the software version.

Parameters

version	A string variable filled in with the version number string.

7.118.3.13 BError Tms::TmsControl::init ( )

Initialises the system including resetting all of the PUPE engines firmware. The call will return an error object indicating success or an error condition as appropriate.

This function restarts the TMS system. It re-initialises each of the TmsPuServer processes running on the Module Controllers and reboots each of the PUPE boards from scratch loading the current FPGA firmware. All errors and statistics values are reset.

7.118.3.14 BError Tms::TmsControl::puServerStarted (BUInt32 *number*)

A TmsPuServer has started.

Parameters

number	The number of the PuServer started.

This is an internal function called by the TmsPuServer processes to indicate to the TmsServer that they have just started running and are present in the system. The TmsServer will initialise the appropriate tmsPuServer program and its individual PUPE engines on receipt of this call.

7.118.3.15 BError Tms::TmsControl::setControlInfo (CycleParam *params*)

Sets the control information for the cycle type given. The parameters for the processing cycle are passed, this includes the Phase and State table information. The call will return an error object indicating success or an error.

Parameters

params	Cycle information parameters (state/phase table information).

This function over-writes or adds an entry in the Cycle Parameter database. The Cycle Parameters define the setting for each processing cycle including the state and phase tables for the PUPE FPGA engines.

7.118.3.16 BError Tms::TmsControl::setNextCycle (BUInt32 *cycleNumber*, BString *cycleType*)

Sets the cycle number and type for the next processing cycle. The call will return an error object indicating success or an error condition as appropriate. This should be called at least 10ms before the next CYCLE_START event.

Parameters

cycleNumber	This is the next cycle number. This should be an incrementing 32bit unsigned value.
cycleType	This is a string defining the cycle type for the next cycle.

This call configures the TMS system for the next processing cycle. It defines the cycle number that will be used to tag data captured during the cycle and it defines the type of machine cycle. The cycleType is used to lookup the appropriate state/phase table information to use in the FPGA's. The call should be made at least 10ms before the CYCLE_START event for the cycle it refers to. This gives time for the FPGA's to be loaded with the appropriate state/phase table information. As the function is time critical, the communications channel should be set to a high priority using the setPriority() call and the processing threads priority should be set to high using the setProcessPriority() call. The call will return the error: "ErrorCycleNumber", "The next cycle has already started" if the call has not completed before the CYCLE_START event. All client data reads, for this cycle, will also return this error message.

7.118.3.17 BError Tms::TmsControl::setProcessPriority (BUInt32 *priority*)

Sets the priority of the process servicing this service.

Parameters

priority	This is the priority of the process. It can be set to one of: PriorityLow, PriorityNormal, PriorityHigh.

7.118.3.18 BError Tms::TmsControl::setPupeConfig (PuChannel *puPhysChannel,* PupeConfig *pupeConfig*)

Sets special PUPE configuration for test purposes.

Parameters

puPhysChannel	The physical channel identifier.
pupeConfig	The configuration parameters to use.

This functions sets up some special configuration parameters for the PUPE channel. It is used mainly for diagnostics and test purposes. The main settings it can affect are: The ADC Clock sources PLL synchronisation, internal timing for the digtital timing signals and the enabling/dissabling of the BLR algorithem.

7.118.3.19 BError Tms::TmsControl::setSimulation (Simulation *simulation*)

Sets overall simulation modes.

7.118.3.20 BError Tms::TmsControl::setTestData (PuChannel *puPhysChannel,* BInt32 *on,* BArray< BUInt32 > *data*)

This function will set a PU channel to sample data from memory rather than the ADC's.

Parameters

puPhysChannel	The physical channel identifier.
on	Boolean to enable the internal data source. 0 is off, 1 is on.
data	The array of 32bit data values to use as the FREF,Sigma,DeltaX and DeltaY test signal.

This call loads the PUPE systems test data SDRAM with the data passed in the data array. It then sets up the individual channel to sources its FREF, Sigma, DeltaX and DelatY signals from the test SDRAM. The data source should have a multiple of 2 samples. The "on" parameter is used to enable or disable the individual channels inputs from this test data SDRAM.

7.118.3.21 BError Tms::TmsControl::setTestMode (PuChannel *puPhysChannel,* BUInt32 *testOutput,* BUInt32 *timingDisableMask*)

The signal source for the digital test output connector. 0: None, 1: FrefLocal. The timingDisableMask bit mask defines which of the timing inputs should be disabled. If a timing input is disabled it can be still operated by software command.

Parameters

puPhysChannel	The physical channel identifier.
testOutput	The signal to output on the test output. 0 is FREF any other value is undefined at the moment.
timingDisableMask	This 8 bit mask defines which of the timing input signals are disabled.

This function sets up a particular pick-up channel's digital test output source and allows the channels input timing signals to be set to a software driven mode rather than taken from the hardware timing inputs. The timing mask bits are: 7 - FREF, 6 - HCHANGE, 5 - INJECTION, 4 - CAL_STOP, 3 - CAL_START, 2 - CYCLE_STOP, 1 - CYCLE_START, 0 - SYSCLOCK

7.118.3.22 BError Tms::TmsControl::setTimingSignals (PuChannel *puPhysChannel,* BUInt32 *timingSignals*)

This function sets the given timing signals to the values as defined in the timingSignals bit array.

Parameters

puPhysChannel	The physical channel identifier.
timingSignals	The 8 bit mask defining the state of the software driven timing signals.

If the setTestMode() function had been used to "enable" particular timing signals to be driven by software, then this function can be used to set/reset particular timing signals for the pick-up channel given. The timing signals bits are: 7 - FREF, 6 - HCHANGE, 5 - INJECTION, 4 - CAL_STOP, 3 - CAL_START, 2 - CYCLE_STOP, 1 - CYCLE_START, 0 - SYSCLOCK

7.118.3.23 BError Tms::TmsControl::test (BList< BError > & *errors*)

Performs a basic test of the system returning a list of errors. The call will return an error object indicating success or an error condition as appropriate.

Parameters

errors	The list of errors is placed in this list object.

This function will perform a test of the TMS system. It will report each test performed and the status of the test in the BError object. A status value of 0 indicates all was Ok, any other value is an error where the number indicates the error. A string gives the test name and the Ok or error condition as a string.

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

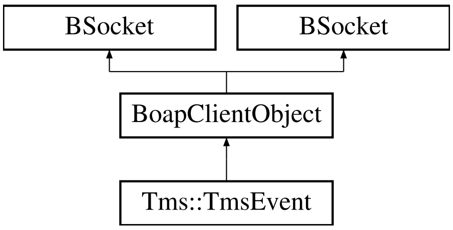
- TmsC.h
- TmsC.cc
- tmsFunctions.dox

7.119 Tms::TmsEvent Class Reference

This interface provides functions for events to be sent to clients from the TMS as a whole.

```
#include <TmsC.h>
```

Inheritance diagram for Tms::TmsEvent:



Public Member Functions

- TmsEvent (BString name="")
- BError errorEvent (BUInt32 cycleNumber, BError error)

  This event function gets called on a system error. The errorEvent object contains and error number and string describing the error. The getStatus() call can be used to fetch further information.
- BError cycleStartEvent (BUInt32 cycleNumber)

  This event function gets called on the CYCLE_START event with the cycle number about to be processed.
- BError cycleStopEvent (BUInt32 cycleNumber)

  This event function gets called on the CYCLE_END event with the cycle number completed.
- BError dataEvent (DataInfo dataInfo)

  This event function gets called when some requested data becomes available. The DataInfo object contains information on the data. The getData() call can be used to fetch the actual data.

Additional Inherited Members

7.119.1 Detailed Description

This interface provides functions for events to be sent to clients from the TMS as a whole.

7.119.2 Constructor & Destructor Documentation

7.119.2.1 Tms::TmsEvent::TmsEvent (BString *name* = "")

7.119.3 Member Function Documentation

7.119.3.1 BError Tms::TmsEvent::cycleStartEvent (BUInt32 *cycleNumber*)

This event function gets called on the CYCLE_START event with the cycle number about to be processed.

7.119.3.2 BError Tms::TmsEvent::cycleStopEvent (BUInt32 *cycleNumber*)

This event function gets called on the CYCLE_END event with the cycle number completed.

7.119.3.3 BError Tms::TmsEvent::dataEvent (DataInfo *dataInfo*)

This event function gets called when some requested data becomes available. The DataInfo object contains information on the data. The getData() call can be used to fetch the actual data.

7.119.3.4 BError Tms::TmsEvent::errorEvent (BUInt32 *cycleNumber,* BError *error*)

This event function gets called on a system error. The errorEvent object contains and error number and string describing the error. The getStatus() call can be used to fetch further information.

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

- TmsC.h
- TmsC.cc

7.120 Tms::TmsEventServerList Class Reference

```
#include <TmsEventServerList.h>
```

Public Member Functions

- TmsEventServerList ()
- ∼TmsEventServerList ()
- BError append (BString name)
- BError del (BString name)
- BError errorEvent (BUInt32 cycleNumber, BError error)

    This event function gets called on a system error. The errorEvent object contains and error number and string describing the error. The getStatus() call can be used to fetch further information.
- BError cycleStartEvent (BUInt32 cycleNumber)

    This event function gets called on the CYCLE_START event with the cycle number about to be processed.
- BError cycleStopEvent (BUInt32 cycleNumber)

This event function gets called on the CYCLE_END event with the cycle number completed.

- BError dataEvent (DataInfo dataInfo)

  This event function gets called when some requested data becomes available. The DataInfo object contains information on the data. The getData() call can be used to fetch the actual data.

Private Attributes

- BMutex olock
- BList< TmsEvent * > oeventServers

7.120.1 Constructor & Destructor Documentation

7.120.1.1 Tms::TmsEventServerList::TmsEventServerList ( )

7.120.1.2 Tms::TmsEventServerList::∼TmsEventServerList ( )

7.120.2 Member Function Documentation

7.120.2.1 BError Tms::TmsEventServerList::append (BString *name*)

7.120.2.2 BError Tms::TmsEventServerList::cycleStartEvent (BUInt32 *cycleNumber*)

This event function gets called on the CYCLE_START event with the cycle number about to be processed.

7.120.2.3 BError Tms::TmsEventServerList::cycleStopEvent (BUInt32 *cycleNumber*)

This event function gets called on the CYCLE_END event with the cycle number completed.

7.120.2.4 BError Tms::TmsEventServerList::dataEvent (DataInfo *dataInfo*)

This event function gets called when some requested data becomes available. The DataInfo object contains information on the data. The getData() call can be used to fetch the actual data.

7.120.2.5 BError Tms::TmsEventServerList::del (BString *name*)

7.120.2.6 BError Tms::TmsEventServerList::errorEvent (BUInt32 *cycleNumber,* BError *error*)

This event function gets called on a system error. The errorEvent object contains and error number and string describing the error. The getStatus() call can be used to fetch further information.

7.120.3 Member Data Documentation

7.120.3.1 BList<TmsEvent*> Tms::TmsEventServerList::oeventServers [private]

7.120.3.2 BMutex Tms::TmsEventServerList::olock [private]

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

- TmsEventServerList.h
- TmsEventServerList.cc

7.121 Tms::TmsPhase Union Reference

The Tms Phase Table Entry.

```
#include <TmsLib.h>
```

Public Attributes

- struct {
    unsigned int lo1:1
    unsigned int blr:1
    unsigned int gate:1
    unsigned int lo2:1
    unsigned int spare:2
    unsigned int meanFilter1:1
    unsigned int meanFilter2:1
  };
- unsigned char value

7.121.1 Detailed Description

The Tms Phase Table Entry.

7.121.2 Member Data Documentation

7.121.2.1 struct { ... }

7.121.2.2 unsigned int Tms::TmsPhase::blr

7.121.2.3 unsigned int Tms::TmsPhase::gate

7.121.2.4 unsigned int Tms::TmsPhase::lo1

7.121.2.5 unsigned int Tms::TmsPhase::lo2

7.121.2.6 unsigned int Tms::TmsPhase::meanFilter1

7.121.2.7 unsigned int Tms::TmsPhase::meanFilter2

7.121.2.8 unsigned int Tms::TmsPhase::spare

7.121.2.9 unsigned char Tms::TmsPhase::value

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

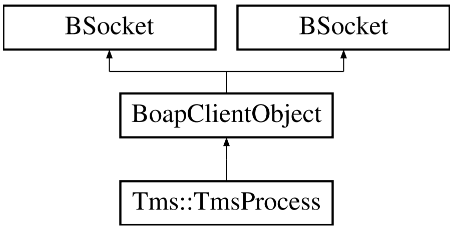
- TmsLib.h

7.122 Tms::TmsProcess Class Reference

This interface provides functions to capture data from the TMS as a whole.

```
#include <TmsC.h>
```

Inheritance diagram for Tms::TmsProcess:



Public Member Functions

- TmsProcess (BString name="")
- BError getVersion (BString &version)

    Gets the software version.
- BError getCycleInfo (BUInt32 &cycleNumber, BString &cycleType)

    Gets the current cycle number and type.
- BError getCycleInformation (BUInt32 cycleNumber, CycleInformation &cycleInformation)

    Gets information on given cycle number.
- BError getCycleTypeInformation (BString cycleType, CycleTypeInformation &cycleTypeInformation)

    Gets information on given cycle Type.
- BError getData (DataInfo dataInfo, Data &data)

    This function returns a set of data from the data present in the data cache or directly from the Pick-Up processing engines. The DataInfo object describes the data required. The call will return the required data along with an error object indicating success or an error condition as appropriate. The call will block until data is ready.
- BError addEventServer (BString name)

    This call adds an event server to call on events such as the "dataEvent" generated by the requestData(0 call as well as error events. The Client will use this to notify the TmsServer of its local TmsEvent object.
- BError requestData (DataInfo dataInfo)

    This adds a request for some data. The DataInfo object defines the data required. This request can be made at any time. If the data is present in cache the data will be available immediately, if not the system will await the data from a subsequent processing cycle. When the data is available a "data" event will be sent to the client. Not that it is not necessary to use requestData. The client can call getData() directly although this call will block until the data is actually ready.

Additional Inherited Members

7.122.1 Detailed Description

This interface provides functions to capture data from the TMS as a whole.

7.122.2 Constructor & Destructor Documentation

7.122.2.1 Tms::TmsProcess::TmsProcess (BString *name* = " ")

Parameters

name	The name of the TmsProcess BOAP object to connect to.

The BOAP object name has the general form: "//HostName/ObjectName"

7.122.3 Member Function Documentation

7.122.3.1 BError Tms::TmsProcess::addEventServer (BString *name*)

This call adds an event server to call on events such as the "dataEvent" generated by the requestData(0 call as well as error events. The Client will use this to notify the TmsServer of its local TmsEvent object.

Parameters

name	The BOAP object name to add.

Adds an event server that gets called on certain TmsServer events such as data ready, CYCLE_START, CYCLE_-STOP and errors.

7.122.3.2 BError Tms::TmsProcess::getCycleInfo (BUInt32 & *cycleNumber,* BString & *cycleType*)

Gets the current cycle number and type.

Parameters

cycleNumber	The current cycle number is returned here
cycleType	The current cycle type is returned here.

This function returns the current TMS cycle number being processed and the type of the cycle.

7.122.3.3 BError Tms::TmsProcess::getCycleInformation (BUInt32 *cycleNumber,* CycleInformation & *cycleInformation*)

Gets information on given cycle number.

Parameters

cycleNumber	The current cycle number to get information on
cycleInformation	The returned cycle information

This function provides information on the given cycle. It interrogates the first PUPE channel and returns the Cycle information based on the internal state of that channel. The main information returned includes the cycle type and a list of all of the Cycle Periods captured and the times and ammount of data in each.

7.122.3.4 BError Tms::TmsProcess::getCycleTypeInformation (BString *cycleType,* CycleTypeInformation & *cycleTypeInformation*)

Gets information on given cycle Type.

7.122.3.5 BError Tms::TmsProcess::getData (DataInfo *dataInfo,* Data & *data*)

This function returns a set of data from the data present in the data cache or directly from the Pick-Up processing engines. The DataInfo object describes the data required. The call will return the required data along with an error object indicating success or an error condition as appropriate. The call will block until data is ready.

Parameters

dataInfo	Information on the type of data required.
data	The raw data is returned in this object.

This is the main user function used by clients of the TMS system. It is used to return portions of the acquired data. The fields of the dataInfo parameter define which data is required and are defined in the DataInfo class documentation.

The call will check to see if the data for the cycle number requested is still present in the PUPE memory. The PUPE memory has enough storage for about 3 seconds worth of data (3 processing cycles). If the data has gone the call will return the error "ErrorDataGone". If the system has not processed the requested cycle, but will do so within 256 seconds, the call will block awaiting the data.

If the channel number is given as 0 the call will interrogate each of the Pick-Up channels and return the combined data from all of them. Note that this could take significant time and may not be possible if the parameter numValues is large. Within the Data structure returned there is an array of error values, one per channel. If an error occurs on any set of the channels the call will return the first error that occurred and the complete list of errors in the errors array. The actual data will be returned for all channels that did not have an error. Those channels that had anb error will have data values of 0 returned.

If the bunch number is given as 0, then the system will return the data for all of the bunches.

The data will be returned in the following order, where B - Bunch, C - Channel:

[C1.B1, C1.B2, C1.B3, C1.B4], [C1.B1, C1.B2, C1.B3, C1.B4], ... [C2.B1, C2.B2, C2.B3, C2.B4], [C2.B1, C2.B2, C2.B3, C2.B4], ...

That is the data is ordered by bunch, then sample, then channel. See the TMS Software documentation manual for more details of this functions operation.

7.122.3.6 BError Tms::TmsProcess::getVersion (BString & *version*)

Gets the software version.

Parameters

version	A string variable filled in with the version number string.

7.122.3.7 BError Tms::TmsProcess::requestData (DataInfo *dataInfo*)

This adds a request for some data. The DataInfo object defines the data required. This request can be made at any time. If the data is present in cache the data will be available immediately, if not the system will await the data from a subsequent processing cycle. When the data is available a "data" event will be sent to the client. Not that it is not necessary to use requestData. The client can call getData() directly although this call will block until the data is actually ready.

Parameters

dataInfo	Information on the type of data required.

This calls sets up a request for data. The dataInfo parameter works in the same manner as the "getData" call, defining the portion of data required. This call will return immediatly. Assuming the client has informed the TMS system of an event server object using the "addEventServer" call, then the client will receive the "dataEvent" event when the data become available. The client can then fetch the data using the conventional "getData" call. In the current version of the software the "requestData" call simply sends a message when the data for the requested cycle is ready. In future implementations the TMS system could actually fetch the data automatically from the PUPE boards and store it in memory ready for later retrieval by the getData call.

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

- TmsC.h
- TmsC.cc
- tmsFunctions.dox

7.123 Tms::TmsState Union Reference

The Tms State entry.

```
#include <TmsLib.h>
```

Public Attributes

- struct {
    unsigned int acquireData:1
    unsigned int pllReference1:1
    unsigned int pllReference2:1
    unsigned int pllFeedbackSelect:1
    unsigned int pllLO1FromAddress:1
    unsigned int pllLO2FromAddress:1

```
        unsigned int bit6:1
        unsigned int bit7:1
        unsigned int cycleStop:4
        unsigned int calStop:4
        unsigned int calStart:4
        unsigned int injection:4
        unsigned int hchange:4
        unsigned int delay:4
    };
```

- unsigned int value

7.123.1 Detailed Description

The Tms State entry.

7.123.2 Member Data Documentation

7.123.2.1 struct { ... }

7.123.2.2 unsigned int Tms::TmsState::acquireData

7.123.2.3 unsigned int Tms::TmsState::bit6

7.123.2.4 unsigned int Tms::TmsState::bit7

7.123.2.5 unsigned int Tms::TmsState::calStart

7.123.2.6 unsigned int Tms::TmsState::calStop

7.123.2.7 unsigned int Tms::TmsState::cycleStop

7.123.2.8 unsigned int Tms::TmsState::delay

7.123.2.9 unsigned int Tms::TmsState::hchange

7.123.2.10 unsigned int Tms::TmsState::injection

7.123.2.11 unsigned int Tms::TmsState::pllFeedbackSelect

7.123.2.12 unsigned int Tms::TmsState::pllLO1FromAddress

7.123.2.13 unsigned int Tms::TmsState::pllLO2FromAddress

7.123.2.14 unsigned int Tms::TmsState::pllReference1

7.123.2.15 unsigned int Tms::TmsState::pllReference2

7.123.2.16 unsigned int Tms::TmsState::value

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

- TmsLib.h

Chapter 8

File Documentation

8.1 /src/cern/tms/beam/libBeam/BArray.h File Reference

```
#include <BTypes.h>
#include <vector>
#include <algorithm>
```

Classes

- class BArray< T >

Macros

- #define BArrayLoop(list, i) for(BUInt i = 0; i < list.number(); i++)

8.1.1 Macro Definition Documentation

8.1.1.1 #define BArrayLoop(*list,* *i*) for(**BUInt** i = 0; i < list.number(); i++)

8.2 /src/cern/tms/beam/libBeam/BAtomic.h File Reference

```
#include <BTypes.h>
```

Classes

- class BAtomic< Type >

  BAtomic class.

Typedefs

- typedef BAtomic< BInt32 > BAtomicInt32
- typedef BAtomic< BInt64 > BAtomicInt64
- typedef BAtomic< BUInt32 > BAtomicUInt32
- typedef BAtomic< BUInt64 > BAtomicUInt64

8.2.1 Typedef Documentation

8.2.1.1 typedef **BAtomic**<**BInt32**> **BAtomicInt32**

8.2.1.2 typedef **BAtomic**<**BInt64**> **BAtomicInt64**

8.2.1.3 typedef **BAtomic**<**BUInt32**> **BAtomicUInt32**

8.2.1.4 typedef **BAtomic**<**BUInt64**> **BAtomicUInt64**

8.3 /src/cern/tms/beam/libBeam/BAtomicCount.h File Reference

```
#include <bits/atomicity.h>
```

Classes

- class BAtomicCount

  BAtomicCount class.

8.4 /src/cern/tms/beam/libBeam/BBuffer.cpp File Reference

```
#include <stdlib.h>
#include <memory.h>
#include <BBuffer.h>
#include <BEndian.h>
#include <BTimeStamp.h>
#include <BComplex.h>
```

Variables

- const int roundSize = 256

8.4.1 Variable Documentation

8.4.1.1 const int roundSize = 256

8.5 /src/cern/tms/beam/libBeam/BBuffer.h File Reference

```
#include <BTypes.h>
#include <BString.h>
#include <BError.h>
#include <BComplex.h>
#include <BEndian.h>
```

Classes

- class BBuffer
- class BBufferStore

Macros

- #define BBigEndian 0

8.5.1 Macro Definition Documentation

8.5.1.1 #define BBigEndian 0

8.6 /src/cern/tms/beam/libBeam/BComms.cpp File Reference

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

8.7 /src/cern/tms/beam/libBeam/BComms.h File Reference

```
#include <BTypes.h>
#include <BEvent.h>
#include <BError.h>
```

Classes

- class BComms

8.8 /src/cern/tms/beam/libBeam/BComplex.h File Reference

```
#include <BTypes.h>
#include <complex>
#include <algorithm>
```

Typedefs

- typedef std::complex< double > BComplex
- typedef std::complex< float > BComplex32
- typedef std::complex< double > BComplex64

8.8.1 Typedef Documentation

8.8.1.1 typedef std::complex<double> BComplex

This is based on the Standard C++ library complex class and has all of the functionality of that class.

8.8.1.2 typedef std::complex<float> BComplex32

8.8.1.3 typedef std::complex<double> BComplex64

8.9 /src/cern/tms/beam/libBeam/BCond.cpp File Reference

```
#include <BCond.h>
#include <sys/time.h>
#include <stdio.h>
```

8.10 /src/cern/tms/beam/libBeam/BCond.h File Reference

```
#include <pthread.h>
```

Classes

- class BCond

8.11 /src/cern/tms/beam/libBeam/BCondInt.cpp File Reference

```
#include <BCondInt.h>
#include <sys/time.h>
#include <stdio.h>
#include <errno.h>
```

Functions

- static struct timespec getTimeout (uint32_t timeOutUs)

8.11.1 Function Documentation

8.11.1.1 static struct timespec getTimeout (uint32_t *timeOutUs*) [static]

8.12 /src/cern/tms/beam/libBeam/BCondInt.h File Reference

```
#include <BTypes.h>
#include <pthread.h>
```

Classes

- class BCondInt

  Thread conditional value.
- class BCondValue

  Thread conditional value.
- class BCondBool

  Thread conditional boolean.
- class BCondWrap
- class BCondResource

  Resource lock.

8.13 /src/cern/tms/beam/libBeam/BConfig.cpp File Reference

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

8.14 /src/cern/tms/beam/libBeam/BConfig.h File Reference

```
#include <BDict.h>
#include <BFile.h>
#include <BMutex.h>
```

Classes

- class BConfig

  This class implements the configuration file access.

8.15 /src/cern/tms/beam/libBeam/BCrc16.cpp File Reference

```
#include <BCrc16.h>
```

Functions

- BUInt16 bcrc16 (void ∗buf, BUInt16 len)

Variables

- static const BUInt8 table_crc_hi []
- static const BUInt8 table_crc_lo []

8.15.1 Function Documentation

8.15.1.1 BUInt16 bcrc16 (void ∗ *buf,* BUInt16 *len*)

8.15.2 Variable Documentation

8.15.2.1 const BUInt8 table_crc_hi[ ] [static]

Initial value:

```
= {
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
    0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
    0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
    0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
    0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
    0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
    0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
    0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
    0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
    0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
}
```

8.15.2.2 const BUInt8 table_crc_lo[ ] [static]

Initial value:

```
= {
    0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06,
    0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD,
    0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
    0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,
    0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4,
    0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
    0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3,
    0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
    0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
    0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29,
    0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED,
    0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
    0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60,
    0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67,
    0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
    0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
    0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E,
    0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
    0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71,
    0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92,
    0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
    0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B,
    0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B,
    0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
    0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42,
    0x43, 0x83, 0x41, 0x81, 0x80, 0x40
}
```

8.16 /src/cern/tms/beam/libBeam/BCrc16.h File Reference

```
#include <BTypes.h>
```

Functions

- BUInt16 bcrc16 (void ∗buf, BUInt16 len)

8.16.1 Function Documentation

8.16.1.1 BUInt16 bcrc16 (void ∗ *buf,* BUInt16 *len*)

8.17 /src/cern/tms/beam/libBeam/BDate.cpp File Reference

```
#include <BDate.h>
#include <sys/time.h>
```

Functions

- void toBString (BDate &v, BString &s)
- void fromBString (BString &s, BDate &v)

Variables

- static int mon_yday [2][13]

8.17.1 Function Documentation

8.17.1.1 void fromBString (BString & *s*, BDate & *v*)

8.17.1.2 void toBString (BDate & *v*, BString & *s*)

8.17.2 Variable Documentation

8.17.2.1 int mon_yday[2][13] [static]

Initial value:

```
= {
        { 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334, 365 },
        { 0, 31, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335, 366 }
}
```

8.18 /src/cern/tms/beam/libBeam/BDate.h File Reference

```
#include <stdint.h>
#include <BError.h>
```

Classes

- class BDate

Functions

- void toBString (BDate &v, BString &s)
- void fromBString (BString &s, BDate &v)

8.18.1 Function Documentation

8.18.1.1 void fromBString (BString & *s*, BDate & *v*)

8.18.1.2 void toBString (BDate & *v*, BString & *s*)

8.19 /src/cern/tms/beam/libBeam/BDebug.cpp File Reference

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <syslog.h>
#include <sys/time.h>
#include <stdarg.h>
#include <fcntl.h>
#include <execinfo.h>
#include <ctype.h>
#include <BDebug.h>
#include <errno.h>
#include <linux/unistd.h>
```

Macros

- #define BTRACE_SIZE 100

Functions

- void hd8 (void ∗data, unsigned int n)
- void hd8a (void ∗data, unsigned int n)
- void hda8 (void ∗data, unsigned int n)
- void hd32 (void ∗data, unsigned int n)
- void hda32 (void ∗data, unsigned int n)
- double getTime ()
- void setDebug (int d)
- void tprintf (int log, const char ∗fmt,...)
- pid_t gettid ()

Variables

- int bdebug
- const unsigned int STRBUF_SIZE = (64 ∗ 1024)

8.19.1 Macro Definition Documentation

8.19.1.1 #define BTRACE_SIZE 100

8.19.2 Function Documentation

8.19.2.1 pid_t gettid ( )

8.19.2.2 double getTime ( )

8.19.2.3 void hd32 (void ∗ *data,* unsigned int *n*)

8.19.2.4 void hd8 (void ∗ *data,* unsigned int *n*)

8.19.2.5 void hd8a (void ∗ *data,* unsigned int *n*)

8.19.2.6 void hda32 (void ∗ *data,* unsigned int *n*)

8.19.2.7 void hda8 (void ∗ *data,* unsigned int *n*)

8.19.2.8 void setDebug (int *d*)

8.19.2.9 void tprintf (int *log,* const char ∗ *fmt,* ...)

8.19.3 Variable Documentation

8.19.3.1 int bdebug

8.19.3.2 const unsigned int STRBUF_SIZE = (64 ∗ 1024)

8.20 /src/cern/tms/beam/libBeam/BDebug.h File Reference

```
#include <stdio.h>
#include <syslog.h>
#include <time.h>
```

Classes

- class BDebugBacktrace

Macros

- #define BDebug_STD 0x000001
- #define dprintf(level, fmt, a...)

  General debug functions.
- #define nprintf(fmt, a...) syslog(LOG_NOTICE, fmt, ##a)

  Warnings and errors logging.
- #define wprintf(fmt, a...) syslog(LOG_WARNING, fmt, ##a)
- #define eprintf(fmt, a...) syslog(LOG_ERR, fmt, ##a)

Functions

- void hd8 (void ∗data, unsigned int n)
- void hd8a (void ∗data, unsigned int n)
- void hda8 (void ∗data, unsigned int n)
- void hd32 (void ∗data, unsigned int n)
- void hds32 (void ∗data, unsigned int n)
- double getTime ()
- void setDebug (int debug)
- void tprintf (int log, const char ∗fmt,...)
- pid_t gettid ()

Variables

- int bdebug

8.20.1 Macro Definition Documentation

8.20.1.1 #define BDebug_STD 0x000001

8.20.1.2 #define dprintf(*level,* *fmt,* *a...*)

General debug functions.

8.20.1.3 #define eprintf(*fmt,* *a...*) syslog(LOG_ERR, fmt, ##a)

8.20.1.4 #define nprintf(*fmt,* *a...*) syslog(LOG_NOTICE, fmt, ##a)

Warnings and errors logging.

8.20.1.5 #define wprintf(*fmt,* *a...*) syslog(LOG_WARNING, fmt, ##a)

8.20.2 Function Documentation

8.20.2.1 pid_t gettid ( )

8.20.2.2 double getTime ( )

8.20.2.3 void hd32 (void ∗ *data,* unsigned int *n*)

8.20.2.4 void hd8 (void ∗ *data,* unsigned int *n*)

8.20.2.5 void hd8a (void ∗ *data,* unsigned int *n*)

8.20.2.6 void hda8 (void ∗ *data,* unsigned int *n*)

8.20.2.7 void hds32 (void ∗ *data,* unsigned int *n*)

8.20.2.8 void setDebug (int *debug*)

8.20.2.9 void tprintf (int *log,* const char ∗ *fmt,* *...*)

8.20.3 Variable Documentation

8.20.3.1 int bdebug

8.21 /src/cern/tms/beam/libBeam/BDict.cpp File Reference

```
#include <BDict.h>
```

Functions

- void toBString (const BDictString &v, BString &s)
- void fromBString (const BString &str, BDictString &v)
- BString bdictStringToString (const BDictString &dict)

8.21.1 Function Documentation

8.21.1.1 BString bdictStringToString (const BDictString & *dict*)

8.21.1.2 void fromBString (const BString & *str*, BDictString & *v*)

8.21.1.3 void toBString (const BDictString & *v*, BString & *s*)

8.22 /src/cern/tms/beam/libBeam/BDict.h File Reference

```
#include <BNameValue.h>
```

Classes

- class BDictItem< Type >

  Template based Dictionary class.
- class BDict< Type >

Typedefs

- typedef BDict< BString > BDictString

Functions

- void toBString (const BDictString &v, BString &s)
- void fromBString (const BString &s, BDictString &v)
- BString bdictStringToString (const BDictString &dict)

8.22.1 Typedef Documentation

8.22.1.1 typedef BDict<BString> BDictString

8.22.2 Function Documentation

8.22.2.1 BString bdictStringToString (const BDictString & *dict*)

8.22.2.2 void fromBString (const BString & *s*, BDictString & *v*)

8.22.2.3 void toBString (const BDictString & *v*, BString & *s*)

8.23 /src/cern/tms/beam/libBeam/BDictMap.h File Reference

```
#include <BString.h>
#include <map>
```

Classes

- class BDictMap< Value >

Typedefs

- typedef BDictMap< BString > BDictMapString

8.23.1 Typedef Documentation

8.23.1.1 typedef BDictMap<BString> BDictMapString

8.24 /src/cern/tms/beam/libBeam/BDir.cpp File Reference

```
#include <BDir.h>
#include <dirent.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
```

Functions

- static int wild (const dirent ∗e)

Variables

- static BString wildString

8.24.1 Function Documentation

8.24.1.1 static int wild (const dirent ∗ *e*) [static]

8.24.2 Variable Documentation

8.24.2.1 BString wildString [static]

8.25 /src/cern/tms/beam/libBeam/BDir.h File Reference

```
#include <BList.h>
#include <BString.h>
#include <BError.h>
#include <sys/stat.h>
```

Classes

- class BDir

  File system directory class.

8.26 /src/cern/tms/beam/libBeam/BDuration.cpp File Reference

```
#include <BDuration.h>
#include <sys/time.h>
```

8.27 /src/cern/tms/beam/libBeam/BDuration.h File Reference

```
#include <stdint.h>
#include <BError.h>
```

Classes

- class BDuration

8.28 /src/cern/tms/beam/libBeam/BEndian.cpp File Reference

```
#include <BEndian.h>
#include <memory.h>
```

Functions

- void bswap_copy (int swap, const void ∗src, void ∗dst, BUInt32 nBytes, const char ∗swapType)

8.28.1 Function Documentation

8.28.1.1 void bswap_copy (int *swap,* const void ∗ *src,* void ∗ *dst,* BUInt32 *nBytes,* const char ∗ *swapType*)

8.29 /src/cern/tms/beam/libBeam/BEndian.h File Reference

```
#include <BTypes.h>
#include <byteswap.h>
```

Macros

- #define htobe16(x) __bswap_16 (x)
- #define htole16(x) (x)
- #define be16toh(x) __bswap_16 (x)
- #define le16toh(x) (x)
- #define htobe32(x) __bswap_32 (x)
- #define htole32(x) (x)
- #define be32toh(x) __bswap_32 (x)
- #define le32toh(x) (x)
- #define htobe64(x) __bswap_64 (x)
- #define htole64(x) (x)
- #define be64toh(x) __bswap_64 (x)
- #define le64toh(x) (x)

Functions

- void bswap_p8 (const void ∗s, void ∗d)
- void bswap_p16 (const void ∗s, void ∗d)
- void bswap_p32 (const void ∗s, void ∗d)

- void bswap_p64 (const void ∗s, void ∗d)
- void bswap_copy (int swap, const void ∗src, void ∗dst, BUInt32 nBytes, const char ∗swapType)
- uint16_t htole (uint16_t v)
- int16_t htole (int16_t v)
- uint32_t htole (uint32_t v)
- int32_t htole (int32_t v)
- uint64_t htole (uint64_t v)
- int64_t htole (int64_t v)
- double htole (double v)
- float htole (float v)
- uint16_t htobe (uint16_t v)
- int16_t htobe (int16_t v)
- uint32_t htobe (uint32_t v)
- int32_t htobe (int32_t v)
- uint64_t htobe (uint64_t v)
- int64_t htobe (int64_t v)
- double htobe (double v)
- float htobe (float v)
- uint16_t letoh (uint16_t v)
- int16_t letoh (int16_t v)
- uint32_t letoh (uint32_t v)
- int32_t letoh (int32_t v)
- uint64_t letoh (uint64_t v)
- int64_t letoh (int64_t v)
- double letoh (double v)
- float letoh (float v)
- uint16_t betoh (uint16_t v)
- int16_t betoh (int16_t v)
- uint32_t betoh (uint32_t v)
- int32_t betoh (int32_t v)
- uint64_t betoh (uint64_t v)
- int64_t betoh (int64_t v)
- double betoh (double v)
- float betoh (float v)

8.29.1 Macro Definition Documentation

8.29.1.1 #define be16toh(*x*) __bswap_16 (x)

8.29.1.2 #define be32toh(*x*) __bswap_32 (x)

8.29.1.3 #define be64toh(*x*) __bswap_64 (x)

8.29.1.4 #define htobe16(*x*) __bswap_16 (x)

8.29.1.5 #define htobe32(*x*) __bswap_32 (x)

8.29.1.6 #define htobe64(*x*) __bswap_64 (x)

8.29.1.7 #define htole16(*x*) (x)

8.29.1.8 #define htole32(*x*) (x)

8.29.1.9 #define htole64(*x*) (x)

8.29.1.10 #define le16toh(*x*) (x)

8.29.1.11 #define le32toh(*x*) (x)

8.29.1.12 #define le64toh(*x*) (x)

8.29.2 Function Documentation

8.29.2.1 uint16_t betoh (uint16_t *v*) [inline]

8.29.2.2 int16_t betoh (int16_t *v*) [inline]

8.29.2.3 uint32_t betoh (uint32_t *v*) [inline]

8.29.2.4 int32_t betoh (int32_t *v*) [inline]

8.29.2.5 uint64_t betoh (uint64_t *v*) [inline]

8.29.2.6 int64_t betoh (int64_t *v*) [inline]

8.29.2.7 double betoh (double *v*) [inline]

8.29.2.8 float betoh (float *v*) [inline]

8.29.2.9 void bswap_copy (int *swap,* const void ∗ *src,* void ∗ *dst,* BUInt32 *nBytes,* const char ∗ *swapType*)

8.29.2.10 void bswap_p16 (const void ∗ *s,* void ∗ *d*) [inline]

8.29.2.11 void bswap_p32 (const void ∗ *s,* void ∗ *d*) [inline]

8.29.2.12 void bswap_p64 (const void ∗ *s,* void ∗ *d*) [inline]

8.29.2.13 void bswap_p8 (const void ∗ *s,* void ∗ *d*) [inline]

8.29.2.14 uint16_t htobe (uint16_t *v*) [inline]

8.29.2.15 int16_t htobe (int16_t *v*) [inline]

8.29.2.16 uint32_t htobe (uint32_t *v*) [inline]

8.29.2.17 int32_t htobe (int32_t *v*) [inline]

8.29.2.18 uint64_t htobe (uint64_t *v*) [inline]

8.29.2.19 int64_t htobe (int64_t *v*) [inline]

8.29.2.20 double htobe (double *v*) [inline]

8.29.2.21 float htobe (float *v*) [inline]

8.29.2.22 uint16_t htole (uint16_t *v*) [inline]

8.29.2.23 int16_t htole (int16_t *v*) [inline]

8.29.2.24 uint32_t htole (uint32_t *v*) [inline]

8.29.2.25 int32_t htole (int32_t *v*) [inline]

8.29.2.26 uint64_t htole (uint64_t *v*) [inline]

8.29.2.27 int64_t htole (int64_t *v*) [inline]

8.29.2.28 double htole (double *v*) [inline]

8.29.2.29 float htole (float *v*) [inline]

8.29.2.30 uint16_t letoh (uint16_t *v*) [inline]

8.29.2.31 int16_t letoh (int16_t *v*) [inline]

8.29.2.32 uint32_t letoh (uint32_t *v*) [inline]

8.29.2.33 int32_t letoh (int32_t *v*) [inline]

8.29.2.34 uint64_t letoh (uint64_t *v*) [inline]

8.29.2.35 int64_t letoh (int64_t *v*) [inline]

8.29.2.36 double letoh (double *v*) [inline]

8.29.2.37 float letoh (float *v*) [inline]

8.30 /src/cern/tms/beam/libBeam/BEntry.cpp File Reference

```
#include <ctype.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <BEntry.h>
```

8.31 /src/cern/tms/beam/libBeam/BEntry.h File Reference

```
#include <BList.h>
#include <BString.h>
```

Classes

- class BEntry

  Manipulate a name value pair.
- class BEntryList

  List of Entries. Where an entry is a name value pair.
- class BEntryFile

  File of Entries.

8.32 /src/cern/tms/beam/libBeam/BError.cpp File Reference

```
#include <BError.h>
```

8.33 /src/cern/tms/beam/libBeam/BError.h File Reference

```
#include <BString.h>
```

Classes

- class BError

  Error return class.

Enumerations

- enum BErrorNum {
  ErrorOk = 0, ErrorMisc = 1, ErrorWarning = 2, ErrorParam = 3,
  ErrorTimeout = 4, ErrorNotAvailable = 5, ErrorData = 6, ErrorChecksum = 7,
  ErrorOverrun = 8, ErrorUnderrun = 9, ErrorInit = 10, ErrorConfig = 11,
  ErrorNotImplemented = 12, ErrorResourceLimit = 13, ErrorEndOfFile = 14, ErrorFile = 15,
  ErrorFormat = 16, ErrorComms = 17, ErrorAccessDenied = 18, ErrorNoData = 19,
  ErrorEndOfData = 20, ErrorDataPresent = 21, ErrorAppBase = 64 }

8.33.1 Enumeration Type Documentation

8.33.1.1 enum **BErrorNum**

Enumerator

ErrorOk

ErrorMisc

ErrorWarning

ErrorParam

ErrorTimeout

ErrorNotAvailable

ErrorData

ErrorChecksum

ErrorOverrun

ErrorUnderrun

ErrorInit

ErrorConfig

ErrorNotImplemented

ErrorResourceLimit

ErrorEndOfFile

ErrorFile

ErrorFormat

ErrorComms

ErrorAccessDenied

ErrorNoData

ErrorEndOfData

ErrorDataPresent

ErrorAppBase

8.34 /src/cern/tms/beam/libBeam/BErrorTime.cpp File Reference

```
#include <BErrorTime.h>
```

8.35 /src/cern/tms/beam/libBeam/BErrorTime.h File Reference

```
#include <BString.h>
#include <BTimeStamp.h>
```

Classes

- class BErrorTime

  Error return class.

8.36 /src/cern/tms/beam/libBeam/BEvent.cpp File Reference

```
#include <BEvent.h>
#include <BPoll.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
```

8.37 /src/cern/tms/beam/libBeam/BEvent.h File Reference

```
#include <BTypes.h>
#include <BQueue.h>
```

Classes

- class BEvent
- class BEventPipe

  This class provides an interface for sending simple integer events via a pipe file descriptor.

Typedefs

- typedef BQueue< BEvent > BEventQueue

  This class provides an interface for sending simple integer events via a BQueue.

Enumerations

- enum BEventType { BEventTypeNone = 0 }

8.37.1 Typedef Documentation

8.37.1.1 typedef BQueue<BEvent> BEventQueue

This class provides an interface for sending simple integer events via a BQueue.

8.37.2 Enumeration Type Documentation

8.37.2.1 enum BEventType

Enumerator

BEventTypeNone

8.38 /src/cern/tms/beam/libBeam/BEvent1.cpp File Reference

```
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <BEvent1.h>
#include <BPoll.h>
```

8.39 /src/cern/tms/beam/libBeam/BEvent1.h File Reference

```
#include <stdint.h>
#include <BError.h>
```

Classes

- class BEvent1

  This class provides a base class for all event objects that can be sent over the events interface.
- class BEvent1Error
- class BEvent1Pipe

  This class provides a base interface for sending events via a pipe. This allows threads to send events that can be picked up by the poll system call.
- class BEvent1Int

  This class provides an interface for sending simple integer events via a file descriptor. This allows threads to send events that can be picked up by the poll system call.

Enumerations

- enum BEvent1Type { BEvent1TypeNone, BEvent1TypeInt, BEvent1TypeError }

8.39.1 Enumeration Type Documentation

8.39.1.1 enum BEvent1Type

Enumerator

BEvent1TypeNone

BEvent1TypeInt

BEvent1TypeError

8.40 /src/cern/tms/beam/libBeam/BFifo.h File Reference

```
#include <BTypes.h>
#include <BError.h>
#include <BMutex.h>
#include <BFifo.inc>
```

Classes

- class BFifo< Type >

8.41 /src/cern/tms/beam/libBeam/BFifo.inc File Reference

8.42 /src/cern/tms/beam/libBeam/BFifoCirc.cpp File Reference

```
#include <BFifoCirc.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/mman.h>
```

Macros

- #define dprintf(fmt, a...)

8.42.1 Macro Definition Documentation

8.42.1.1 #define dprintf(*fmt, a...*)

8.43 /src/cern/tms/beam/libBeam/BFifoCirc.h File Reference

```
#include <stdint.h>
#include <BError.h>
#include <BCondInt.h>
#include <BMutex.h>
#include <BFifoCirc.inc>
```

Classes

- class BFifoCircPos

  This class implements a pointer into the Fifo's circular buffer.
- class BFifoCirc< Type >

  This class implements a thread safe FIFO buffer.

8.44 /src/cern/tms/beam/libBeam/BFifoCirc.inc File Reference

8.45 /src/cern/tms/beam/libBeam/BFile.cpp File Reference

```
#include <stdarg.h>
#include <BFile.h>
#include <sys/stat.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
```

Macros

- #define STRBUF 10240

8.45.1 Macro Definition Documentation

8.45.1.1 #define STRBUF 10240

8.46 /src/cern/tms/beam/libBeam/BFile.h File Reference

```
#include <stdio.h>
#include <BTypes.h>
#include <BString.h>
#include <BError.h>
```

Classes

- class BFile

  File operations class.

8.47 /src/cern/tms/beam/libBeam/BFileCsv.cpp File Reference

```
#include <BFileCsv.h>
#include <errno.h>
```

8.48 /src/cern/tms/beam/libBeam/BFileCsv.h File Reference

```
#include <BFile.h>
```

Classes

- class BFileCsv

8.49 /src/cern/tms/beam/libBeam/BFileData.cpp File Reference

```
#include <BFileCsv.h>
#include <BFileData.h>
#include <errno.h>
```

8.50 /src/cern/tms/beam/libBeam/BFileData.h File Reference

```
#include <BError.h>
```

Classes

- class BFileData

8.51 /src/cern/tms/beam/libBeam/BList.h File Reference

```
#include <BList_func.h>
```

Classes

- class BNode
- class BIter

  Iterator for BList.
- class BList< T >

  Template based list class.
- class BList< T >::Node

Macros

- #define BListLoop(list, i) for(BIter i = list.begin(); !list.isEnd(i); list.next(i))

8.51.1 Macro Definition Documentation

8.51.1.1 #define BListLoop(*list, i*) for(BIter i = list.begin(); !list.isEnd(i); list.next(i))

8.52 /src/cern/tms/beam/libBeam/BList_func.h File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <memory.h>
```

8.53 /src/cern/tms/beam/libBeam/BMutex.cpp File Reference

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

Macros

- #define MDEBUG 0

8.53.1 Macro Definition Documentation

8.53.1.1 #define MDEBUG 0

8.54 /src/cern/tms/beam/libBeam/BMutex.h File Reference

```
#include <pthread.h>
```

Classes

- class BMutex

  Mutex class.
- class BMutexLock

8.55 /src/cern/tms/beam/libBeam/BMysql.cpp File Reference

```
#include <stdlib.h>
#include <string.h>
#include <BMysql.h>
```

8.56 /src/cern/tms/beam/libBeam/BMysql.h File Reference

```
#include <BTypes.h>
#include <BError.h>
#include <BDict.h>
#include <BMutex.h>
#include <mysql/mysql.h>
```

Classes

- class BMysql

8.57 /src/cern/tms/beam/libBeam/BNameValue.h File Reference

```
#include <BList.h>
#include <BString.h>
```

Classes

- class BNameValue< T >
- class BNameValueList< T >

8.58 /src/cern/tms/beam/libBeam/Boap.cpp File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/tcp.h>
#include <Boap.h>
#include <byteswap.h>
#include <BoapnsD.h>
#include <BoapnsC.h>
```

Macros

- #define DEBUG 0
- #define APIVERSION_TEST 1
- #define dprintf(fmt, a...)
- #define IS_BIG_ENDIAN 1

Variables

- const int boapPort = 12000

  The default BOAP connection port.

8.58.1 Macro Definition Documentation

8.58.1.1 #define APIVERSION_TEST 1

8.58.1.2 #define DEBUG 0

8.58.1.3 #define dprintf(*fmt, a...*)

8.58.1.4 #define IS_BIG_ENDIAN 1

8.58.2 Variable Documentation

8.58.2.1 const int boapPort = 12000

The default BOAP connection port.

8.59 /src/cern/tms/beam/libBeam/Boap.h File Reference

```
#include <stdint.h>
#include <BTypes.h>
#include <BPoll.h>
#include <BSocket.h>
#include <BThread.h>
#include <BError.h>
#include <BEvent1.h>
#include <BMutex.h>
#include <BTimeStamp.h>
#include <BBuffer.h>
```

Classes

- struct BoapPacketHead
- class BoapPacket
- class BoapClientObject
- class BoapSignalObject
- class BoapServiceEntry
- class BoapServerConnection
- class BoapServer
- class BoapFuncEntry
- class BoapServiceObject

Namespaces

- Boapns

Typedefs

- typedef BUInt32 BoapService
- typedef BError(BoapServiceObject::∗ BoapFunc)(BoapServerConnection ∗conn, BoapPacket &rx, BoapPacket &tx)

Enumerations

- enum BoapType {
  BoapTypeRpc, BoapTypeRpcReply, BoapTypeSignal, BoapTypeRpcError,
  BoapTypeRpc, BoapTypeSignal }
- enum BoapPriority { BoapPriorityLow, BoapPriorityNormal, BoapPriorityHigh }

Variables

- const BUInt32 BoapMagic = 0x424F4100

8.59.1 Typedef Documentation

8.59.1.1 typedef BError(BoapServiceObject::∗ BoapFunc)(BoapServerConnection ∗conn, BoapPacket &rx, BoapPacket &tx)

8.59.1.2 typedef BUInt32 BoapService

8.59.2 Enumeration Type Documentation

8.59.2.1 enum BoapPriority

Enumerator

BoapPriorityLow

BoapPriorityNormal

BoapPriorityHigh

8.59.2.2 enum BoapType

Enumerator

BoapTypeRpc

BoapTypeRpcReply

BoapTypeSignal

BoapTypeRpcError

BoapTypeRpc

BoapTypeSignal

8.59.3 Variable Documentation

8.59.3.1 const BUInt32 BoapMagic = 0x424F4100

8.60 /src/cern/tms/beam/libBeam/BoapMc.cpp File Reference

```
#include <BoapMc.h>
#include <BCrc16.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
```

Macros

- #define DEBUG_LOCAL 0
- #define DEBUG_LOCAL1 0
- #define dlprintf(fmt, a...)
- #define dl1printf(fmt, a...)

8.60.1 Macro Definition Documentation

8.60.1.1 #define DEBUG_LOCAL 0

8.60.1.2 #define DEBUG_LOCAL1 0

8.60.1.3 #define dl1printf(*fmt, a...*)

8.60.1.4 #define dlprintf(*fmt, a...*)

8.61 /src/cern/tms/beam/libBeam/BoapMc.h File Reference

```
#include <BTypes.h>
#include <BMutex.h>
#include <BSemaphore.h>
#include <BQueue.h>
#include <BFifo.h>
#include <BComms.h>
```

Classes

- struct BoapMcPacketHead
- class BoapMcPacket
- class BoapMcClientObject
- class BoapMcSignalObject
- class BoapMcServiceObject
- class BoapMcComms

Enumerations

- enum BoapMcType { BoapMcTypeRequest = 0x00, BoapMcTypeReply = 0x80 }

Functions

- struct BoapMcPacketHead __attribute__ ((aligned(8), packed))

Variables

- BUInt8 length
- BUInt8 addressTo
- BUInt8 addressFrom
- BUInt8 cmd
- BUInt16 error
- BUInt16 checksum
- class BoapMcPacket __attribute__

8.61.1 Enumeration Type Documentation

8.61.1.1 enum BoapMcType

Enumerator

BoapMcTypeRequest

BoapMcTypeReply

8.61.2 Function Documentation

8.61.2.1 struct **BoapMcPacketHead** __attribute__ ((aligned(8), packed))

8.61.3 Variable Documentation

8.61.3.1 class **BoapMcPacket** __attribute__

8.61.3.2 **BUInt8** addressFrom

8.61.3.3 **BUInt8** addressTo

8.61.3.4 **BUInt16** checksum

8.61.3.5 **BUInt8** cmd

8.61.3.6 **BUInt16** error

8.61.3.7 **BUInt8** length

8.62 /src/cern/tms/beam/libBeam/BoapnsC.cpp File Reference

```
#include <BoapnsC.h>
```

Namespaces

- Boapns

8.63 /src/cern/tms/beam/libBeam/BoapnsC.h File Reference

```
#include <stdlib.h>
#include <stdint.h>
#include <Boap.h>
#include <BString.h>
#include <BList.h>
#include <BArray.h>
#include <BoapnsD.h>
```

Classes

- class Boapns::Boapns

Namespaces

- Boapns

Variables

- const BUInt32 Boapns::apiVersion = 0

8.64 /src/cern/tms/beam/libBeam/BoapnsD.cpp File Reference

```
#include <BoapnsD.h>
```

Namespaces

- Boapns

8.65 /src/cern/tms/beam/libBeam/BoapnsD.h File Reference

```
#include <Boap.h>
#include <BObj.h>
#include <BDate.h>
#include <BTimeStamp.h>
#include <BComplex.h>
#include <BList.h>
#include <BArray.h>
```

Classes

- class Boapns::BoapEntry

Namespaces

- Boapns

8.66 /src/cern/tms/beam/libBeam/BoapSimple.cc File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <Boap.h>
#include <BoapnsD.h>
#include <BoapnsC.h>
```

Macros

- #define DEBUG 0
- #define dprintf(fmt, a...)

Variables

- const int roundSize = 256

8.66.1 Macro Definition Documentation

8.66.1.1 #define DEBUG 0

8.66.1.2 #define dprintf(*fmt, a...*)

8.66.2 Variable Documentation

8.66.2.1 const int roundSize = 256

8.67 /src/cern/tms/beam/libBeam/BoapSimple.h File Reference

```
#include <stdint.h>
#include <BPoll.h>
#include <BSocket.h>
#include <BError.h>
```

Classes

- struct BoapPacketHead
- class BoapPacket
- class BoapClientObject
- class BoapSignalObject
- class BoapServiceEntry
- class BoapServer
- class BoapFuncEntry
- class BoapServiceObject

Typedefs

- typedef int8_t Int8
- typedef uint8_t UInt8
- typedef int16_t Int16
- typedef uint16_t UInt16
- typedef int32_t Int32
- typedef uint32_t UInt32
- typedef double Double
- typedef uint32_t BoapService
- typedef BError(BoapServiceObject::∗ BoapFunc)(BoapPacket &rx, BoapPacket &tx)

Enumerations

- enum BoapType {
  BoapTypeRpc, BoapTypeRpcReply, BoapTypeSignal, BoapTypeRpcError,
  BoapTypeRpc, BoapTypeSignal }

8.67.1 Typedef Documentation

8.67.1.1 typedef BError(BoapServiceObject::∗ BoapFunc)(BoapPacket &rx, BoapPacket &tx)

8.67.1.2 typedef uint32_t BoapService

8.67.1.3 typedef double Double

8.67.1.4 typedef int16_t Int16

8.67.1.5 typedef int32_t Int32

8.67.1.6 typedef int8_t Int8

8.67.1.7 typedef uint16_t UInt16

8.67.1.8 typedef uint32_t UInt32

8.67.1.9 typedef uint8_t UInt8

8.67.2 Enumeration Type Documentation

8.67.2.1 enum BoapType

Enumerator

BoapTypeRpc

BoapTypeRpcReply

BoapTypeSignal

BoapTypeRpcError

BoapTypeRpc

BoapTypeSignal

8.68 /src/cern/tms/beam/libBeam/BObj.cpp File Reference

```
#include <BObj.h>
```

8.69 /src/cern/tms/beam/libBeam/BObj.h File Reference

```
#include <BTypes.h>
#include <BDict.h>
#include <BString.h>
#include <BError.h>
```

Classes

- class BObj

8.70 /src/cern/tms/beam/libBeam/BObjStringFormat.cpp File Reference

```
#include <BObjStringFormat.h>
#include <BTime.h>
#include <math.h>
```

Functions

- BString toBString (BString n, Bool v)
- BString toBString (BString n, BInt8 v)
- BString toBString (BString n, BUInt8 v)
- BString toBString (BString n, BInt16 v)
- BString toBString (BString n, BUInt16 v)
- BString toBString (BString n, BInt32 v)
- BString toBString (BString n, BUInt32 v)
- BString toBString (BString n, BInt64 v)
- BString toBString (BString n, BUInt64 v)
- BString toBString (BString n, BFloat32 v)
- BString toBString (BString n, BFloat64 v)
- BString toBString (BString n, BChar v)
- BString toBString (BString n, const BChar ∗v)
- BString toBString (BString n, BString v)
- BString toBString (BString n, BError v)
- BString toBString (BString n, BTime v)
- BString toBString (BString name, const BObjMember ∗m, const void ∗obj, BStringList ignoreFields)
- BString toBString (BString n, BObj &obj)
- BString toBStringJson (BString n, Bool v)
- BString toBStringJson (BString n, BInt8 v)
- BString toBStringJson (BString n, BUInt8 v)
- BString toBStringJson (BString n, BInt16 v)
- BString toBStringJson (BString n, BUInt16 v)
- BString toBStringJson (BString n, BInt32 v)
- BString toBStringJson (BString n, BUInt32 v)
- BString toBStringJson (BString n, BInt64 v)
- BString toBStringJson (BString n, BUInt64 v)
- BString toBStringJson (BString n, BFloat32 v)
- BString toBStringJson (BString n, BFloat64 v)
- BString toBStringJson (BString n, BChar v)
- BString toBStringJson (BString n, const BChar ∗v)
- BString toBStringJson (BString n, BString v)
- BString toBStringJson (BString n, BError v)
- BString toBStringJson (BString n, BTime v)
- BString toBStringJson (BString n, const BObjMember ∗m, const void ∗obj, BStringList ignoreFields)
- BString toBStringJson (BString n, BObj &obj)
- BError toBDictStringFromJson (BString json, BDictString &ds)

8.70.1 Function Documentation

8.70.1.1 BError toBDictStringFromJson (BString *json*, BDictString & *ds*)

8.70.1.2 BString toBString (BString *n*, Bool *v*)

8.70.1.3 BString toBString (BString *n*, BInt8 *v*)

8.70.1.4 BString toBString (BString *n*, BUInt8 *v*)

8.70.1.5 BString toBString (BString *n*, BInt16 *v*)

8.70.1.6 BString toBString (BString *n*, BUInt16 *v*)

8.70.1.7 BString toBString (BString *n*, BInt32 *v*)

8.70.1.8 BString toBString (BString *n*, BUInt32 *v*)

8.70.1.9 BString toBString (BString *n*, BInt64 *v*)

8.70.1.10 BString toBString (BString *n*, BUInt64 *v*)

8.70.1.11 BString toBString (BString *n*, BFloat32 *v*)

8.70.1.12 BString toBString (BString *n*, BFloat64 *v*)

8.70.1.13 BString toBString (BString *n*, BChar *v*)

8.70.1.14 BString toBString (BString *n*, const BChar ∗ *v*)

8.70.1.15 BString toBString (BString *n*, BString *v*)

8.70.1.16 BString toBString (BString *n*, BError *v*)

8.70.1.17 BString toBString (BString *n*, BTime *v*)

8.70.1.18 BString toBString (BString *name*, const BObjMember ∗ *m*, const void ∗ *obj*, BStringList *ignoreFields*)

8.70.1.19 BString toBString (BString *n*, BObj & *obj*)

8.70.1.20 BString toBStringJson (BString *n*, Bool *v*)

8.70.1.21 BString toBStringJson (BString *n*, BInt8 *v*)

8.70.1.22 BString toBStringJson (BString *n*, BUInt8 *v*)

8.70.1.23 BString toBStringJson (BString *n*, BInt16 *v*)

8.70.1.24 BString toBStringJson (BString *n*, BUInt16 *v*)

8.70.1.25 BString toBStringJson (BString *n*, BInt32 *v*)

8.70.1.26 BString toBStringJson (BString *n*, BUInt32 *v*)

8.70.1.27 BString toBStringJson (BString *n*, BInt64 *v*)

8.70.1.28 BString toBStringJson (BString *n*, BUInt64 *v*)

8.70.1.29 BString toBStringJson (BString *n*, BFloat32 *v*)

8.70.1.30 BString toBStringJson (BString *n*, BFloat64 *v*)

8.70.1.31 BString toBStringJson (BString *n*, BChar *v*)

8.70.1.32 BString toBStringJson (BString *n*, const BChar ∗ *v*)

8.70.1.33 BString toBStringJson (BString *n*, BString *v*)

8.70.1.34 BString toBStringJson (BString *n*, BError *v*)

8.70.1.35 BString toBStringJson (BString *n*, BTime *v*)

8.70.1.36 BString toBStringJson (BString *n*, const BObjMember ∗ *m*, const void ∗ *obj*, BStringList *ignoreFields*)

8.70.1.37 BString toBStringJson (BString *n*, BObj & *obj*)

8.71 /src/cern/tms/beam/libBeam/BObjStringFormat.h File Reference

```
#include <BObj.h>
#include <BString.h>
#include <BTime.h>
```

Functions

- BString toBString (BString name, Bool value)
- BString toBString (BString name, BInt8 value)
- BString toBString (BString name, BUInt8 value)
- BString toBString (BString name, BInt16 value)
- BString toBString (BString name, BUInt16 value)
- BString toBString (BString name, BInt32 value)
- BString toBString (BString name, BUInt32 value)
- BString toBString (BString name, BFloat32 value)
- BString toBString (BString name, BFloat64 value)
- BString toBString (BString name, BChar value)
- BString toBString (BString name, const BChar ∗value)
- BString toBString (BString name, BString value)
- BString toBString (BString name, BError value)
- BString toBString (BString name, BTime time)
- BString toBString (BString name, const BObjMember ∗members, const void ∗obj, BStringList ignoreFields=BStringList())
- BString toBString (BString name, BObj &obj)
- BString toBStringJson (BString name, Bool value)
- BString toBStringJson (BString name, BInt8 value)
- BString toBStringJson (BString name, BUInt8 value)
- BString toBStringJson (BString name, BInt16 value)
- BString toBStringJson (BString name, BUInt16 value)
- BString toBStringJson (BString name, BInt32 value)
- BString toBStringJson (BString name, BUInt32 value)
- BString toBStringJson (BString name, BFloat32 value)

- BString toBStringJson (BString name, BFloat64 value)
- BString toBStringJson (BString name, BChar value)
- BString toBStringJson (BString name, const BChar ∗value)
- BString toBStringJson (BString name, BString value)
- BString toBStringJson (BString name, BError value)
- BString toBStringJson (BString name, BTime time)
- BString toBStringJson (BString name, const BObjMember ∗members, const void ∗obj, BStringList ignoreFields=BStringList())
- BString toBStringJson (BString name, BObj &obj)
- BError toBDictStringFromJson (BString json, BDictString &ds)
- BString base64_encode (void ∗data, BUInt len)
- BError base64_decode (BString strIn, BString &strOut)

8.71.1 Function Documentation

8.71.1.1 BError base64_decode (BString *strIn,* BString & *strOut*)

8.71.1.2 BString base64_encode (void ∗ *data,* BUInt *len*)

8.71.1.3 BError toBDictStringFromJson (BString *json,* BDictString & *ds*)

8.71.1.4 BString toBString (BString *name,* Bool *value*)

8.71.1.5 BString toBString (BString *name,* BInt8 *value*)

8.71.1.6 BString toBString (BString *name,* BUInt8 *value*)

8.71.1.7 BString toBString (BString *name,* BInt16 *value*)

8.71.1.8 BString toBString (BString *name,* BUInt16 *value*)

8.71.1.9 BString toBString (BString *name,* BInt32 *value*)

8.71.1.10 BString toBString (BString *name,* BUInt32 *value*)

8.71.1.11 BString toBString (BString *name,* BFloat32 *value*)

8.71.1.12 BString toBString (BString *name,* BFloat64 *value*)

8.71.1.13 BString toBString (BString *name,* BChar *value*)

8.71.1.14 BString toBString (BString *name,* const BChar ∗ *value*)

8.71.1.15 BString toBString (BString *name,* BString *value*)

8.71.1.16 BString toBString (BString *name,* BError *value*)

8.71.1.17 BString toBString (BString *name,* BTime *time*)

8.71.1.18 BString toBString (BString *name,* const BObjMember ∗ *members,* const void ∗ *obj,* BStringList *ignoreFields* = `BStringList()`)

8.71.1.19 BString toBString (BString *name,* BObj & *obj*)

8.71.1.20 BString toBStringJson (BString *name,* Bool *value*)

8.71.1.21 BString toBStringJson (BString *name,* BInt8 *value*)

8.71.1.22 BString toBStringJson (BString *name,* BUInt8 *value*)

8.71.1.23 BString toBStringJson (BString *name,* BInt16 *value*)

8.71.1.24 BString toBStringJson (BString *name,* BUInt16 *value*)

8.71.1.25 BString toBStringJson (BString *name,* BInt32 *value*)

8.71.1.26 BString toBStringJson (BString *name,* BUInt32 *value*)

8.71.1.27 BString toBStringJson (BString *name,* BFloat32 *value*)

8.71.1.28 BString toBStringJson (BString *name,* BFloat64 *value*)

8.71.1.29 BString toBStringJson (BString *name,* BChar *value*)

8.71.1.30 BString toBStringJson (BString *name,* const BChar ∗ *value*)

8.71.1.31 BString toBStringJson (BString *name,* BString *value*)

8.71.1.32 BString toBStringJson (BString *name,* BError *value*)

8.71.1.33 BString toBStringJson (BString *name,* BTime *time*)

8.71.1.34 BString toBStringJson (BString *name,* const BObjMember ∗ *members,* const void ∗ *obj,* BStringList *ignoreFields =* `BStringList()`)

8.71.1.35 BString toBStringJson (BString *name,* BObj & *obj*)

8.72 /src/cern/tms/beam/libBeam/BPoll.cpp File Reference

```
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <BPoll.h>
```

8.73 /src/cern/tms/beam/libBeam/BPoll.h File Reference

```
#include <BList.h>
#include <BError.h>
#include <sys/poll.h>
```

Classes

- class BPoll

  This class provides an interface for polling a number of file descriptors. It uses round robin polling.

8.74 /src/cern/tms/beam/libBeam/BQueue.h File Reference

```
#include <BTypes.h>
#include <BError.h>
#include <BList.h>
#include <BMutex.h>
#include <BCondInt.h>
```

Classes

- class BQueue< T >

  Queue class.

Typedefs

- typedef BQueue< BInt32 > BQueueInt

8.74.1 Typedef Documentation

8.74.1.1 typedef **BQueue<BInt32> BQueueInt**

8.75 /src/cern/tms/beam/libBeam/BRefData.cpp File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <BRefData.h>
```

Macros

- #define CHUNK 16

8.75.1 Macro Definition Documentation

8.75.1.1 #define CHUNK 16

8.76 /src/cern/tms/beam/libBeam/BRefData.h File Reference

```
#include <BAtomicCount.h>
```

Classes

- class BRefData

8.77 /src/cern/tms/beam/libBeam/BRtc.cpp File Reference

```
#include <BRtc.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/rtc.h>
```

8.78 /src/cern/tms/beam/libBeam/BRtc.h File Reference

```
#include <BError.h>
#include <BThread.h>
#include <BCond.h>
```

Classes

- class BRtc

  Realtime clock.
- class BRtcThreaded

  Threaded real time clock.

8.79 /src/cern/tms/beam/libBeam/BRWLock.cpp File Reference

```
#include <BRWLock.h>
```

8.80 /src/cern/tms/beam/libBeam/BRWLock.h File Reference

```
#include <pthread.h>
```

Classes

- class BRWLock

  thread read-write locks

8.81 /src/cern/tms/beam/libBeam/BSema.cpp File Reference

```
#include <BSema.h>
#include <errno.h>
#include <sys/time.h>
```

8.82 /src/cern/tms/beam/libBeam/BSema.h File Reference

```
#include <sys/types.h>
#include <semaphore.h>
```

Classes

- class BSema

  Sempahore class.

8.83 /src/cern/tms/beam/libBeam/BSemaphore.cpp File Reference

```
#include <BSemaphore.h>
#include <sys/time.h>
```

8.84 /src/cern/tms/beam/libBeam/BSemaphore.h File Reference

```
#include <BTypes.h>
#include <BMutex.h>
#include <semaphore.h>
```

Classes

- class BSemaphore

  Semaphore class.
- class BSemaphoreCount

8.85 /src/cern/tms/beam/libBeam/BSocket.cpp File Reference

```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <net/if.h>
#include "BSocket.h"
```

Macros

- #define IP_MTU 14

8.85.1 Macro Definition Documentation

8.85.1.1 #define IP_MTU 14

8.86 /src/cern/tms/beam/libBeam/BSocket.h File Reference

```
#include <BString.h>
#include <BError.h>
#include <BTypes.h>
#include <stdint.h>
#include <sys/types.h>
#include <sys/prctl.h>
```

Classes

- class BSocketAddress

  Socket Address.

- class BSocketAddressINET

  IP aware socket address.

- class BSocket

8.87 /src/cern/tms/beam/libBeam/BSpi.cpp File Reference

```
#include <BSpi.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/ioctl.h>
#include <linux/spi/spidev.h>
```

8.88 /src/cern/tms/beam/libBeam/BSpi.h File Reference

```
#include <BTypes.h>
#include <BError.h>
```

Classes

- class BSpi

  BSpi class.

8.89 /src/cern/tms/beam/libBeam/BString.cpp File Reference

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdarg.h>
#include <ctype.h>
#include <BString.h>
#include <BError.h>
#include <regex.h>
```

Macros

- #define STRIP 0x7f
- #define MINUS '-'

Functions

- static int gmatch (const char ∗s, const char ∗p)
- std::ostream & operator<< (std::ostream &o, BString &s)
- std::istream & operator>> (std::istream &i, BString &s)
- int bstringListinList (BStringList &list, BString s)
- BString blistToString (const BStringList &list)
- BStringList bstringToList (BString str, int stripSpaces)
- BStringList charToList (const char ∗∗str)
- BString barrayToString (const BStringArray &list)
- BStringArray bstringToArray (BString str, int stripSpaces)
- BStringArray charToArray (const char ∗∗str)
- void toBString (BString &v, BString &s)
- void toBString (BStringList &v, BString &s)
- void toBString (BInt32 &v, BString &s)
- void toBString (BUInt32 &v, BString &s)
- void toBString (BUInt64 &v, BString &s)
- void toBString (BFloat64 &v, BString &s)
- void fromBString (BString &s, BString &v)
- void fromBString (BString &s, BStringList &v)
- void fromBString (BString &s, BInt32 &v)
- void fromBString (BString &s, BUInt32 &v)
- void fromBString (BString &s, BUInt64 &v)
- void fromBString (BString &s, BFloat64 &v)

Variables

- static const BUInt8 base64_decode_table [ ]

8.89.1 Macro Definition Documentation

8.89.1.1 #define MINUS '-'

8.89.1.2 #define STRIP 0x7f

8.89.2 Function Documentation

8.89.2.1 BString barrayToString (const BStringArray & *list*)

8.89.2.2 BString blistToString (const BStringList & *list*)

8.89.2.3 int bstringListinList (BStringList & *list,* BString *s*)

8.89.2.4 BStringArray bstringToArray (BString *str,* int *stripSpaces*)

8.89.2.5 BStringList bstringToList (BString *str,* int *stripSpaces*)

8.89.2.6 BStringArray charToArray (const char ** *str*)

8.89.2.7 BStringList charToList (const char ** *str*)

8.89.2.8 void fromBString (BString & *s,* BString & *v*)

8.89.2.9 void fromBString (BString & *s,* BStringList & *v*)

8.89.2.10 void fromBString (BString & *s,* BInt32 & *v*)

8.89.2.11 void fromBString (BString & *s,* BUInt32 & *v*)

8.89.2.12 void fromBString (BString & *s,* BUInt64 & *v*)

8.89.2.13 void fromBString (BString & *s,* BFloat64 & *v*)

8.89.2.14 static int gmatch (const char * *s,* const char * *p*) [static]

8.89.2.15 std::ostream& operator<< (std::ostream & *o,* BString & *s*)

8.89.2.16 std::istream& operator>> (std::istream & *i,* BString & *s*)

8.89.2.17 void toBString (BString & *v,* BString & *s*)

8.89.2.18 void toBString (BStringList & *v,* BString & *s*)

8.89.2.19 void toBString (BInt32 & *v,* BString & *s*)

8.89.2.20 void toBString (BUInt32 & *v,* BString & *s*)

8.89.2.21 void toBString (BUInt64 & *v,* BString & *s*)

8.89.2.22 void toBString (BFloat64 & *v,* BString & *s*)

8.89.3 Variable Documentation

8.89.3.1 const BUInt8 base64_decode_table[ ] [static]

Initial value:

```
= {
        66,66,66,66,66,66,66,66,66,66,64,66,66,66,66,66,66,66,66,66,66,66,66,66,66,
        66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,62,66,66,66,63,52,53,
        54,55,56,57,58,59,60,61,66,66,66,65,66,66,66, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
        10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,66,66,66,66,66,66,26,27,28,
        29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,66,66,
        66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,
        66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,
        66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,
        66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,
        66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,
        66,66,66,66,66,66
}
```

8.90 /src/cern/tms/beam/libBeam/BString.h File Reference

```
#include <BTypes.h>
#include <BRefData.h>
#include <BList.h>
#include <BArray.h>
#include <iostream>
```

Classes

- class BString

Functions

- std::ostream & operator<< (std::ostream &o, BString &s)
- std::istream & operator>> (std::istream &i, BString &s)
- void toBString (BString &v, BString &s)
- void toBString (BStringList &v, BString &s)
- void toBString (BInt32 &v, BString &s)
- void toBString (BUInt32 &v, BString &s)
- void toBString (BUInt64 &v, BString &s)
- void toBString (BFloat64 &v, BString &s)
- void fromBString (BString &s, BString &v)
- void fromBString (BString &s, BStringList &v)
- void fromBString (BString &s, BInt32 &v)
- void fromBString (BString &s, BUInt32 &v)
- void fromBString (BString &s, BUInt64 &v)
- void fromBString (BString &s, BFloat64 &v)

8.90.1 Function Documentation

8.90.1.1 void fromBString (BString & *s*, BString & *v*)

8.90.1.2 void fromBString (BString & *s*, BStringList & *v*)

8.90.1.3 void fromBString (BString & *s*, BInt32 & *v*)

8.90.1.4 void fromBString (BString & *s*, BUInt32 & *v*)

8.90.1.5 void fromBString (BString & *s*, BUInt64 & *v*)

8.90.1.6 void fromBString (BString & *s*, BFloat64 & *v*)

8.90.1.7 std::ostream& operator<< (std::ostream & *o*, BString & *s*)

8.90.1.8 std::istream& operator>> (std::istream & *i*, BString & *s*)

8.90.1.9 void toBString (BString & *v*, BString & *s*)

8.90.1.10 void toBString (BStringList & *v*, BString & *s*)

8.90.1.11 void toBString (BInt32 & *v*, BString & *s*)

8.90.1.12 void toBString (BUInt32 & *v*, BString & *s*)

8.90.1.13 void toBString (BUInt64 & *v*, BString & *s*)

8.90.1.14 void toBString (BFloat64 & *v*, BString & *s*)

8.91 /src/cern/tms/beam/libBeam/BStringLocked.h File Reference

```
#include <BString.h>
#include <BMutex.h>
```

Classes

- class BStringMutex
- class BStringLocked

8.92 /src/cern/tms/beam/libBeam/BTable.cpp File Reference

```
#include <BTable.h>
```

8.93 /src/cern/tms/beam/libBeam/BTable.h File Reference

```
#include <BArray.h>
#include <BString.h>
```

Classes

- class BTable

8.94 /src/cern/tms/beam/libBeam/BThread.cpp File Reference

```
#include <BThread.h>
#include <unistd.h>
#include <errno.h>
#include <sys/types.h>
```

8.95 /src/cern/tms/beam/libBeam/BThread.h File Reference

```
#include <pthread.h>
```

Classes

- class BThread

8.96 /src/cern/tms/beam/libBeam/BTime.cpp File Reference

```
#include <BTime.h>
```

Functions

- static bool yearIsLeap (BUInt16 year)
- static BUInt16 yearDays (BUInt16 year)

Variables

- static BUInt16 monDays [2][13]

8.96.1 Function Documentation

8.96.1.1 static BUInt16 yearDays (**BUInt16** *year*) [inline], [static]

8.96.1.2 static bool yearIsLeap (**BUInt16** *year*) [inline], [static]

8.96.2 Variable Documentation

8.96.2.1 BUInt16 monDays[2][13] [static]

Initial value:

```
= {
        { 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334, 365 },
        { 0, 31, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335, 366 }
}
```

8.97 /src/cern/tms/beam/libBeam/BTime.h File Reference

```
#include <BTypes.h>
#include <BError.h>
#include <BString.h>
```

Classes

- class BTime

8.98 /src/cern/tms/beam/libBeam/BTimer.cpp File Reference

```
#include <BTimer.h>
#include <sys/time.h>
```

8.99 /src/cern/tms/beam/libBeam/BTimer.h File Reference

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

Classes

- class BTimer

  Stopwatch style timer.

8.100 /src/cern/tms/beam/libBeam/BTimeStamp.cpp File Reference

```
#include <BTimeStamp.h>
#include <BTimeStampMs.h>
#include <sys/time.h>
```

Functions

- void toBString (BTimeStamp &v, BString &s)
- void fromBString (BString &s, BTimeStamp &v)

Variables

- static int mon_yday [2][13]

8.100.1 Function Documentation

8.100.1.1 void fromBString (**BString & *s*, BTimeStamp & *v***)

8.100.1.2 void toBString (**BTimeStamp & *v*, BString & *s***)

8.100.2 Variable Documentation

8.100.2.1 int mon_yday[2][13] `[static]`

Initial value:

```
= {
        { 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334, 365 },
        { 0, 31, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335, 366 }
}
```

8.101 /src/cern/tms/beam/libBeam/BTimeStamp.h File Reference

```
#include <stdint.h>
#include <BError.h>
```

Classes

- class BTimeStamp

Functions

- void toBString (BTimeStamp &v, BString &s)
- void fromBString (BString &s, BTimeStamp &v)

8.101.1 Function Documentation

8.101.1.1 void fromBString (BString & *s*, BTimeStamp & *v*)

8.101.1.2 void toBString (BTimeStamp & *v*, BString & *s*)

8.102 /src/cern/tms/beam/libBeam/BTimeStampMs.cpp File Reference

```
#include <BTimeStampMs.h>
#include <sys/time.h>
```

Variables

- static int mon_yday [2][13]

8.102.1 Variable Documentation

8.102.1.1 int mon_yday[2][13] [static]

Initial value:

```
= {
        { 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334, 365 },
        { 0, 31, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335, 366 }
}
```

8.103 /src/cern/tms/beam/libBeam/BTimeStampMs.h File Reference

```
#include <stdint.h>
#include <BError.h>
```

Classes

- class BTimeStampMs

8.104 /src/cern/tms/beam/libBeam/BTypes.h File Reference

```
#include <stdint.h>
#include <sys/types.h>
#include <vector>
```

Classes

- struct BObjMember

Typedefs

- typedef bool Bool
- typedef int8_t BInt8
- typedef uint8_t BUInt8
- typedef int16_t BInt16
- typedef uint16_t BUInt16
- typedef int32_t BInt32
- typedef uint32_t BUInt32
- typedef int64_t BInt64
- typedef uint64_t BUInt64
- typedef float BFloat32
- typedef double BFloat64
- typedef char BChar
- typedef BInt32 BInt
- typedef BUInt32 BUInt
- typedef BFloat32 BFloat
- typedef BFloat64 BDouble
- typedef size_t BSize
- typedef std::vector< BFloat32 > BArrayFloat
- typedef std::vector< BFloat64 > BArrayDouble
- typedef BUInt32 BTimeout

Enumerations

- enum BType {
  BTypeNone, BTypeBool, BTypeInt8, BTypeUInt8,
  BTypeInt16, BTypeUInt16, BTypeInt32, BTypeUInt32,
  BTypeInt64, BTypeUInt64, BTypeFloat32, BTypeFloat64,
  BTypeChar, BTypeString, BTypeError, BTypeTime,
  BTypeObj = 100 }
- enum BTypeComp {
  BTypeCompSingle, BTypeCompArray, BTypeCompArrayFixed, BTypeCompList,
  BTypeCompDict }

Functions

- BTimeout timeoutTicks (BTimeout timeoutUs)
- void byteSwap8 (void ∗d, void ∗s)
- void byteSwap16 (void ∗d, void ∗s)
- void byteSwap32 (void ∗d, void ∗s)
- void byteSwap64 (void ∗d, void ∗s)

Variables

- const BTimeout BTimeoutForever = 0xFFFFFFFF

8.104.1 Typedef Documentation

8.104.1.1 typedef std::vector<BFloat64> BArrayDouble

8.104.1.2 typedef std::vector<BFloat32> BArrayFloat

8.104.1.3 typedef char BChar

8.104.1.4 typedef BFloat64 BDouble

8.104.1.5 typedef BFloat32 BFloat

8.104.1.6 typedef float BFloat32

8.104.1.7 typedef double BFloat64

8.104.1.8 typedef BInt32 BInt

8.104.1.9 typedef int16_t BInt16

8.104.1.10 typedef int32_t BInt32

8.104.1.11 typedef int64_t BInt64

8.104.1.12 typedef int8_t BInt8

8.104.1.13 typedef bool Bool

8.104.1.14 typedef size_t BSize

8.104.1.15 typedef BUInt32 BTimeout

8.104.1.16 typedef BUInt32 BUInt

8.104.1.17 typedef uint16_t BUInt16

8.104.1.18 typedef uint32_t BUInt32

8.104.1.19 typedef uint64_t BUInt64

8.104.1.20 typedef uint8_t BUInt8

8.104.2 Enumeration Type Documentation

8.104.2.1 enum BType

Enumerator

BTypeNone

BTypeBool

BTypeInt8

BTypeUInt8

BTypeInt16

BTypeUInt16

BTypeInt32

BTypeUInt32

BTypeInt64

BTypeUInt64

BTypeFloat32

BTypeFloat64

BTypeChar

BTypeString

BTypeError

BTypeTime

BTypeObj

8.104.2.2 enum BTypeComp

Enumerator

BTypeCompSingle

BTypeCompArray

BTypeCompArrayFixed

BTypeCompList

BTypeCompDict

8.104.3 Function Documentation

8.104.3.1 void byteSwap16 (void ∗ *d*, void ∗ *s*) [inline]

8.104.3.2 void byteSwap32 (void ∗ *d*, void ∗ *s*) [inline]

8.104.3.3 void byteSwap64 (void ∗ *d*, void ∗ *s*) [inline]

8.104.3.4 void byteSwap8 (void ∗ *d*, void ∗ *s*) [inline]

8.104.3.5 BTimeout timeoutTicks (BTimeout *timeoutUs*) [inline]

8.104.4 Variable Documentation

8.104.4.1 const BTimeout BTimeoutForever = 0xFFFFFFFF

8.105 /src/cern/tms/beam/libBeam/BUrl.cpp File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <memory.h>
#include <BUrl.h>
#include <curl/curl.h>
```

8.106 /src/cern/tms/beam/libBeam/BUrl.h File Reference

```
#include <stdio.h>
#include <BString.h>
#include <BError.h>
```

Classes

- class BUrl

  Basic access to a Url.

8.107 overview.dox File Reference

8.108 SigGen.cpp File Reference

```
#include <SigGen.h>
#include <math.h>
#include <time.h>
```

Macros

- #define DEBUG 0
- #define dprintf(fmt, a...)

8.108.1 Macro Definition Documentation

8.108.1.1 #define DEBUG 0

8.108.1.2 #define dprintf(*fmt,* *a...*)

8.109 SigGen.h File Reference

```
#include <BList.h>
#include <BError.h>
```

Classes

- class BSignal
- class SigGen
- class SigGenSine
- class SigGenSquare
- class SigGenNoise
- class SigGenPulse
- class SigGenBeam

Typedefs

- typedef float Sample
- typedef BList< BSignal > BSignalList

8.109.1 Typedef Documentation

8.109.1.1 typedef **BList<BSignal> BSignalList**

8.109.1.2 typedef float **Sample**

8.110 test1.cpp File Reference

```
#include <stdio.h>
#include <TmsLib.h>
```

Functions

- void printCycleParams (Tms::CycleParam p)
- int main ()

8.110.1 Function Documentation

8.110.1.1 int main ( )

8.110.1.2 void printCycleParams (**Tms::CycleParam** *p*)

8.111 TmsC.cc File Reference

```
#include <TmsC.h>
```

Namespaces

- Tms

8.112 TmsC.h File Reference

This file contains the TmsAPi class definitions.

```
#include <stdlib.h>
#include <stdint.h>
#include <Boap.h>
#include <BString.h>
#include <BList.h>
#include <BArray.h>
#include <TmsD.h>
```

Classes

- class Tms::PuControl

  This class defines the parameters for a test data capture.
- class Tms::PuProcess

  This interface provides functions to configure and capture data from individual pick-up.
- class Tms::TmsControl

  This interface provides functions to control, test and get statistics from the TMS as a whole.
- class Tms::TmsProcess

  This interface provides functions to capture data from the TMS as a whole.
- class Tms::TmsEvent

  This interface provides functions for events to be sent to clients from the TMS as a whole.

Namespaces

- Tms

Variables

- const BUInt32 Tms::apiVersion = 0

8.112.1 Detailed Description

This file contains the TmsAPi class definitions.

8.113 TmsCycleParam.cc File Reference

```
#include <stdlib.h>
#include <stdint.h>
#include <unistd.h>
#include <errno.h>
#include <math.h>
#include <TmsCycleParam.h>
#include <BFile.h>
#include <BEntry.h>
```

Namespaces

- Tms

8.114 TmsCycleParam.h File Reference

```
#include <TmsLib.h>
```

Classes

- class Tms::CycleParamState
- class Tms::CycleParamEdit

  Cycle Parameter management class.

Namespaces

- Tms

8.115 TmsD.cc File Reference

```
#include <TmsD.h>
```

Namespaces

- Tms

8.116 TmsD.h File Reference

```
#include <Boap.h>
#include <BObj.h>
#include <BDate.h>
#include <BTimeStamp.h>
#include <BComplex.h>
#include <BList.h>
#include <BArray.h>
```

Classes

- class Tms::NameValue
- class Tms::PuChannel

  This class stores a Physical Pick-Up channel id.
- class Tms::PuStatus

  This class stores the status of an individual Pick-Up.
- class Tms::ConfigInfo

  This class describes the configuration of the TMS.
- class Tms::DataInfo

This class defines the data to be acquired and/or fetched.

- class Tms::DataValue

    This is the definition of a single data value.
- class Tms::Data

    This class stores the raw data.
- class Tms::PuStateTable

    This class defines the Pick-Up state table.
- class Tms::CycleParam

    This class defines the parameters for a PS processing cycle.
- class Tms::CycleParamItem
- class Tms::TestCaptureInfo

    This class defines the parameters for a test data capture.
- class Tms::PupeConfig
- class Tms::CycleInformationPeriod

    Cycle information.
- class Tms::CycleInformation
- class Tms::CycleTypeInformationPeriod

    Cycle Type information.
- class Tms::CycleTypeInformation
- class Tms::Simulation

Namespaces

- Tms

Enumerations

- enum Tms::CyclePeriod {
  Tms::CyclePeriodAll, Tms::CyclePeriodCalibration, Tms::CyclePeriodEvent0, Tms::CyclePeriodEvent1,
  Tms::CyclePeriodEvent2, Tms::CyclePeriodEvent3, Tms::CyclePeriodEvent4, Tms::CyclePeriodEvent5,
  Tms::CyclePeriodEvent6, Tms::CyclePeriodEvent7, Tms::CyclePeriodEvent8, Tms::CyclePeriodEvent9 }
- enum Tms::DataType { Tms::DataTypeRaw }
- enum Tms::DataFunction {
  Tms::DataFunctionRaw, Tms::DataFunctionMean, Tms::DataFunctionMeanAll, Tms::DataFunctionMean0,
  Tms::DataFunctionMean1 }
- enum Tms::TestOutput { Tms::TestOutputFrefLocal, Tms::TestOutputPllL1, Tms::TestOutputPllL2 }
- enum Tms::Priority { Tms::PriorityLow, Tms::PriorityNormal, Tms::PriorityHigh }

8.117 TmsEventServerList.cc File Reference

```
#include <stdlib.h>
#include <stdint.h>
#include <TmsEventServerList.h>
```

Namespaces

- Tms

8.118 TmsEventServerList.h File Reference

```
#include <TmsD.h>
#include <TmsC.h>
```

Classes

- class Tms::TmsEventServerList

Namespaces

- Tms

8.119 tmsFunctions.dox File Reference

Namespaces

- Tms

8.120 TmsLib.cc File Reference

```
#include <stdlib.h>
#include <stdint.h>
#include <unistd.h>
#include <math.h>
#include <TmsLib.h>
#include <BDir.h>
#include <BEntry.h>
#include <BFile.h>
#include <TmsCycleParam.h>
```

Namespaces

- Tms

8.121 TmsLib.h File Reference

```
#include <TmsD.h>
#include <TmsC.h>
```

Classes

- union Tms::TmsState

  The Tms State entry.
- union Tms::TmsPhase

  The Tms Phase Table Entry.

- class Tms::CycleParamDb

  Internal CycleParameter management class.

Namespaces

- Tms

Enumerations

- enum Tms::TimingSig {
  Tms::TimingSigClock = 0x01, Tms::TimingSigCycleStart = 0x02, Tms::TimingSigCycleStop = 0x04, Tms::TimingSigCalStart = 0x08,
  Tms::TimingSigCalStop = 0x10, Tms::TimingSigInjection = 0x20, Tms::TimingSigHChange = 0x40, Tms::TimingSigFRef = 0x80 }

  The timing signal bits.
- enum Tms::CaptureClock {
  Tms::ClkAdcDiv_1 = 0x00, Tms::ClkAdcDiv_2 = 0x01, Tms::ClkAdcDiv_5 = 0x02, Tms::ClkAdcDiv_10 = 0x03,
  Tms::ClkAdcDiv_20 = 0x04, Tms::ClkAdcDiv_50 = 0x05, Tms::ClkAdcDiv_100 = 0x06, Tms::ClkAdcDiv_200 = 0x07,
  Tms::ClkAdcDiv_500 = 0x08, Tms::ClkAdcDiv_1000 = 0x09, Tms::ClkAdcDiv_2000 = 0x0A, Tms::ClkAdcDiv_5000 = 0x0B,
  Tms::ClkAdcDiv_10000 = 0x0C, Tms::ClkAdcDiv_20000 = 0x0D, Tms::ClkAdcDiv_50000 = 0x0E, Tms::ClkAdcDiv_100000 = 0x0F,
  Tms::ClkMs = 0x10, Tms::ClkFref = 0x11 }

  The Diagnostics Capture Clock settings.

Variables

- const unsigned int Tms::tmsNumPickups = 40

  The default number of pick ups.
- const unsigned int Tms::tmsPhaseTableSize = 512

  The size of the Phase Table.

8.122 TmsS.cc File Reference

```
#include <TmsC.h>
#include <TmsS.h>
```

Namespaces

- Tms

8.123 TmsT.cc File Reference

```
#include <stdlib.h>
#include <stdint.h>
#include <TmsT.h>
```

Index