streamout

Generated by Doxygen 1.9.2

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 Buffer Class Reference

`#include <Buffer.h>`

Public Member Functions

- Buffer ()
- Buffer (const bit8_t b[ ], const std::size_t &i)
- Buffer (const char b[ ], const std::size_t &i)
- template<typename T >
  Buffer (const std::vector< T > &rawdata)
- template<typename T , std::size_t N>
  Buffer (const std::array< T, N > &rawdata)
- std::size_t size () const
- std::size_t capacity () const
- void set (unsigned char *b)
- bit8_t * begin ()
- bit8_t * end ()
- bit8_t & operator[ ] (const std::size_t &pos)
- bit8_t & operator[ ] (const std::size_t &pos) const
- void setSize (const std::size_t &size)
- virtual ~Buffer ()

4.1.1 Detailed Description

Definition at line 13 of file Buffer.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Buffer() [1/5]

```
Buffer::Buffer ( )  [inline]
```

Definition at line 16 of file Buffer.h.

```
00016 : m_Buffer(nullptr), m_Size(0), m_Capacity(0) {}
```

4.1.2.2 Buffer() [2/5]

```
Buffer::Buffer (
            const bit8_t b[],
            const std::size_t & i )  [inline]
```

Definition at line 17 of file Buffer.h.

```
00017 : m_Buffer(const_cast<bit8_t*>(&b[0])), m_Size(i), m_Capacity(i) {}
```

4.1.2.3 Buffer() [3/5]

```
Buffer::Buffer (
            const char b[],
            const std::size_t & i )  [inline]
```

Definition at line 18 of file Buffer.h.

```
00018 : m_Buffer(const_cast<bit8_t*>(reinterpret_cast<const bit8_t*>(&b[0]))), m_Size(i), m_Capacity(i) {}
```

4.1.2.4 Buffer() [4/5]

```
template<typename T >
Buffer::Buffer (
            const std::vector< T > & rawdata )  [inline]
```

Definition at line 19 of file Buffer.h.

```
00019 : m_Buffer(const_cast<bit8_t*>(reinterpret_cast<const bit8_t*>(rawdata.data()))),
       m_Size(rawdata.size() * sizeof(T)), m_Capacity(rawdata.capacity() * sizeof(T)) {}
```

4.1.2.5 Buffer() [5/5]

```
template<typename T , std::size_t N>
Buffer::Buffer (
            const std::array< T, N > & rawdata )  [inline]
```

Definition at line 20 of file Buffer.h.

```
00020 : m_Buffer(const_cast<bit8_t*>(reinterpret_cast<const bit8_t*>(rawdata.data()))),
       m_Size(rawdata.size() * sizeof(T)), m_Capacity(rawdata.size() * sizeof(T)) {}
```

4.1.2.6 ~Buffer()

```
Buffer::~Buffer ( )  [virtual]
```

Definition at line 8 of file Buffer.cc.
```
00008 {}
```

4.1.3 Member Function Documentation

4.1.3.1 begin()

```
bit8_t * Buffer::begin ( )  [inline]
```

Definition at line 26 of file Buffer.h.
```
00026 { return m_Buffer; }
```

4.1.3.2 capacity()

```
std::size_t Buffer::capacity ( ) const  [inline]
```

Definition at line 23 of file Buffer.h.
```
00023 { return m_Capacity; }
```

4.1.3.3 end()

```
bit8_t * Buffer::end ( )  [inline]
```

Definition at line 27 of file Buffer.h.
```
00027 { return m_Buffer + m_Size; }
```

4.1.3.4 operator[]() [1/2]

```
bit8_t & Buffer::operator[] (
            const std::size_t & pos )  [inline]
```

Definition at line 28 of file Buffer.h.
```
00028 { return m_Buffer[pos]; }
```

4.1.3.5 operator[]() [2/2]

```
bit8_t & Buffer::operator[] (
            const std::size_t & pos ) const  [inline]
```

Definition at line 29 of file Buffer.h.

```
00029 { return m_Buffer[pos]; }
```

4.1.3.6 set()

```
void Buffer::set (
            unsigned char * b )  [inline]
```

Definition at line 25 of file Buffer.h.

```
00025 { m_Buffer = b; }
```

4.1.3.7 setSize()

```
void Buffer::setSize (
            const std::size_t & size )  [inline]
```

Definition at line 31 of file Buffer.h.

```
00031 { m_Size = size; }
```

4.1.3.8 size()

```
std::size_t Buffer::size ( ) const  [inline]
```

Definition at line 22 of file Buffer.h.

```
00022 { return m_Size; }
```

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

- /home/runner/work/streamout/streamout/libs/core/include/Buffer.h
- /home/runner/work/streamout/streamout/libs/core/src/Buffer.cc

4.2 ROOTtreeDest::DATA Struct Reference

```
#include <ROOTtreeDest.h>
```

Public Attributes

- UInt_t DIFid
- UInt_t ASICid
- UInt_t CHANNELid
- UInt_t Thresh
- UInt_t DTC
- UInt_t GTC
- UInt_t DIF_BCID
- UInt_t frame_BCID
- UInt_t timeStamp
- ULong64_t AbsoluteBCID

4.2.1 Detailed Description

Definition at line 16 of file ROOTtreeDest.h.

4.2.2 Member Data Documentation

4.2.2.1 AbsoluteBCID

```
ULong64_t ROOTtreeDest::DATA::AbsoluteBCID
```

Definition at line 21 of file ROOTtreeDest.h.

4.2.2.2 ASICid

```
UInt_t ROOTtreeDest::DATA::ASICid
```

Definition at line 18 of file ROOTtreeDest.h.

4.2.2.3 CHANNELid

```
UInt_t ROOTtreeDest::DATA::CHANNELid
```

Definition at line 18 of file ROOTtreeDest.h.

4.2.2.4 DIF_BCID

```
UInt_t ROOTtreeDest::DATA::DIF_BCID
```

Definition at line 20 of file ROOTtreeDest.h.

4.2.2.5 DIFid

```
UInt_t ROOTtreeDest::DATA::DIFid
```

Definition at line 18 of file ROOTtreeDest.h.

4.2.2.6 DTC

```
UInt_t ROOTtreeDest::DATA::DTC
```

Definition at line 20 of file ROOTtreeDest.h.

4.2.2.7 frame_BCID

```
UInt_t ROOTtreeDest::DATA::frame_BCID
```

Definition at line 20 of file ROOTtreeDest.h.

4.2.2.8 GTC

```
UInt_t ROOTtreeDest::DATA::GTC
```

Definition at line 20 of file ROOTtreeDest.h.

4.2.2.9 Thresh

```
UInt_t ROOTtreeDest::DATA::Thresh
```

Definition at line 19 of file ROOTtreeDest.h.

4.2.2.10 timeStamp

```
UInt_t ROOTtreeDest::DATA::timeStamp
```

Definition at line 20 of file ROOTtreeDest.h.

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

- /home/runner/work/streamout/streamout/libs/interface/ROOT/include/ROOTtreeDest.h

4.3 DIFPtr Class Reference

```
#include <DIFPtr.h>
```

Public Member Functions

- DIFPtr (unsigned char *p, const std::uint32_t &max_size)
- unsigned char * getPtr ()
- std::uint32_t getGetFramePtrReturn ()
- std::vector< unsigned char * > & getFramesVector ()
- std::vector< unsigned char * > & getLinesVector ()
- std::uint32_t getID ()
- std::uint32_t getDTC ()
- std::uint32_t getGTC ()
- std::uint64_t getAbsoluteBCID ()
- std::uint32_t getBCID ()
- std::uint32_t getLines ()
- bool hasLine (uint32_t line)
- std::uint32_t getTASU1 ()
- std::uint32_t getTASU2 ()
- std::uint32_t getTDIF ()
- float getTemperatureDIF ()
- float getTemperatureASU1 ()
- float getTemperatureASU2 ()
- bool hasTemperature ()
- bool hasAnalogReadout ()
- std::uint32_t getNumberOfFrames ()
- unsigned char * getFramePtr (uint32_t i)
- std::uint32_t getFrameAsicHeader (uint32_t i)
- std::uint32_t getFrameBCID (uint32_t i)
- std::uint32_t getFrameTimeToTrigger (uint32_t i)
- bool getFrameLevel (uint32_t i, uint32_t ipad, uint32_t ilevel)
- uint32_t getDIFid ()
- uint32_t getASICid (uint32_t i)
- uint32_t getThresholdStatus (uint32_t i, uint32_t ipad)

4.3.1 Detailed Description

Definition at line 10 of file DIFPtr.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 DIFPtr()

```
DIFPtr::DIFPtr (
            unsigned char * p,
            const std::uint32_t & max_size )
```

Definition at line 11 of file DIFPtr.cc.

```
00011                                                   : theDIF_(p), theSize_(max_size)
00012 {
00013   theFrames_.clear();
00014   theLines_.clear();
00015   try
00016   {
00017     theGetFramePtrReturn_ = DIFUnpacker::getFramePtr(theFrames_, theLines_, theSize_, theDIF_);
00018   }
00019   catch(const std::string& e)
00020   {
00021     spdlog::get("streamout")->error(" DIF {} T ? {} {}", getID(), hasTemperature(), e);
00022   }
00023 }
```

4.3.3 Member Function Documentation

4.3.3.1 getAbsoluteBCID()

```
std::uint64_t DIFPtr::getAbsoluteBCID ( )  [inline]
```

Definition at line 21 of file DIFPtr.h.

```
00021 { return DIFUnpacker::getAbsoluteBCID(theDIF_); }
```

4.3.3.2 getASICid()

```
uint32_t DIFPtr::getASICid (
            uint32_t i )  [inline]
```

Definition at line 48 of file DIFPtr.h.

```
00048 { return getFrameAsicHeader(i) & 0xFF; }
```

4.3.3.3 getBCID()

```
std::uint32_t DIFPtr::getBCID ( )  [inline]
```

Definition at line 22 of file DIFPtr.h.

```
00022 { return DIFUnpacker::getBCID(theDIF_); }
```

4.3.3.4 getDIFid()

```
uint32_t DIFPtr::getDIFid ( )  [inline]
```

Definition at line 47 of file DIFPtr.h.

```
00047 { return getID() & 0xFF; }
```

4.3.3.5 getDTC()

```
std::uint32_t DIFPtr::getDTC ( )  [inline]
```

Definition at line 19 of file DIFPtr.h.

```
00019 { return DIFUnpacker::getDTC(theDIF_); }
```

4.3.3.6 getFrameAsicHeader()

```
std::uint32_t DIFPtr::getFrameAsicHeader (
            uint32_t i )  [inline]
```

Definition at line 35 of file DIFPtr.h.

```
00035 { return DIFUnpacker::getFrameAsicHeader(theFrames_[i]); }
```

4.3.3.7 getFrameBCID()

```
std::uint32_t DIFPtr::getFrameBCID (
            uint32_t i )  [inline]
```

Definition at line 36 of file DIFPtr.h.

```
00036 { return DIFUnpacker::getFrameBCID(theFrames_[i]); }
```

4.3.3.8 getFrameLevel()

```
bool DIFPtr::getFrameLevel (
            uint32_t i,
            uint32_t ipad,
            uint32_t ilevel )  [inline]
```

Definition at line 38 of file DIFPtr.h.

```
00038 { return DIFUnpacker::getFrameLevel(theFrames_[i], ipad, ilevel); }
```

4.3.3.9 getFramePtr()

```
unsigned char * DIFPtr::getFramePtr (
            uint32_t i )  [inline]
```

Definition at line 34 of file DIFPtr.h.

```
00034 { return theFrames_[i]; }
```

4.3.3.10 getFramesVector()

```
std::vector< unsigned char * > & DIFPtr::getFramesVector ( )  [inline]
```

Definition at line 16 of file DIFPtr.h.

```
00016 { return theFrames_; }
```

4.3.3.11 getFrameTimeToTrigger()

```
std::uint32_t DIFPtr::getFrameTimeToTrigger (
            uint32_t i )  [inline]
```

Definition at line 37 of file DIFPtr.h.

```
00037 { return getBCID() - getFrameBCID(i); }
```

4.3.3.12 getGetFramePtrReturn()

```
std::uint32_t DIFPtr::getGetFramePtrReturn ( )  [inline]
```

Definition at line 15 of file DIFPtr.h.

```
00015 { return theGetFramePtrReturn_; }
```

4.3.3.13 getGTC()

```
std::uint32_t DIFPtr::getGTC ( )  [inline]
```

Definition at line 20 of file DIFPtr.h.

```
00020 { return DIFUnpacker::getGTC(theDIF_); }
```

4.3.3.14 getID()

```
std::uint32_t DIFPtr::getID ( )  [inline]
```

Definition at line 18 of file DIFPtr.h.

```
00018 { return DIFUnpacker::getID(theDIF_); }
```

4.3.3.15 getLines()

```
std::uint32_t DIFPtr::getLines ( )  [inline]
```

Definition at line 23 of file DIFPtr.h.

```
00023 { return DIFUnpacker::getLines(theDIF_); }
```

4.3.3.16 getLinesVector()

```
std::vector< unsigned char * > & DIFPtr::getLinesVector ( )  [inline]
```

Definition at line 17 of file DIFPtr.h.

```
00017 { return theLines_; }
```

4.3.3.17 getNumberOfFrames()

```
std::uint32_t DIFPtr::getNumberOfFrames ( )  [inline]
```

Definition at line 33 of file DIFPtr.h.

```
00033 { return theFrames_.size(); }
```

4.3.3.18 getPtr()

```
unsigned char * DIFPtr::getPtr ( )  [inline]
```

Definition at line 14 of file DIFPtr.h.

```
00014 { return theDIF_; }
```

4.3.3.19 getTASU1()

```
std::uint32_t DIFPtr::getTASU1 ( )  [inline]
```

Definition at line 25 of file DIFPtr.h.

```
00025 { return DIFUnpacker::getTASU1(theDIF_); }
```

4.3.3.20 getTASU2()

```
std::uint32_t DIFPtr::getTASU2 ( )  [inline]
```

Definition at line 26 of file DIFPtr.h.

```
00026 { return DIFUnpacker::getTASU2(theDIF_); }
```

4.3.3.21 getTDIF()

```
std::uint32_t DIFPtr::getTDIF ( )  [inline]
```

Definition at line 27 of file DIFPtr.h.

```
00027 { return DIFUnpacker::getTDIF(theDIF_); }
```

4.3.3.22 getTemperatureASU1()

```
float DIFPtr::getTemperatureASU1 ( )  [inline]
```

Definition at line 29 of file DIFPtr.h.

```
00029 { return (getTASU1() >> 3) * 0.0625; }
```

4.3.3.23 getTemperatureASU2()

```
float DIFPtr::getTemperatureASU2 ( )  [inline]
```

Definition at line 30 of file DIFPtr.h.

```
00030 { return (getTASU2() >> 3) * 0.0625; }
```

4.3.3.24 getTemperatureDIF()

```
float DIFPtr::getTemperatureDIF ( )  [inline]
```

Definition at line 28 of file DIFPtr.h.

```
00028 { return 0.508 * getTDIF() - 9.659; }
```

4.3.3.25 getThresholdStatus()

```
uint32_t DIFPtr::getThresholdStatus (
            uint32_t i,
            uint32_t ipad )  [inline]
```

Definition at line 49 of file DIFPtr.h.

```
00049 { return (((uint32_t)getFrameLevel(i, ipad, 1)) << 1) | ((uint32_t)getFrameLevel(i, ipad, 0)); }
```

4.3.3.26 hasAnalogReadout()

```
bool DIFPtr::hasAnalogReadout ( )  [inline]
```

Definition at line 32 of file DIFPtr.h.

```
00032 { return DIFUnpacker::hasAnalogReadout(theDIF_); }
```

4.3.3.27 hasLine()

```
bool DIFPtr::hasLine (
            uint32_t line )  [inline]
```

Definition at line 24 of file DIFPtr.h.

```
00024 { return DIFUnpacker::hasLine(line, theDIF_); }
```

4.3.3.28 hasTemperature()

```
bool DIFPtr::hasTemperature ( )  [inline]
```

Definition at line 31 of file DIFPtr.h.

```
00031 { return DIFUnpacker::hasTemperature(theDIF_); }
```

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

- /home/runner/work/streamout/streamout/libs/core/include/DIFPtr.h
- /home/runner/work/streamout/streamout/libs/core/src/DIFPtr.cc

4.4 DIFSlowControl Class Reference

Handler of DIF Slow Control info.

```
#include <DIFSlowControl.h>
```

Public Member Functions

- DIFSlowControl (const std::uint8_t &version, const std::uint8_t &DIFid, unsigned char *buf)
  Constructor.
- std::uint8_t getDIFId ()
  get DIF id
- std::map< int, std::map< std::string, int > > getChipsMap ()
  Get chips map.
- std::map< std::string, int > getChipSlowControl (const int &asicid)
  Get one chip map.
- int getChipSlowControl (const std::int8_t &asicid, const std::string &param)
  Get one Chip value.
- void Dump ()
  print out full map

4.4.1 Detailed Description

Handler of DIF Slow Control info.

Author

L.Mirabito

Date

March 2010

Version

1.0

Definition at line 19 of file DIFSlowControl.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 DIFSlowControl()

```
DIFSlowControl::DIFSlowControl (
            const std::uint8_t & version,
            const std::uint8_t & DIFid,
            unsigned char * buf )
```

Constructor.

Parameters

version	Data format version
DIFid	DIF id
buf	Pointer to the Raw data buffer

Definition at line 10 of file DIFSlowControl.cc.

```
00010
      : m_Version(version), m_DIFId(DIfId), m_AsicType(2)
00011 {
00012   if(cbuf[0] != 0xb1) return;
00013   int header_shift{6};
00014   if(m_Version < 8) m_NbrAsic = cbuf[5];
00015   else
00016   {
00017     m_DIFId      = cbuf[1];
00018     m_NbrAsic    = cbuf[2];
00019     header_shift = 3;
00020   }
00021   int size_hardroc1 = m_NbrAsic * 72 + header_shift + 1;
00022   if(cbuf[size_hardroc1 - 1] != 0xa1) size_hardroc1 = 0;
00023
00024   int size_hardroc2 = m_NbrAsic * 109 + header_shift + 1;
00025   if(cbuf[size_hardroc2 - 1] != 0xa1) size_hardroc2 = 0;
```

```
00026   if(size_hardroc1 != 0)
00027   {
00028     FillHR1(header_shift, cbuf);
00029     m_AsicType = 1;
00030   }
00031   else if(size_hardroc2 != 0)
00032     FillHR2(header_shift, cbuf);
00033   else
00034     return;
00035 }
```

4.4.3 Member Function Documentation

4.4.3.1 Dump()

```
void DIFSlowControl::Dump ( )
```

print out full map

Definition at line 45 of file DIFSlowControl.cc.

```
00046 {
00047   for(std::map<int, std::map<std::string, int»::iterator it = m_MapSC.begin(); it != m_MapSC.end();
       it++)
00048   {
00049     std::cout « "ASIC " « it->first « std::endl;
00050     for(std::map<std::string, int>::iterator jt = (it->second).begin(); jt != (it->second).end();
       jt++) std::cout « jt->first « " : " « jt->second « std::endl;
00051   }
00052 }
```

4.4.3.2 getChipSlowControl() [1/2]

```
std::map< std::string, int > DIFSlowControl::getChipSlowControl (
            const int & asicid )  [inline]
```

Get one chip map.

Parameters

asicid	ASIC ID

Returns

a map of <string (parameter name),int (parameter value) >

Definition at line 41 of file DIFSlowControl.cc.

```
00041 { return m_MapSC[asicid]; }
```

4.4.3.3 getChipSlowControl() [2/2]

```
int DIFSlowControl::getChipSlowControl (
            const std::int8_t & asicid,
            const std::string & param )  [inline]
```

Get one Chip value.

Parameters

asicid	ASic ID
param	Parameter name

Definition at line 43 of file DIFSlowControl.cc.

```
00043 { return getChipSlowControl(asicid)[param]; }
```

4.4.3.4 getChipsMap()

```
std::map< int, std::map< std::string, int > > DIFSlowControl::getChipsMap ( )  [inline]
```

Get chips map.

Returns

a map of < Asic Id, map of <string (parameter name),int (parameter value) >

Definition at line 39 of file DIFSlowControl.cc.

```
00039 { return m_MapSC; }
```

4.4.3.5 getDIFId()

```
std::uint8_t DIFSlowControl::getDIFId ( )  [inline]
```

get DIF id

Definition at line 37 of file DIFSlowControl.cc.

```
00037 { return m_DIFId; }
```

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

- /home/runner/work/streamout/streamout/libs/core/include/DIFSlowControl.h
- /home/runner/work/streamout/streamout/libs/core/src/DIFSlowControl.cc

4.5 DIFUnpacker Class Reference

```
#include <DIFUnpacker.h>
```

Static Public Member Functions

- static std::uint64_t GrayToBin (const std::uint64_t &n)
- static std::uint32_t getStartOfDIF (const unsigned char *cbuf, const std::uint32_t &size_buf, const std::uint32_t &start=92)
- static std::uint32_t getID (const unsigned char *cb, const std::uint32_t &idx=0)
- static std::uint32_t getDTC (const unsigned char *cb, const std::uint32_t &idx=0)
- static std::uint32_t getGTC (const unsigned char *cb, const std::uint32_t &idx=0)
- static std::uint64_t getAbsoluteBCID (const unsigned char *cb, const std::uint32_t &idx=0)
- static std::uint32_t getBCID (const unsigned char *cb, const std::uint32_t &idx=0)
- static std::uint32_t getLines (const unsigned char *cb, const std::uint32_t &idx=0)
- static bool hasLine (const std::uint32_t &line, const unsigned char *cb, const std::uint32_t &idx=0)
- static std::uint32_t getTASU1 (const unsigned char *cb, const std::uint32_t &idx=0)
- static std::uint32_t getTASU2 (const unsigned char *cb, const std::uint32_t &idx=0)
- static std::uint32_t getTDIF (const unsigned char *cb, const std::uint32_t &idx=0)
- static bool hasTemperature (const unsigned char *cb, const std::uint32_t &idx=0)
- static bool hasAnalogReadout (const unsigned char *cb, const std::uint32_t &idx=0)
- static std::uint32_t getFrameAsicHeader (const unsigned char *framePtr)
- static std::uint32_t getFrameBCID (const unsigned char *framePtr)
- static bool getFramePAD (const unsigned char *framePtr, const std::uint32_t &ip)
- static bool getFrameLevel (const unsigned char *framePtr, const std::uint32_t &ip, const std::uint32_t &level)
- static std::uint32_t getAnalogPtr (std::vector< unsigned char * > &vLines, unsigned char *cb, const std::uint32_t &idx=0)
- static std::uint32_t getFramePtr (std::vector< unsigned char * > &vFrame, std::vector< unsigned char * > &vLines, const std::uint32_t &max_size, unsigned char *cb, const std::uint32_t &idx=0)
- static void dumpFrameOld (const unsigned char *buf)
- static std::uint32_t swap_bytes (const unsigned char *buf)

4.5.1 Detailed Description

Definition at line 10 of file DIFUnpacker.h.

4.5.2 Member Function Documentation

4.5.2.1 dumpFrameOld()

```
void DIFUnpacker::dumpFrameOld (
            const unsigned char * buf )  [static]
```

Definition at line 140 of file DIFUnpacker.cc.

```
00141 {
00142   bool         PAD[128];
00143   bool         l0[64];
00144   bool         l1[64];
00145   std::uint8_t un{1};
00146   for(std::size_t ip = 0; ip < 128; ip++) { PAD[ip] = false; }  // init PADs
00147   std::uint32_t idx1{4};
00148   for(int ik = 0; ik < 4; ik++)
00149   {
00150     std::uint32_t PadEtat{swap_bytes(&buf[idx1])};
00151     idx1 += 4;
00152     for(int e = 0; e < 32; e++)
00153     {
00154       PAD[((3 - ik) * 32) + (31 - e)] = PadEtat & un;  // binary operation
```

```
00155       PadEtat                         = PadEtat » 1;  // décalage des bit de 1
00156     }
00157   }
00158   // fill bool arrays
00159   for(int p = 0; p < 64; p++)
00160   {
00161     l0[p] = static_cast<bool>(PAD[(2 * p)]);      // _Lev0 (PAD paire)
00162     l1[p] = static_cast<bool>(PAD[(2 * p) + 1]);  // _Lev1 (PAD impaires)
00163   }
00164   std::bitset<64> bs0(0);
00165   std::bitset<64> bs1(0);
00166   for(std::uint32_t ip = 0; ip < 64; ip++)
00167   {
00168     bs0.set(ip, l0[ip]);
00169     bs1.set(ip, l1[ip]);
00170   }
00171   std::cout « "\t \t" « bs0 « std::endl;
00172   std::cout « "\t \t" « bs1 « std::endl;
00173 }
```

4.5.2.2 getAbsoluteBCID()

```
std::uint64_t DIFUnpacker::getAbsoluteBCID (
            const unsigned char * cb,
            const std::uint32_t & idx = 0 )  [static]
```

Definition at line 47 of file DIFUnpacker.cc.

```
00048 {
00049   std::uint64_t Shift{16777216ULL};  // to shift the value from the 24 first bits
00050   std::uint64_t pos{idx + DU::ABCID_SHIFT};
00051   std::uint64_t LBC = ((cb[pos] « 16) | (cb[pos + 1] « 8) | (cb[pos + 2])) * Shift + ((cb[pos + 3] «
       16) | (cb[pos + 4] « 8) | (cb[pos + 5]));
00052   return LBC;
00053 }
```

4.5.2.3 getAnalogPtr()

```
std::uint32_t DIFUnpacker::getAnalogPtr (
            std::vector< unsigned char * > & vLines,
            unsigned char * cb,
            const std::uint32_t & idx = 0 )  [static]
```

Definition at line 86 of file DIFUnpacker.cc.

```
00087 {
00088   std::uint32_t fshift{idx};
00089   if(cb[fshift] != DU::START_OF_LINES) return fshift;
00090   fshift++;
00091   while(cb[fshift] != DU::END_OF_LINES)
00092   {
00093     vLines.push_back(&cb[fshift]);
00094     std::uint32_t nchip{cb[fshift]};
00095     fshift += 1 + nchip * 64 * 2;
00096   }
00097   return fshift++;
00098 }
```

4.5.2.4 getBCID()

```
std::uint32_t DIFUnpacker::getBCID (
            const unsigned char * cb,
            const std::uint32_t & idx = 0 )  [static]
```

Definition at line 55 of file DIFUnpacker.cc.

```
00055 { return (cb[idx + DU::BCID_SHIFT] « 16) + (cb[idx + DU::BCID_SHIFT + 1] « 8) + cb[idx +
       DU::BCID_SHIFT + 2]; }
```

4.5.2.5 getDTC()

```
std::uint32_t DIFUnpacker::getDTC (
            const unsigned char * cb,
            const std::uint32_t & idx = 0 )  [static]
```

Definition at line 43 of file DIFUnpacker.cc.

```
00043 { return (cb[idx + DU::DTC_SHIFT] « 24) + (cb[idx + DU::DTC_SHIFT + 1] « 16) + (cb[idx + DU::DTC_SHIFT
       + 2] « 8) + cb[idx + DU::DTC_SHIFT + 3]; }
```

4.5.2.6 getFrameAsicHeader()

```
std::uint32_t DIFUnpacker::getFrameAsicHeader (
            const unsigned char * framePtr )  [static]
```

Definition at line 70 of file DIFUnpacker.cc.

```
00070 { return (framePtr[DU::FRAME_ASIC_HEADER_SHIFT]); }
```

4.5.2.7 getFrameBCID()

```
std::uint32_t DIFUnpacker::getFrameBCID (
            const unsigned char * framePtr )  [static]
```

Definition at line 72 of file DIFUnpacker.cc.

```
00073 {
00074   std::uint32_t igray = (framePtr[DU::FRAME_BCID_SHIFT] « 16) + (framePtr[DU::FRAME_BCID_SHIFT + 1] «
       8) + framePtr[DU::FRAME_BCID_SHIFT + 2];
00075   return DIFUnpacker::GrayToBin(igray);
00076 }
```

4.5.2.8 getFrameLevel()

```
bool DIFUnpacker::getFrameLevel (
            const unsigned char * framePtr,
            const std::uint32_t & ip,
            const std::uint32_t & level )  [static]
```

Definition at line 84 of file DIFUnpacker.cc.

```
00084 { return ((framePtr[DU::FRAME_DATA_SHIFT + ((3 - ip / 16) * 4 + (ip % 16) / 4)] » (7 - (((ip % 16) %
       4) * 2 + level))) & 0x1); }
```

4.5.2.9 getFramePAD()

```
bool DIFUnpacker::getFramePAD (
            const unsigned char * framePtr,
            const std::uint32_t & ip )  [static]
```

Definition at line 78 of file DIFUnpacker.cc.

```
00079 {
00080   std::uint32_t* iframe{(std::uint32_t*)&framePtr[DU::FRAME_DATA_SHIFT]};
00081   return ((iframe[3 - ip / 32] » (ip % 32)) & 0x1);
00082 }
```

4.5.2.10 getFramePtr()

```
std::uint32_t DIFUnpacker::getFramePtr (
            std::vector< unsigned char * > & vFrame,
            std::vector< unsigned char * > & vLines,
            const std::uint32_t & max_size,
            unsigned char * cb,
            const std::uint32_t & idx = 0 )  [static]
```

Definition at line 100 of file DIFUnpacker.cc.

```
00101 {
00102   std::uint32_t fshift{0};
00103   if(DATA_FORMAT_VERSION >= 13)
00104   {
00105     fshift = idx + DU::LINES_SHIFT + 1;
00106     if(DIFUnpacker::hasTemperature(cb, idx)) fshift = idx + DU::TDIF_SHIFT + 1;
        // jenlev 1
00107     if(DIFUnpacker::hasAnalogReadout(cb, idx)) fshift = DIFUnpacker::getAnalogPtr(vLines, cb, fshift);
        // to be implemented
00108   }
00109   else
00110     std::uint32_t fshift = idx + DU::BCID_SHIFT + 3;
00111   if(cb[fshift] != DU::START_OF_FRAME)
00112   {
00113     std::cout « "This is not a start of frame " « cb[fshift] « "\n";
00114     return fshift;
00115   }
00116   do {
00117     // printf("fshift %d and %d \n",fshift,max_size);
00118     if(cb[fshift] == DU::END_OF_DIF) return fshift;
00119     if(cb[fshift] == DU::START_OF_FRAME) fshift++;
00120     if(cb[fshift] == DU::END_OF_FRAME)
00121     {
00122       fshift++;
00123       continue;
00124     }
00125     std::uint32_t header = DIFUnpacker::getFrameAsicHeader(&cb[fshift]);
00126     if(header == DU::END_OF_FRAME) return (fshift + 2);
00127     // std::cout«header«" "«fshift«std::endl;
00128     if(header < 1 || header > 48) { throw header + " Header problem " + fshift; }
00129     vFrame.push_back(&cb[fshift]);
00130     fshift += DU::FRAME_SIZE;
00131     if(fshift > max_size)
00132     {
00133       std::cout « "fshift " « fshift « " exceed " « max_size « "\n";
00134       return fshift;
00135     }
00136     if(cb[fshift] == DU::END_OF_FRAME) fshift++;
00137   } while(true);
00138 }
```

4.5.2.11 getGTC()

```
std::uint32_t DIFUnpacker::getGTC (
            const unsigned char * cb,
            const std::uint32_t & idx = 0 )  [static]
```

Definition at line 45 of file DIFUnpacker.cc.

```
00045 { return (cb[idx + DU::GTC_SHIFT] « 24) + (cb[idx + DU::GTC_SHIFT + 1] « 16) + (cb[idx + DU::GTC_SHIFT
      + 2] « 8) + cb[idx + DU::GTC_SHIFT + 3]; }
```

4.5.2.12 getID()

```
std::uint32_t DIFUnpacker::getID (
            const unsigned char * cb,
            const std::uint32_t & idx = 0 )  [static]
```

Definition at line 41 of file DIFUnpacker.cc.

```
00041 { return cb[idx + DU::ID_SHIFT]; }
```

4.5.2.13 getLines()

```
std::uint32_t DIFUnpacker::getLines (
            const unsigned char * cb,
            const std::uint32_t & idx = 0 )  [static]
```

Definition at line 56 of file DIFUnpacker.cc.

```
00056 { return (cb[idx + DU::LINES_SHIFT] » 4) & 0x5; }
```

4.5.2.14 getStartOfDIF()

```
std::uint32_t DIFUnpacker::getStartOfDIF (
            const unsigned char * cbuf,
            const std::uint32_t & size_buf,
            const std::uint32_t & start = 92 )  [static]
```

Definition at line 28 of file DIFUnpacker.cc.

```
00029 {
00030   std::uint32_t id0{0};
00031   for(std::uint32_t i = start; i < size_buf; i++)
00032   {
00033     if(cbuf[i] != DU::START_OF_DIF && cbuf[i] != DU::START_OF_DIF_TEMP) continue;
00034     id0 = i;
00035     // if (cbuf[id0+DU::ID_SHIFT]>0xFF) continue;
00036     break;
00037   }
00038   return id0;
00039 }
```

4.5.2.15 getTASU1()

```
std::uint32_t DIFUnpacker::getTASU1 (
            const unsigned char * cb,
            const std::uint32_t & idx = 0 )  [static]
```

Definition at line 60 of file DIFUnpacker.cc.

```
00060 { return (cb[idx + DU::TASU1_SHIFT] << 24) + (cb[idx + DU::TASU1_SHIFT + 1] << 16) + (cb[idx +
       DU::TASU1_SHIFT + 2] << 8) + cb[idx + DU::TASU1_SHIFT + 3]; }
```

4.5.2.16 getTASU2()

```
std::uint32_t DIFUnpacker::getTASU2 (
            const unsigned char * cb,
            const std::uint32_t & idx = 0 )  [static]
```

Definition at line 62 of file DIFUnpacker.cc.

```
00062 { return (cb[idx + DU::TASU2_SHIFT] << 24) + (cb[idx + DU::TASU2_SHIFT + 1] << 16) + (cb[idx +
       DU::TASU2_SHIFT + 2] << 8) + cb[idx + DU::TASU2_SHIFT + 3]; }
```

4.5.2.17 getTDIF()

```
std::uint32_t DIFUnpacker::getTDIF (
            const unsigned char * cb,
            const std::uint32_t & idx = 0 )  [static]
```

Definition at line 64 of file DIFUnpacker.cc.

```
00064 { return (cb[idx + DU::TDIF_SHIFT]); }
```

4.5.2.18 GrayToBin()

```
std::uint64_t DIFUnpacker::GrayToBin (
            const std::uint64_t & n )  [static]
```

Definition at line 13 of file DIFUnpacker.cc.

```
00014 {
00015   std::uint64_t ish{1};
00016   std::uint64_t anss{n};
00017   std::uint64_t idiv{0};
00018   std::uint64_t ishmax{sizeof(std::uint64_t) * 8};
00019   while(true)
00020   {
00021     idiv = anss >> ish;
00022     anss ^= idiv;
00023     if(idiv <= 1 || ish == ishmax) return anss;
00024     ish <<= 1;
00025   }
00026 }
```

4.5.2.19 hasAnalogReadout()

```
bool DIFUnpacker::hasAnalogReadout (
            const unsigned char * cb,
            const std::uint32_t & idx = 0 )  [static]
```

Definition at line 68 of file DIFUnpacker.cc.

```
00068 { return (DIFUnpacker::getLines(cb, idx) != 0); }
```

4.5.2.20 hasLine()

```
bool DIFUnpacker::hasLine (
            const std::uint32_t & line,
            const unsigned char * cb,
            const std::uint32_t & idx = 0 )  [static]
```

Definition at line 58 of file DIFUnpacker.cc.

```
00058 { return ((cb[idx + DU::LINES_SHIFT] >> line) & 0x1); }
```

4.5.2.21 hasTemperature()

```
bool DIFUnpacker::hasTemperature (
            const unsigned char * cb,
            const std::uint32_t & idx = 0 )  [static]
```

Definition at line 66 of file DIFUnpacker.cc.

```
00066 { return (cb[idx] == DU::START_OF_DIF_TEMP); }
```

4.5.2.22 swap_bytes()

```
std::uint32_t DIFUnpacker::swap_bytes (
            const unsigned char * buf )  [static]
```

Definition at line 175 of file DIFUnpacker.cc.

```
00176 {
00177   unsigned char Swapped[4];
00178   for(std::size_t i = 0; i < 4; i++) Swapped[i] = buf[4 - 1 - i];
00179   return *reinterpret_cast<std::uint32_t*>(&Swapped[0]);
00180 }
```

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

- /home/runner/work/streamout/streamout/libs/core/include/DIFUnpacker.h
- /home/runner/work/streamout/streamout/libs/core/src/DIFUnpacker.cc

4.6 DU Class Reference

`#include <Words.h>`

Static Public Attributes

- static const std::uint32_t START_OF_DIF {0xB0}
- static const std::uint32_t START_OF_DIF_TEMP {0xBB}
- static const std::uint32_t END_OF_DIF {0xA0}
- static const std::uint32_t START_OF_LINES {0xC4}
- static const std::uint32_t END_OF_LINES {0xD4}
- static const std::uint32_t START_OF_FRAME {0xB4}
- static const std::uint32_t END_OF_FRAME {0xA3}
- static const std::uint32_t ID_SHIFT {1}
- static const std::uint32_t DTC_SHIFT {2}
- static const std::uint32_t GTC_SHIFT {10}
- static const std::uint32_t ABCID_SHIFT {14}
- static const std::uint32_t BCID_SHIFT {20}
- static const std::uint32_t LINES_SHIFT {23}
- static const std::uint32_t TASU1_SHIFT {24}
- static const std::uint32_t TASU2_SHIFT {28}
- static const std::uint32_t TDIF_SHIFT {32}
- static const std::uint32_t FRAME_ASIC_HEADER_SHIFT {0}
- static const std::uint32_t FRAME_BCID_SHIFT {1}
- static const std::uint32_t FRAME_DATA_SHIFT {4}
- static const std::uint32_t FRAME_SIZE {20}

4.6.1 Detailed Description

Definition at line 7 of file Words.h.

4.6.2 Member Data Documentation

4.6.2.1 ABCID_SHIFT

```
const std::uint32_t DU::ABCID_SHIFT {14}  [static]
```

Definition at line 22 of file Words.h.

4.6.2.2 BCID_SHIFT

```
const std::uint32_t DU::BCID_SHIFT {20}  [static]
```

Definition at line 23 of file Words.h.

4.6.2.3 DTC_SHIFT

```
const std::uint32_t DU::DTC_SHIFT {2}  [static]
```

Definition at line 20 of file Words.h.

4.6.2.4 END_OF_DIF

```
const std::uint32_t DU::END_OF_DIF {0xA0}  [static]
```

Definition at line 12 of file Words.h.

4.6.2.5 END_OF_FRAME

```
const std::uint32_t DU::END_OF_FRAME {0xA3}  [static]
```

Definition at line 17 of file Words.h.

4.6.2.6 END_OF_LINES

```
const std::uint32_t DU::END_OF_LINES {0xD4}  [static]
```

Definition at line 14 of file Words.h.

4.6.2.7 FRAME_ASIC_HEADER_SHIFT

```
const std::uint32_t DU::FRAME_ASIC_HEADER_SHIFT {0}  [static]
```

Definition at line 29 of file Words.h.

4.6.2.8 FRAME_BCID_SHIFT

```
const std::uint32_t DU::FRAME_BCID_SHIFT {1}  [static]
```

Definition at line 30 of file Words.h.

4.6.2.9 FRAME_DATA_SHIFT

```
const std::uint32_t DU::FRAME_DATA_SHIFT {4}  [static]
```

Definition at line 31 of file Words.h.

4.6.2.10 FRAME_SIZE

```
const std::uint32_t DU::FRAME_SIZE {20}  [static]
```

Definition at line 32 of file Words.h.

4.6.2.11 GTC_SHIFT

```
const std::uint32_t DU::GTC_SHIFT {10}  [static]
```

Definition at line 21 of file Words.h.

4.6.2.12 ID_SHIFT

```
const std::uint32_t DU::ID_SHIFT {1}  [static]
```

Definition at line 19 of file Words.h.

4.6.2.13 LINES_SHIFT

```
const std::uint32_t DU::LINES_SHIFT {23}  [static]
```

Definition at line 24 of file Words.h.

4.6.2.14 START_OF_DIF

```
const std::uint32_t DU::START_OF_DIF {0xB0}  [static]
```

Definition at line 10 of file Words.h.

4.6.2.15 START_OF_DIF_TEMP

```
const std::uint32_t DU::START_OF_DIF_TEMP {0xBB}  [static]
```

Definition at line 11 of file Words.h.

4.6.2.16 START_OF_FRAME

```
const std::uint32_t DU::START_OF_FRAME {0xB4}  [static]
```

Definition at line 16 of file Words.h.

4.6.2.17 START_OF_LINES

```
const std::uint32_t DU::START_OF_LINES {0xC4}  [static]
```

Definition at line 13 of file Words.h.

4.6.2.18 TASU1_SHIFT

```
const std::uint32_t DU::TASU1_SHIFT {24}  [static]
```

Definition at line 25 of file Words.h.

4.6.2.19 TASU2_SHIFT

```
const std::uint32_t DU::TASU2_SHIFT {28}  [static]
```

Definition at line 26 of file Words.h.

4.6.2.20 TDIF_SHIFT

```
const std::uint32_t DU::TDIF_SHIFT {32}  [static]
```

Definition at line 27 of file Words.h.

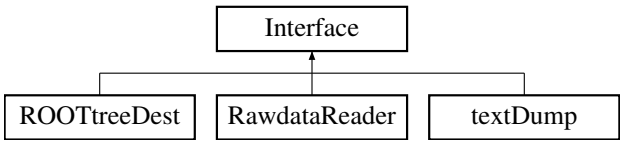
The documentation for this class was generated from the following file:

- /home/runner/work/streamout/streamout/libs/core/include/Words.h

4.7 Interface Class Reference

`#include <Interface.h>`

Inheritance diagram for Interface:



Public Member Functions

- Interface ()
- virtual ~Interface ()
- std::shared_ptr< spdlog::logger > & log ()
- void setLogger (const std::shared_ptr< spdlog::logger > &logger)

4.7.1 Detailed Description

Definition at line 11 of file Interface.h.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 Interface()

`Interface::Interface ( )  [inline]`

Definition at line 14 of file Interface.h.

```
00014 {}
```

4.7.2.2 ~Interface()

`virtual Interface::~Interface ( )  [inline], [virtual]`

Definition at line 15 of file Interface.h.

```
00015 {}
```

4.7.3 Member Function Documentation

4.7.3.1 log()

```
std::shared_ptr< spdlog::logger > & Interface::log ( )  [inline]
```

Definition at line 16 of file Interface.h.

```
00016 {return m_Logger;}
```

4.7.3.2 setLogger()

```
void Interface::setLogger (
            const std::shared_ptr< spdlog::logger > & logger )  [inline]
```

Definition at line 17 of file Interface.h.

```
00018   {
00019     m_Logger=logger;
00020   }
```

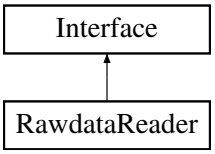
The documentation for this class was generated from the following file:

- /home/runner/work/streamout/streamout/libs/core/include/Interface.h

4.8 RawdataReader Class Reference

```
#include <RawdataReader.h>
```

Inheritance diagram for RawdataReader:



Public Member Functions

- RawdataReader (const char *fileName)
- void start ()
- void end ()
- float getFileSize ()
- void openFile (const std::string &fileName)
- void closeFile ()
- bool nextEvent ()
- bool nextDIFbuffer ()
- Buffer getSDHCALBuffer ()
- virtual ~RawdataReader ()

Static Public Member Functions

- static void setDefaultBufferSize (const std::size_t &size)

4.8.1 Detailed Description

Definition at line 17 of file RawdataReader.h.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 RawdataReader()

```
RawdataReader::RawdataReader (
            const char * fileName )  [explicit]
```

Definition at line 15 of file RawdataReader.cc.

```
00016 {
00017   m_buf.reserve(m_BufferSize);
00018   m_Filename = fileName;
00019 }
```

4.8.2.2 ~RawdataReader()

```
virtual RawdataReader::~RawdataReader ( )  [inline], [virtual]
```

Definition at line 29 of file RawdataReader.h.

```
00029 { closeFile(); }
```

4.8.3 Member Function Documentation

4.8.3.1 closeFile()

```
void RawdataReader::closeFile ( )
```

Definition at line 41 of file RawdataReader.cc.

```
00042 {
00043   try
00044   {
00045     if(m_FileStream.is_open()) m_FileStream.close();
00046   }
00047   catch(const std::ios_base::failure& e)
00048   {
00049     log()->error("Caught an ios_base::failure in closeFile : {} {}", e.what(), e.code().value());
00050     throw;
00051   }
00052 }
```

4.8.3.2 end()

```
void RawdataReader::end ( )
```

Definition at line 23 of file RawdataReader.cc.

```
00023 { closeFile(); }
```

4.8.3.3 getFileSize()

```
float RawdataReader::getFileSize ( )
```

Definition at line 123 of file RawdataReader.cc.

```
00123 { return m_FileSize; }
```

4.8.3.4 getSDHCALBuffer()

```
Buffer RawdataReader::getSDHCALBuffer ( )
```

Definition at line 115 of file RawdataReader.cc.

```
00116 {
00117   uncompress();
00118   return m_Buffer;
00119 }
```

4.8.3.5 nextDIFbuffer()

```
bool RawdataReader::nextDIFbuffer ( )
```

Definition at line 89 of file RawdataReader.cc.

```
00090 {
00091   try
00092   {
00093     static int DIF_processed{0};
00094     if(DIF_processed >= m_NumberOfDIF)
00095     {
00096       DIF_processed = 0;
00097       return false;
00098     }
00099     else
00100     {
00101       DIF_processed++;
00102       std::uint32_t bsize{0};
00103       m_FileStream.read(reinterpret_cast<char*>(&bsize), sizeof(std::uint32_t));
00104       m_FileStream.read(reinterpret_cast<char*>(&m_buf[0]), bsize);
00105       m_Buffer = Buffer(m_buf);
00106     }
00107   }
00108   catch(const std::ios_base::failure& e)
00109   {
00110     return false;
00111   }
00112   return true;
00113 }
```

4.8.3.6 nextEvent()

```
bool RawdataReader::nextEvent ( )
```

Definition at line 75 of file RawdataReader.cc.

```
00076 {
00077   try
00078   {
00079     m_FileStream.read(reinterpret_cast<char*>(&m_EventNumber), sizeof(std::uint32_t));
00080     m_FileStream.read(reinterpret_cast<char*>(&m_NumberOfDIF), sizeof(std::uint32_t));
00081   }
00082   catch(const std::ios_base::failure& e)
00083   {
00084     return false;
00085   }
00086   return true;
00087 }
```

4.8.3.7 openFile()

```
void RawdataReader::openFile (
            const std::string & fileName )
```

Definition at line 54 of file RawdataReader.cc.

```
00055 {
00056   try
00057   {
00058     m_FileStream.rdbuf()->pubsetbuf(0, 0);
00059     m_FileStream.exceptions(std::ifstream::failbit | std::ifstream::badbit);
00060     m_FileStream.open(fileName.c_str(), std::ios::in | std::ios::binary | std::ios::ate);  // Start at
      the end to directly calculate the size of the file then come back to beginning
00061     m_FileStream.rdbuf()->pubsetbuf(0, 0);
00062     if(m_FileStream.is_open())
00063     {
00064       setFileSize(m_FileStream.tellg());
00065       m_FileStream.seekg(0, std::ios::beg);
00066     }
00067   }
00068   catch(const std::ios_base::failure& e)
00069   {
00070     log()->error("Caught an ios_base::failure in openFile : {} {}", e.what(), e.code().value());
00071     throw;
00072   }
00073 }
```

4.8.3.8 setDefaultBufferSize()

```
void RawdataReader::setDefaultBufferSize (
            const std::size_t & size )  [static]
```

Definition at line 13 of file RawdataReader.cc.

```
00013 { m_BufferSize = size; }
```

4.8.3.9 start()

```
void RawdataReader::start ( )
```

Definition at line 21 of file RawdataReader.cc.

```
00021 { openFile(m_Filename); }
```

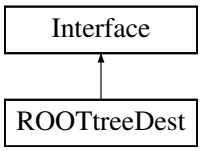
The documentation for this class was generated from the following files:

- /home/runner/work/streamout/streamout/libs/interface/RawDataReader/include/RawdataReader.h
- /home/runner/work/streamout/streamout/libs/interface/RawDataReader/src/RawdataReader.cc

4.9 ROOTtreeDest Class Reference

`#include <ROOTtreeDest.h>`

Inheritance diagram for ROOTtreeDest:



Classes

- struct DATA

Public Member Functions

- ROOTtreeDest ()
- void start ()
- void processDIF (DIFPtr *)
- void processFrame (DIFPtr *, std::uint32_t frameIndex)
- void processPadInFrame (DIFPtr *, std::uint32_t frameIndex, std::uint32_t channelIndex)
- void processSlowControl (const Buffer &)
- void end ()

4.9.1 Detailed Description

Definition at line 13 of file ROOTtreeDest.h.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 ROOTtreeDest()

```
ROOTtreeDest::ROOTtreeDest ( )
```

Definition at line 8 of file ROOTtreeDest.cc.

```
00009 {
00010   dataReset();
00011   _tree = new TTree("RawData", "Raw SDHCAL data tree");
00012   _tree->Branch("data", &_data,
       "DIFid/i:ASICid:CHANNELid:Thresh:DTC:GTC:DIF_BCID:frame_BCID:timeStamp:AbsoluteBCID/l");
00013 }
```

4.9.3 Member Function Documentation

4.9.3.1 end()

```
void ROOTtreeDest::end ( )  [inline]
```

Definition at line 31 of file ROOTtreeDest.h.

```
00031 { ; }
```

4.9.3.2 processDIF()

```
void ROOTtreeDest::processDIF (
            DIFPtr * d )
```

Definition at line 25 of file ROOTtreeDest.cc.

```
00026 {
00027   _data.DIFid        = d->getDIFid();
00028   _data.DTC          = d->getDTC();
00029   _data.GTC          = d->getGTC();
00030   _data.DIF_BCID     = d->getBCID();
00031   _data.AbsoluteBCID = d->getAbsoluteBCID();
00032 }
```

4.9.3.3 processFrame()

```
void ROOTtreeDest::processFrame (
            DIFPtr * d,
            std::uint32_t frameIndex )
```

Definition at line 34 of file ROOTtreeDest.cc.

```
00035 {
00036   _data.ASICid     = d->getASICid(frameIndex);
00037   _data.frame_BCID = d->getFrameBCID(frameIndex);
00038   _data.timeStamp  = d->getFrameTimeToTrigger(frameIndex);
00039 }
```

4.9.3.4 processPadInFrame()

```
void ROOTtreeDest::processPadInFrame (
            DIFPtr * d,
            std::uint32_t frameIndex,
            std::uint32_t channelIndex )
```

Definition at line 41 of file ROOTtreeDest.cc.

```
00042 {
00043   _data.CHANNELid = channelIndex;
00044   _data.Thresh    = d->getThresholdStatus(frameIndex, channelIndex);
00045   if(_data.Thresh != 0) _tree->Fill();
00046 }
```

4.9.3.5 processSlowControl()

```
void ROOTtreeDest::processSlowControl (
            const Buffer &  )  [inline]
```

Definition at line 30 of file ROOTtreeDest.h.
```
00030 { ; }
```

4.9.3.6 start()

```
void ROOTtreeDest::start ( )
```

Definition at line 23 of file ROOTtreeDest.cc.
```
00023 { dataReset(); }
```

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

- /home/runner/work/streamout/streamout/libs/interface/ROOT/include/ROOTtreeDest.h
- /home/runner/work/streamout/streamout/libs/interface/ROOT/src/ROOTtreeDest.cc

4.10 SDHCAL_buffer_loop< SOURCE, DESTINATION > Class Template Reference

```
#include <SDHCAL_buffer_loop.h>
```

Public Member Functions

- SDHCAL_buffer_loop (SOURCE &source, DESTINATION &dest, bool debug=false)
- void addSink (const spdlog::sink_ptr &sink, const spdlog::level::level_enum &level=spdlog::get_level())
- void loop (const std::int32_t &m_NbrEventsToProcess=0)
- void printAllCounters ()
- std::shared_ptr< spdlog::logger > log ()

4.10.1 Detailed Description

template<typename SOURCE, typename DESTINATION>
class SDHCAL_buffer_loop< SOURCE, DESTINATION >

Definition at line 34 of file SDHCAL_buffer_loop.h.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 SDHCAL_buffer_loop()

```
template<typename SOURCE , typename DESTINATION >
SDHCAL_buffer_loop< SOURCE, DESTINATION >::SDHCAL_buffer_loop (
            SOURCE & source,
            DESTINATION & dest,
            bool debug = false )  [inline]
```

Definition at line 37 of file SDHCAL_buffer_loop.h.

```
00037                                                                    : m_Source(source),
      m_Destination(dest), m_Debug(debug)
00038   {
00039     m_Logger = spdlog::create<spdlog::sinks::null_sink_mt>("streamout");
00040     if(!spdlog::get("streamout")) { spdlog::register_logger(m_Logger); }
00041     m_Source.setLogger(m_Logger);
00042     m_Destination.setLogger(m_Logger);
00043   }
```

4.10.3 Member Function Documentation

4.10.3.1 addSink()

```
template<typename SOURCE , typename DESTINATION >
void SDHCAL_buffer_loop< SOURCE, DESTINATION >::addSink (
            const spdlog::sink_ptr & sink,
            const spdlog::level::level_enum & level = spdlog::get_level() )  [inline]
```

Definition at line 45 of file SDHCAL_buffer_loop.h.

```
00046   {
00047     sink->set_level(level);
00048     m_Sinks.push_back(sink);
00049     m_Logger = std::make_shared<spdlog::logger>("streamout", begin(m_Sinks), end(m_Sinks));
00050     m_Source.setLogger(m_Logger);
00051     m_Destination.setLogger(m_Logger);
00052   }
```

4.10.3.2 log()

```
template<typename SOURCE , typename DESTINATION >
std::shared_ptr< spdlog::logger > SDHCAL_buffer_loop< SOURCE, DESTINATION >::log ( )  [inline]
```

Definition at line 120 of file SDHCAL_buffer_loop.h.

```
00120 { return m_Logger; }
```

4.10.3.3 loop()

```
template<typename SOURCE , typename DESTINATION >
void SDHCAL_buffer_loop< SOURCE, DESTINATION >::loop (
            const std::int32_t & m_NbrEventsToProcess = 0 )  [inline]
```

Definition at line 54 of file SDHCAL_buffer_loop.h.

```
00055   {
00056     m_Source.start();
00057     m_Destination.start();
00058     while(m_Source.nextEvent() && (m_NbrEventsToProcess == 0 || m_NbrEventsToProcess >= m_NbrEvents))
00059     {
00060       m_Logger->warn("===*** Event number {} ***===", m_NbrEvents);
00061       while(m_Source.nextDIFbuffer())
00062       {
00063         Buffer                     buffer           = m_Source.getSDHCALBuffer();
00064         unsigned char*             debug_variable_1 = buffer.end();
00065         SDHCAL_RawBuffer_Navigator bufferNavigator(buffer);
00066         unsigned char*             debug_variable_2 = bufferNavigator.getDIFBuffer().end();
00067         m_Logger->info("DIF BUFFER END {} {}", debug_variable_1, debug_variable_2);
00068         if(m_Debug) assert(debug_variable_1 == debug_variable_2);
00069         uint32_t idstart = bufferNavigator.getStartOfDIF();
00070         if(m_Debug && idstart == 0) m_Logger->info(to_hex(buffer));
00071         c.DIFStarter[idstart]++;
00072         if(!bufferNavigator.validBuffer()) continue;
00073         DIFPtr* d = bufferNavigator.getDIFPtr();
00074         if(m_Debug) assert(d != nullptr);
00075         if(d != nullptr)
00076         {
00077
       c.DIFPtrValueAtReturnedPos[bufferNavigator.getDIFBufferStart()[d->getGetFramePtrReturn()]]++;
00078           if(m_Debug) assert(bufferNavigator.getDIFBufferStart()[d->getGetFramePtrReturn()] == 0xa0);
00079         }
00080         c.SizeAfterDIFPtr[bufferNavigator.getSizeAfterDIFPtr()]++;
00081         m_Destination.processDIF(d);
00082         for(uint32_t i = 0; i < d->getNumberOfFrames(); i++)
00083         {
00084           m_Destination.processFrame(d, i);
00085           for(uint32_t j = 0; j < 64; j++) m_Destination.processPadInFrame(d, i, j);
00086         }
00087
00088         bool processSC = false;
00089         if(bufferNavigator.hasSlowControlData())
00090         {
00091           c.hasSlowControl++;
00092           processSC = true;
00093         }
00094         if(bufferNavigator.badSCData())
00095         {
00096           c.hasBadSlowControl++;
00097           processSC = false;
00098         }
00099         if(processSC) { m_Destination.processSlowControl(bufferNavigator.getSCBuffer()); }
00100
00101         Buffer eod = bufferNavigator.getEndOfAllData();
00102         c.SizeAfterAllData[eod.size()]++;
00103         unsigned char* debug_variable_3 = eod.end();
00104         m_Logger->info("END DATA BUFFER END {} {}", debug_variable_1, debug_variable_3);
00105         if(m_Debug) assert(debug_variable_1 == debug_variable_3);
00106         m_Logger->info("End of Data remaining stuff : {}", to_hex(eod));
00107
00108         int nonzeroCount = 0;
00109         for(unsigned char* it = eod.begin(); it != eod.end(); it++)
00110           if(static_cast<int>(*it) != 0) nonzeroCount++;
00111         c.NonZeroValusAtEndOfData[nonzeroCount]++;
00112       }  // end of DIF while loop
00113       m_Logger->warn("***=== Event number {} ===***", m_NbrEvents);
00114       m_NbrEvents++;
00115     }  // end of event while loop
00116     m_Destination.end();
00117     m_Source.end();
00118   }
```

4.10.3.4 printAllCounters()

```
template<typename SOURCE , typename DESTINATION >
void SDHCAL_buffer_loop< SOURCE, DESTINATION >::printAllCounters ( )  [inline]
```

Definition at line 119 of file SDHCAL_buffer_loop.h.

```
00119 { c.printAllCounters(m_Logger); }
```

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

- /home/runner/work/streamout/streamout/libs/core/include/SDHCAL_buffer_loop.h

4.11 SDHCAL_buffer_LoopCounter Struct Reference

`#include <SDHCAL_buffer_LoopCounter.h>`

Public Member Functions

- void printCounter (const std::string &description, const std::map< int, int > &m, const std::shared_ptr< spdlog::logger > &logger)
- void printAllCounters (const std::shared_ptr< spdlog::logger > &logger)

Public Attributes

- int hasSlowControl = 0
- int hasBadSlowControl = 0
- std::map< int, int > DIFStarter
- std::map< int, int > DIFPtrValueAtReturnedPos
- std::map< int, int > SizeAfterDIFPtr
- std::map< int, int > SizeAfterAllData
- std::map< int, int > NonZeroValusAtEndOfData

4.11.1 Detailed Description

Definition at line 12 of file SDHCAL_buffer_LoopCounter.h.

4.11.2 Member Function Documentation

4.11.2.1 printAllCounters()

```
void SDHCAL_buffer_LoopCounter::printAllCounters (
            const std::shared_ptr< spdlog::logger > & logger )
```

Definition at line 9 of file SDHCAL_buffer_LoopCounter.cc.

```
00010 {
00011   spdlog::level::level_enum level = logger->level();
00012   logger->set_level(spdlog::level::trace);
00013   logger->critical("BUFFER LOOP FINAL STATISTICS : ");
00014   printCounter("Start of DIF header", DIFStarter, logger);
00015   printCounter("Value after DIF data are processed", DIFPtrValueAtReturnedPos, logger);
00016   printCounter("Size remaining in buffer after end of DIF data", SizeAfterDIFPtr, logger);
00017   logger->critical("Number of Slow Control found {}  out of which {} are bad", hasSlowControl,
      hasBadSlowControl);
00018   printCounter("Size remaining after all of data have been processed", SizeAfterAllData, logger);
00019   printCounter("Number on non zero values in end of data buffer", NonZeroValusAtEndOfData, logger);
00020   logger->set_level(level);
00021 }
```

4.11.2.2 printCounter()

```
void SDHCAL_buffer_LoopCounter::printCounter (
            const std::string & description,
            const std::map< int, int > & m,
            const std::shared_ptr< spdlog::logger > & logger )
```

Definition at line 23 of file SDHCAL_buffer_LoopCounter.cc.

```
00024 {
00025   std::string out{"statistics for " + description + " : "};
00026   for(std::map<int, int>::const_iterator it = m.begin(); it != m.end(); it++)
00027   {
00028     if(it != m.begin()) out += ",";
00029     out += " [" + std::to_string(it->first) + "]=" + std::to_string(it->second);
00030   }
00031   logger->critical(out);
00032 }
```

4.11.3 Member Data Documentation

4.11.3.1 DIFPtrValueAtReturnedPos

```
std::map<int, int> SDHCAL_buffer_LoopCounter::DIFPtrValueAtReturnedPos
```

Definition at line 18 of file SDHCAL_buffer_LoopCounter.h.

4.11.3.2 DIFStarter

```
std::map<int, int> SDHCAL_buffer_LoopCounter::DIFStarter
```

Definition at line 17 of file SDHCAL_buffer_LoopCounter.h.

4.11.3.3 hasBadSlowControl

```
int SDHCAL_buffer_LoopCounter::hasBadSlowControl = 0
```

Definition at line 16 of file SDHCAL_buffer_LoopCounter.h.

4.11.3.4 hasSlowControl

```
int SDHCAL_buffer_LoopCounter::hasSlowControl = 0
```

Definition at line 15 of file SDHCAL_buffer_LoopCounter.h.

4.11.3.5 NonZeroValusAtEndOfData

```
std::map<int, int> SDHCAL_buffer_LoopCounter::NonZeroValusAtEndOfData
```

Definition at line 21 of file SDHCAL_buffer_LoopCounter.h.

4.11.3.6 SizeAfterAllData

```
std::map<int, int> SDHCAL_buffer_LoopCounter::SizeAfterAllData
```

Definition at line 20 of file SDHCAL_buffer_LoopCounter.h.

4.11.3.7 SizeAfterDIFPtr

```
std::map<int, int> SDHCAL_buffer_LoopCounter::SizeAfterDIFPtr
```

Definition at line 19 of file SDHCAL_buffer_LoopCounter.h.

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

- /home/runner/work/streamout/streamout/libs/core/include/SDHCAL_buffer_LoopCounter.h
- /home/runner/work/streamout/streamout/libs/core/src/SDHCAL_buffer_LoopCounter.cc

4.12 SDHCAL_RawBuffer_Navigator Class Reference

```
#include <SDHCAL_RawBuffer_Navigator.h>
```

Public Member Functions

- SDHCAL_RawBuffer_Navigator (const Buffer &b, const int &start=-1)
- ~SDHCAL_RawBuffer_Navigator ()
- bool validBuffer ()
- std::uint32_t getStartOfDIF ()
- unsigned char * getDIFBufferStart ()
- std::uint32_t getDIFBufferSize ()
- Buffer getDIFBuffer ()
- DIFPtr * getDIFPtr ()
- std::uint32_t getEndOfDIFData ()
- std::uint32_t getSizeAfterDIFPtr ()
- std::uint32_t getDIF_CRC ()
- bool hasSlowControlData ()
- Buffer getSCBuffer ()
- bool badSCData ()
- Buffer getEndOfAllData ()

Static Public Member Functions

- static void StartAt (const int &start)

4.12.1 Detailed Description

Definition at line 11 of file SDHCAL_RawBuffer_Navigator.h.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 SDHCAL_RawBuffer_Navigator()

```
SDHCAL_RawBuffer_Navigator::SDHCAL_RawBuffer_Navigator (
            const Buffer & b,
            const int & start = -1 )  [explicit]
```

Definition at line 15 of file SDHCAL_RawBuffer_Navigator.cc.

```
00015                                                                                : m_Buffer(b)
00016 {
00017   StartAt(start);
00018   m_DIFstartIndex = DIFUnpacker::getStartOfDIF(m_Buffer.begin(), m_Buffer.size(), m_Start);
00019 }
```

4.12.2.2 ~SDHCAL_RawBuffer_Navigator()

```
SDHCAL_RawBuffer_Navigator::~SDHCAL_RawBuffer_Navigator ( )
```

Definition at line 21 of file SDHCAL_RawBuffer_Navigator.cc.

```
00022 {
00023   if(m_TheDIFPtr != nullptr) delete m_TheDIFPtr;
00024 }
```

4.12.3 Member Function Documentation

4.12.3.1 badSCData()

```
bool SDHCAL_RawBuffer_Navigator::badSCData ( )
```

Definition at line 63 of file SDHCAL_RawBuffer_Navigator.cc.

```
00064 {
00065   setSCBuffer();
00066   return m_BadSCdata;
00067 }
```

4.12.3.2 getDIF_CRC()

```
uint32_t SDHCAL_RawBuffer_Navigator::getDIF_CRC ( )
```

Definition at line 46 of file SDHCAL_RawBuffer_Navigator.cc.

```
00047 {
00048   uint32_t i{getEndOfDIFData()};
00049   uint32_t ret{0};
00050   ret |= ((m_Buffer.begin()[i - 2]) « 8);
00051   ret |= m_Buffer.begin()[i - 1];
00052   return ret;
00053 }
```

4.12.3.3 getDIFBuffer()

```
Buffer SDHCAL_RawBuffer_Navigator::getDIFBuffer ( )
```

Definition at line 34 of file SDHCAL_RawBuffer_Navigator.cc.

```
00034 { return Buffer(getDIFBufferStart(), getDIFBufferSize()); }
```

4.12.3.4 getDIFBufferSize()

```
std::uint32_t SDHCAL_RawBuffer_Navigator::getDIFBufferSize ( )
```

Definition at line 32 of file SDHCAL_RawBuffer_Navigator.cc.

```
00032 { return m_Buffer.size() - m_DIFstartIndex; }
```

4.12.3.5 getDIFBufferStart()

```
unsigned char * SDHCAL_RawBuffer_Navigator::getDIFBufferStart ( )
```

Definition at line 30 of file SDHCAL_RawBuffer_Navigator.cc.

```
00030 { return &(m_Buffer.begin()[m_DIFstartIndex]); }
```

4.12.3.6 getDIFPtr()

```
DIFPtr * SDHCAL_RawBuffer_Navigator::getDIFPtr ( )
```

Definition at line 36 of file SDHCAL_RawBuffer_Navigator.cc.

```
00037 {
00038   if(m_TheDIFPtr == nullptr) m_TheDIFPtr = new DIFPtr(getDIFBufferStart(), getDIFBufferSize());
00039   return m_TheDIFPtr;
00040 }
```

4.12.3.7 getEndOfAllData()

```
Buffer SDHCAL_RawBuffer_Navigator::getEndOfAllData ( )
```

Definition at line 102 of file SDHCAL_RawBuffer_Navigator.cc.

```
00103 {
00104   setSCBuffer();
00105   if(hasSlowControlData() && !m_BadSCdata) { return Buffer(&(m_SCbuffer.begin()[m_SCbuffer.size()]),
      getSizeAfterDIFPtr() - 3 - m_SCbuffer.size()); }
00106   else
00107     return Buffer(&(getDIFBufferStart()[getEndOfDIFData()]), getSizeAfterDIFPtr() - 3);  // remove the
      2 bytes for CRC and the DIF trailer
00108 }
```

4.12.3.8 getEndOfDIFData()

```
std::uint32_t SDHCAL_RawBuffer_Navigator::getEndOfDIFData ( )
```

Definition at line 42 of file SDHCAL_RawBuffer_Navigator.cc.

```
00042 { return getDIFPtr()->getGetFramePtrReturn() + 3; }
```

4.12.3.9 getSCBuffer()

```
Buffer SDHCAL_RawBuffer_Navigator::getSCBuffer ( )
```

Definition at line 57 of file SDHCAL_RawBuffer_Navigator.cc.

```
00058 {
00059   setSCBuffer();
00060   return m_SCbuffer;
00061 }
```

4.12.3.10 getSizeAfterDIFPtr()

```
std::uint32_t SDHCAL_RawBuffer_Navigator::getSizeAfterDIFPtr ( )
```

Definition at line 44 of file SDHCAL_RawBuffer_Navigator.cc.

```
00044 { return getDIFBufferSize() - getDIFPtr()->getGetFramePtrReturn(); }
```

4.12.3.11 getStartOfDIF()

```
std::uint32_t SDHCAL_RawBuffer_Navigator::getStartOfDIF ( )
```

Definition at line 28 of file SDHCAL_RawBuffer_Navigator.cc.

```
00028 { return m_DIFstartIndex; }
```

4.12.3.12 hasSlowControlData()

```
bool SDHCAL_RawBuffer_Navigator::hasSlowControlData ( )
```

Definition at line 55 of file SDHCAL_RawBuffer_Navigator.cc.

```
00055 { return getDIFBufferStart()[getEndOfDIFData()] == 0xb1; }
```

4.12.3.13 StartAt()

```
void SDHCAL_RawBuffer_Navigator::StartAt (
            const int & start )  [static]
```

Definition at line 10 of file SDHCAL_RawBuffer_Navigator.cc.

```
00011 {
00012   if(start >= 0) m_Start = start;
00013 }
```

4.12.3.14 validBuffer()

```
bool SDHCAL_RawBuffer_Navigator::validBuffer ( )
```

Definition at line 26 of file SDHCAL_RawBuffer_Navigator.cc.

```
00026 { return m_DIFstartIndex != 0; }
```

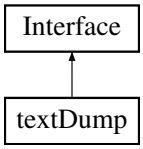
The documentation for this class was generated from the following files:

- /home/runner/work/streamout/streamout/libs/core/include/SDHCAL_RawBuffer_Navigator.h
- /home/runner/work/streamout/streamout/libs/core/src/SDHCAL_RawBuffer_Navigator.cc

4.13 textDump Class Reference

```
#include <textDump.h>
```

Inheritance diagram for textDump:



Public Member Functions

- textDump ()
- void start ()
- void processDIF (DIFPtr *)
- void processFrame (DIFPtr *, uint32_t frameIndex)
- void processPadInFrame (DIFPtr *, uint32_t frameIndex, uint32_t channelIndex)
- void processSlowControl (Buffer)
- void end ()
- std::shared_ptr< spdlog::logger > & print ()
- void setLevel (const spdlog::level::level_enum &level)

4.13.1 Detailed Description

Definition at line 15 of file textDump.h.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 textDump()

```
textDump::textDump ( )  [inline]
```

Definition at line 18 of file textDump.h.

```
00019   {
00020     m_InternalLogger = std::make_shared<spdlog::logger>("textDump",
       std::make_shared<spdlog::sinks::stdout_color_sink_mt>());
00021     m_InternalLogger->set_level(spdlog::level::trace);
00022   }
```

4.13.3 Member Function Documentation

4.13.3.1 end()

```
void textDump::end ( )
```

Definition at line 43 of file textDump.cc.

```
00043 { print()->info("textDump end of report"); }
```

4.13.3.2 print()

```
std::shared_ptr< spdlog::logger > & textDump::print ( )  [inline]
```

Definition at line 29 of file textDump.h.

```
00029 { return m_InternalLogger; }
```

4.13.3.3 processDIF()

```
void textDump::processDIF (
            DIFPtr * d )
```

Definition at line 9 of file textDump.cc.

```
00010 {
00011   if(nullptr == d)
00012   {
00013     print()->info("DIFPtr is nullptr");
00014     return;
00015   }
00016   print()->info("DIF number is {}", d->getDIFid());
00017   print()->info("DTC value is {}", d->getDTC());
00018   print()->info("GTC value is {}", d->getGTC());
00019   print()->info("DIF BCID is {}", d->getBCID());
00020   print()->info("Absolute BCID is {}", d->getAbsoluteBCID());
00021   print()->info("The number of frame is {}", d->getNumberOfFrames());
00022 }
```

4.13.3.4 processFrame()

```
void textDump::processFrame (
            DIFPtr * d,
            uint32_t frameIndex )
```

Definition at line 24 of file textDump.cc.

```
00025 {
00026   print()->info("Displaying frame number {}", frameIndex);
00027   print()->info("ASIC ID is {}", d->getASICid(frameIndex));
00028   print()->info("Frame BCID is {}", d->getFrameBCID(frameIndex));
00029   print()->info("Frame Time To Trigger (a.k.a timestamp) is {}",
       d->getFrameTimeToTrigger(frameIndex));
00030 }
```

4.13.3.5 processPadInFrame()

```
void textDump::processPadInFrame (
            DIFPtr * d,
            uint32_t frameIndex,
            uint32_t channelIndex )
```

Definition at line 32 of file textDump.cc.

```
00033 {
00034   if(d->getThresholdStatus(frameIndex, channelIndex) > 0)
00035   {
00036     print()->info("Displaying channel number {}", channelIndex);
00037     print()->info("Threshold status is {}", d->getThresholdStatus(frameIndex, channelIndex));
00038   }
00039 }
```

4.13.3.6 processSlowControl()

```
void textDump::processSlowControl (
            Buffer  )
```

Definition at line 41 of file textDump.cc.

```
00041 { print()->error("textDump::processSlowControl not implemented yet."); }
```

4.13.3.7 setLevel()

```
void textDump::setLevel (
            const spdlog::level::level_enum & level )  [inline]
```

Definition at line 30 of file textDump.h.

```
00030 { m_InternalLogger->set_level(level); }
```

4.13.3.8 start()

```
void textDump::start ( )
```

Definition at line 7 of file textDump.cc.

```
00007 { print()->info("Will dump bunch of DIF data"); }
```

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

- /home/runner/work/streamout/streamout/libs/interface/Dump/include/textDump.h
- /home/runner/work/streamout/streamout/libs/interface/Dump/src/textDump.cc

Chapter 5

File Documentation

5.1 /home/runner/work/streamout/streamout/libs/core/include/Bits.h File Reference

```
#include <cstdint>
#include <iosfwd>
```

Typedefs

- using bit8_t = std::uint8_t
- using bit16_t = std::uint16_t
- using bit32_t = std::uint32_t
- using bit64_t = std::uint64_t

Functions

- std::ostream & operator<< (std::ostream &os, const bit8_t &c)

  Stream operator to print bit8_t aka std::uint8_t and not char or unsigned char.

5.1.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Bits.h.

5.1.2 Typedef Documentation

5.1.2.1 bit16_t

```
using bit16_t = std::uint16_t
```

Definition at line 11 of file Bits.h.

5.1.2.2 bit32_t

```
using bit32_t = std::uint32_t
```

Definition at line 12 of file Bits.h.

5.1.2.3 bit64_t

```
using bit64_t = std::uint64_t
```

Definition at line 13 of file Bits.h.

5.1.2.4 bit8_t

```
using bit8_t = std::uint8_t
```

Definition at line 10 of file Bits.h.

5.1.3 Function Documentation

5.1.3.1 operator<<()

```
std::ostream & operator<< (
            std::ostream & os,
            const bit8_t & c )
```

Stream operator to print bit8_t aka std::uint8_t and not char or unsigned char.

Definition at line 8 of file Bits.cc.

```
00008 { return os << c + 0; }
```

5.2 Bits.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include <cstdint>
00008 #include <iosfwd>
00009
00010 using bit8_t  = std::uint8_t;  /*<! type to represent 8bits words (1 byte) */
00011 using bit16_t = std::uint16_t; /*<! type to represent 16bits words (2 bytes) */
00012 using bit32_t = std::uint32_t; /*<! type to represent 32bits words (4 bytes) */
00013 using bit64_t = std::uint64_t; /*<! type to represent 64bits words (8 bytes) */
00014
00016 std::ostream& operator«(std::ostream& os, const bit8_t& c);
```

5.3 /home/runner/work/streamout/streamout/libs/core/include/Buffer.h File Reference

```
#include "Bits.h"
#include <array>
#include <vector>
```

Classes

- class Buffer

5.4 Buffer.h

Go to the documentation of this file.

```
00001
00006 #pragma once
00007
00008 #include "Bits.h"
00009
00010 #include <array>
00011 #include <vector>
00012
00013 class Buffer
00014 {
00015 public:
00016   Buffer() : m_Buffer(nullptr), m_Size(0), m_Capacity(0) {}
00017   Buffer(const bit8_t b[], const std::size_t& i) : m_Buffer(const_cast<bit8_t*>(&b[0])), m_Size(i),
       m_Capacity(i) {}
00018   Buffer(const char b[], const std::size_t& i) : m_Buffer(const_cast<bit8_t*>(reinterpret_cast<const
       bit8_t*>(&b[0]))), m_Size(i), m_Capacity(i) {}
00019   template<typename T> Buffer(const std::vector<T>& rawdata) :
       m_Buffer(const_cast<bit8_t*>(reinterpret_cast<const bit8_t*>(rawdata.data()))), m_Size(rawdata.size()
       * sizeof(T)), m_Capacity(rawdata.capacity() * sizeof(T)) {}
00020   template<typename T, std::size_t N> Buffer(const std::array<T, N>& rawdata) :
       m_Buffer(const_cast<bit8_t*>(reinterpret_cast<const bit8_t*>(rawdata.data()))), m_Size(rawdata.size()
       * sizeof(T)), m_Capacity(rawdata.size() * sizeof(T)) {}
00021
00022   std::size_t size() const { return m_Size; }
00023   std::size_t capacity() const { return m_Capacity; }
00024
00025   void    set(unsigned char* b) { m_Buffer = b; }
00026   bit8_t* begin() { return m_Buffer; }
00027   bit8_t* end() { return m_Buffer + m_Size; }
00028   bit8_t& operator[](const std::size_t& pos) { return m_Buffer[pos]; }
00029   bit8_t& operator[](const std::size_t& pos) const { return m_Buffer[pos]; }
00030
00031   void setSize(const std::size_t& size) { m_Size = size; }
00032   virtual ~Buffer();
00033
00034 private:
00035   bit8_t*     m_Buffer{nullptr};
00036   std::size_t m_Size{0};
00037   std::size_t m_Capacity{0};
00038 };
```

5.5 /home/runner/work/streamout/streamout/libs/core/include/DIFPtr.h File Reference

```
#include "DIFUnpacker.h"
#include <vector>
```

Classes

- class DIFPtr

5.5.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file DIFPtr.h.

5.6 DIFPtr.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006 #include "DIFUnpacker.h"
00007
00008 #include <vector>
00009
00010 class DIFPtr
00011 {
00012 public:
00013   DIFPtr(unsigned char* p, const std::uint32_t& max_size);
00014   inline unsigned char*               getPtr() { return theDIF_; }
00015   inline std::uint32_t                getGetFramePtrReturn() { return theGetFramePtrReturn_; }
00016   inline std::vector<unsigned char*>& getFramesVector() { return theFrames_; }
00017   inline std::vector<unsigned char*>& getLinesVector() { return theLines_; }
00018   inline std::uint32_t                getID() { return DIFUnpacker::getID(theDIF_); }
00019   inline std::uint32_t                getDTC() { return DIFUnpacker::getDTC(theDIF_); }
00020   inline std::uint32_t                getGTC() { return DIFUnpacker::getGTC(theDIF_); }
00021   inline std::uint64_t                getAbsoluteBCID() { return
       DIFUnpacker::getAbsoluteBCID(theDIF_); }
00022   inline std::uint32_t                getBCID() { return DIFUnpacker::getBCID(theDIF_); }
00023   inline std::uint32_t                getLines() { return DIFUnpacker::getLines(theDIF_); }
00024   inline bool                         hasLine(uint32_t line) { return DIFUnpacker::hasLine(line,
       theDIF_); }
00025   inline std::uint32_t                getTASU1() { return DIFUnpacker::getTASU1(theDIF_); }
00026   inline std::uint32_t                getTASU2() { return DIFUnpacker::getTASU2(theDIF_); }
00027   inline std::uint32_t                getTDIF() { return DIFUnpacker::getTDIF(theDIF_); }
00028   inline float                        getTemperatureDIF() { return 0.508 * getTDIF() - 9.659; }
00029   inline float                        getTemperatureASU1() { return (getTASU1() >> 3) * 0.0625; }
00030   inline float                        getTemperatureASU2() { return (getTASU2() >> 3) * 0.0625; }
00031   inline bool                         hasTemperature() { return DIFUnpacker::hasTemperature(theDIF_);
       }
00032   inline bool                         hasAnalogReadout() { return
       DIFUnpacker::hasAnalogReadout(theDIF_); }
00033   inline std::uint32_t                getNumberOfFrames() { return theFrames_.size(); }
00034   inline unsigned char*               getFramePtr(uint32_t i) { return theFrames_[i]; }
00035   inline std::uint32_t                getFrameAsicHeader(uint32_t i) { return
       DIFUnpacker::getFrameAsicHeader(theFrames_[i]); }
00036   inline std::uint32_t                getFrameBCID(uint32_t i) { return
       DIFUnpacker::getFrameBCID(theFrames_[i]); }
00037   inline std::uint32_t                getFrameTimeToTrigger(uint32_t i) { return getBCID() -
       getFrameBCID(i); }
00038   inline bool                         getFrameLevel(uint32_t i, uint32_t ipad, uint32_t ilevel) {
       return DIFUnpacker::getFrameLevel(theFrames_[i], ipad, ilevel); }
```

```
00039   /*void                                dumpDIFInfo()
00040   {
00041     printf("DIF %d DTC %d GTC %d ABCID %lld BCID %d Lines %d Temperature %d \n", getID(), getDTC(),
      getGTC(), getAbsoluteBCID(), getBCID(), getLines(), hasTemperature());
00042
00043     if(hasTemperature()) printf("T: ASU1 %d %f ASU2 %d %f DIF %d  %f \n", getTASU1(),
      getTemperatureASU1(), getTASU2(), getTemperatureASU2(), getTDIF(), getTemperatureDIF());
00044     printf("Found %ld Lines and %ld Frames \n", theLines_.size(), theFrames_.size());
00045   }*/
00046   // Addition by GG
00047   inline uint32_t                    getDIFid() { return getID() & 0xFF; }
00048   inline uint32_t                    getASICid(uint32_t i) { return getFrameAsicHeader(i) & 0xFF; }
00049   inline uint32_t                    getThresholdStatus(uint32_t i, uint32_t ipad) { return
      (((uint32_t)getFrameLevel(i, ipad, 1)) << 1) | ((uint32_t)getFrameLevel(i, ipad, 0)); }
00050
00051 private:
00052   std::uint32_t               theSize_;
00053   std::uint32_t               theGetFramePtrReturn_;
00054   unsigned char*              theDIF_;
00055   std::vector<unsigned char*> theFrames_;
00056   std::vector<unsigned char*> theLines_;
00057 };
```

5.7 /home/runner/work/streamout/streamout/libs/core/include/DIFSlowControl.h File Reference

```
#include <bitset>
#include <cstdint>
#include <map>
#include <string>
```

Classes

- class DIFSlowControl

  Handler of DIF Slow Control info.

5.7.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file DIFSlowControl.h.

5.8 DIFSlowControl.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include <bitset>
00008 #include <cstdint>
00009 #include <map>
00010 #include <string>
00019 class DIFSlowControl
00020 {
00021 public:
00023
00028   DIFSlowControl(const std::uint8_t& version, const std::uint8_t& DIFid, unsigned char* buf);
```

```
00029
00031   inline std::uint8_t getDIFId();
00032
00034
00037   inline std::map<int, std::map<std::string, int» getChipsMap();
00038
00040
00044   inline std::map<std::string, int> getChipSlowControl(const int& asicid);
00045
00047
00051   inline int getChipSlowControl(const std::int8_t& asicid, const std::string& param);
00052
00054   void Dump();
00055
00056 private:
00058   DIFSlowControl() = delete;
00060   void FillHR1(const int& header_shift, unsigned char* cbuf);
00062   void FillHR2(const int& header_shift, unsigned char* cbuf);
00064   void FillAsicHR1(const std::bitset<72 * 8>& bs);
00066   void FillAsicHR2(const std::bitset<109 * 8>& bs);
00067
00068   unsigned int                              m_DIFId{0};
00069   unsigned int                              m_Version{0};
00070   unsigned int                              m_AsicType{0};  // asicType_
00071   unsigned int                              m_NbrAsic{0};
00072   std::map<int, std::map<std::string, int» m_MapSC;
00073 };
```

5.9 /home/runner/work/streamout/streamout/libs/core/include/↩ DIFUnpacker.h File Reference

```
#include <cstdint>
#include <vector>
```

Classes

- class DIFUnpacker

5.9.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file DIFUnpacker.h.

5.10 DIFUnpacker.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include <cstdint>
00008 #include <vector>
00009
00010 class DIFUnpacker
00011 {
00012 public:
00013   static std::uint64_t GrayToBin(const std::uint64_t& n);
00014   static std::uint32_t getStartOfDIF(const unsigned char* cbuf, const std::uint32_t& size_buf, const
       std::uint32_t& start = 92);
```

```
00015   static std::uint32_t getID(const unsigned char* cb, const std::uint32_t& idx = 0);
00016   static std::uint32_t getDTC(const unsigned char* cb, const std::uint32_t& idx = 0);
00017   static std::uint32_t getGTC(const unsigned char* cb, const std::uint32_t& idx = 0);
00018   static std::uint64_t getAbsoluteBCID(const unsigned char* cb, const std::uint32_t& idx = 0);
00019   static std::uint32_t getBCID(const unsigned char* cb, const std::uint32_t& idx = 0);
00020   static std::uint32_t getLines(const unsigned char* cb, const std::uint32_t& idx = 0);
00021   static bool          hasLine(const std::uint32_t& line, const unsigned char* cb, const
       std::uint32_t& idx = 0);
00022   static std::uint32_t getTASU1(const unsigned char* cb, const std::uint32_t& idx = 0);
00023   static std::uint32_t getTASU2(const unsigned char* cb, const std::uint32_t& idx = 0);
00024   static std::uint32_t getTDIF(const unsigned char* cb, const std::uint32_t& idx = 0);
00025   static bool          hasTemperature(const unsigned char* cb, const std::uint32_t& idx = 0);
00026   static bool          hasAnalogReadout(const unsigned char* cb, const std::uint32_t& idx = 0);
00027
00028   static std::uint32_t getFrameAsicHeader(const unsigned char* framePtr);
00029   static std::uint32_t getFrameBCID(const unsigned char* framePtr);
00030
00031   static bool getFramePAD(const unsigned char* framePtr, const std::uint32_t& ip);
00032   static bool getFrameLevel(const unsigned char* framePtr, const std::uint32_t& ip, const
       std::uint32_t& level);
00033
00034   static std::uint32_t getAnalogPtr(std::vector<unsigned char*>& vLines, unsigned char* cb, const
       std::uint32_t& idx = 0);
00035   static std::uint32_t getFramePtr(std::vector<unsigned char*>& vFrame, std::vector<unsigned char*>&
       vLines, const std::uint32_t& max_size, unsigned char* cb, const std::uint32_t& idx = 0);
00036   static void          dumpFrameOld(const unsigned char* buf);
00037   static std::uint32_t swap_bytes(const unsigned char* buf);  // Stolen from DCBufferReader
00038 };
```

5.11 /home/runner/work/streamout/streamout/libs/core/include/Formatters.h File Reference

```
#include "Bits.h"
#include <iosfwd>
#include <string>
```

Functions

- std::string to_dec (const Buffer &b, const std::size_t &begin=0, const std::size_t &end=-1)
- std::string to_dec (const bit8_t &)
- std::string to_dec (const bit16_t &)
- std::string to_dec (const bit32_t &)
- std::string to_dec (const bit64_t &)
- std::string to_hex (const Buffer &b, const std::size_t &begin=0, const std::size_t &end=-1)
- std::string to_hex (const bit8_t &)
- std::string to_hex (const bit16_t &)
- std::string to_hex (const bit32_t &)
- std::string to_hex (const bit64_t &)
- std::string to_bin (const Buffer &b, const std::size_t &begin=0, const std::size_t &end=-1)
- std::string to_bin (const bit8_t &)
- std::string to_bin (const bit16_t &)
- std::string to_bin (const bit32_t &)
- std::string to_bin (const bit64_t &)
- std::string to_oct (const Buffer &b, const std::size_t &begin=0, const std::size_t &end=-1)
- std::string to_oct (const bit8_t &)
- std::string to_oct (const bit16_t &)
- std::string to_oct (const bit32_t &)
- std::string to_oct (const bit64_t &)

5.11.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Formatters.h.

5.11.2 Function Documentation

5.11.2.1 to_bin() [1/5]

```
std::string to_bin (
            const bit16_t & b )
```

Definition at line 92 of file Formatters.cc.

```
00093 {
00094   return fmt::format("{:#016b}",b);
00095 }
```

5.11.2.2 to_bin() [2/5]

```
std::string to_bin (
            const bit32_t & b )
```

Definition at line 97 of file Formatters.cc.

```
00098 {
00099   return fmt::format("{:#032b}",b);
00100 }
```

5.11.2.3 to_bin() [3/5]

```
std::string to_bin (
            const bit64_t & b )
```

Definition at line 102 of file Formatters.cc.

```
00103 {
00104   return fmt::format("{:#064b}",b);
00105 }
```

5.11.2.4 to_bin() [4/5]

```
std::string to_bin (
              const bit8_t & b )
```

Definition at line 87 of file Formatters.cc.

```
00088 {
00089   return fmt::format("{:#08b}",b);
00090 }
```

5.11.2.5 to_bin() [5/5]

```
std::string to_bin (
              const Buffer & b,
              const std::size_t & begin = 0,
              const std::size_t & end = -1 )
```

Definition at line 74 of file Formatters.cc.

```
00075 {
00076   std::size_t iend =end;
00077   if(iend==-1) iend=b.size();
00078   std::string ret;
00079   for(std::size_t k = begin; k < iend; k++)
00080   {
00081     ret+= to_bin(b[k]);
00082     ret+=" - ";
00083   }
00084   return ret;
00085 }
```

5.11.2.6 to_dec() [1/5]

```
std::string to_dec (
              const bit16_t & b )
```

Definition at line 26 of file Formatters.cc.

```
00027 {
00028   return fmt::format("{:#016d}",b);
00029 }
```

5.11.2.7 to_dec() [2/5]

```
std::string to_dec (
              const bit32_t & b )
```

Definition at line 31 of file Formatters.cc.

```
00032 {
00033   return fmt::format("{:#032d}",b);
00034 }
```

5.11.2.8 to_dec() [3/5]

```
std::string to_dec (
            const bit64_t & b )
```

Definition at line 36 of file Formatters.cc.

```
00037 {
00038   return fmt::format("{:#064d}",b);
00039 }
```

5.11.2.9 to_dec() [4/5]

```
std::string to_dec (
            const bit8_t & b )
```

Definition at line 21 of file Formatters.cc.

```
00022 {
00023     return fmt::format("{:#08d}",b);
00024 }
```

5.11.2.10 to_dec() [5/5]

```
std::string to_dec (
            const Buffer & b,
            const std::size_t & begin = 0,
            const std::size_t & end = -1 )
```

Definition at line 8 of file Formatters.cc.

```
00009 {
00010   std::size_t iend =end;
00011   if(iend==-1) iend=b.size();
00012   std::string ret;
00013   for(std::size_t k = begin; k < iend; k++)
00014   {
00015     ret+= to_dec(b[k]);
00016     ret+=" - ";
00017   }
00018   return ret;
00019 }
```

5.11.2.11 to_hex() [1/5]

```
std::string to_hex (
            const bit16_t & b )
```

Definition at line 59 of file Formatters.cc.

```
00060 {
00061   return fmt::format("{:#016x}",b);
00062 }
```

5.11.2.12 to_hex() [2/5]

```
std::string to_hex (
            const bit32_t & b )
```

Definition at line 64 of file Formatters.cc.

```
00065 {
00066   return fmt::format("{:#032x}",b);
00067 }
```

5.11.2.13 to_hex() [3/5]

```
std::string to_hex (
            const bit64_t & b )
```

Definition at line 69 of file Formatters.cc.

```
00070 {
00071   return fmt::format("{:#064x}",b);
00072 }
```

5.11.2.14 to_hex() [4/5]

```
std::string to_hex (
            const bit8_t & b )
```

Definition at line 54 of file Formatters.cc.

```
00055 {
00056   return fmt::format("{:#08x}",b);
00057 }
```

5.11.2.15 to_hex() [5/5]

```
std::string to_hex (
            const Buffer & b,
            const std::size_t & begin = 0,
            const std::size_t & end = -1 )
```

Definition at line 41 of file Formatters.cc.

```
00042 {
00043   std::size_t iend =end;
00044   if(iend==-1) iend=b.size();
00045   std::string ret;
00046   for(std::size_t k = begin; k < iend; k++)
00047   {
00048     ret+= to_hex(b[k]);
00049     ret+=" - ";
00050   }
00051   return ret;
00052 }
```

5.11.2.16 to_oct() [1/5]

```
std::string to_oct (
            const bit16_t & b )
```

Definition at line 125 of file Formatters.cc.

```
00126 {
00127   return fmt::format("{:#016o}",b);
00128 }
```

5.11.2.17 to_oct() [2/5]

```
std::string to_oct (
            const bit32_t & b )
```

Definition at line 130 of file Formatters.cc.

```
00131 {
00132   return fmt::format("{:#032o}",b);
00133 }
```

5.11.2.18 to_oct() [3/5]

```
std::string to_oct (
            const bit64_t & b )
```

Definition at line 135 of file Formatters.cc.

```
00136 {
00137   return fmt::format("{:#064o}",b);
00138 }
```

5.11.2.19 to_oct() [4/5]

```
std::string to_oct (
            const bit8_t & b )
```

Definition at line 120 of file Formatters.cc.

```
00121 {
00122   return fmt::format("{:#08o}",b);
00123 }
```

5.11.2.20 to_oct() [5/5]

```
std::string to_oct (
              const Buffer & b,
              const std::size_t & begin = 0,
              const std::size_t & end = -1 )
```

Definition at line 107 of file Formatters.cc.

```
00108 {
00109   std::size_t iend =end;
00110   if(iend==-1) iend=b.size();
00111   std::string ret;
00112   for(std::size_t k = begin; k < iend; k++)
00113   {
00114     ret+= to_oct(b[k]);
00115     ret+=" - ";
00116   }
00117   return ret;
00118 }
```

5.12 Formatters.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include "Bits.h"
00008
00009 #include <iosfwd>
00010 #include <string>
00011
00012 #include "Bits.h"
00013
00014 class Buffer;
00015
00016 std::string to_dec(const Buffer& b,const std::size_t& begin=0,const std::size_t& end=-1);
00017 std::string to_dec(const bit8_t&);
00018 std::string to_dec(const bit16_t&);
00019 std::string to_dec(const bit32_t&);
00020 std::string to_dec(const bit64_t&);
00021
00022 std::string to_hex(const Buffer& b,const std::size_t& begin=0,const std::size_t& end=-1);
00023 std::string to_hex(const bit8_t&);
00024 std::string to_hex(const bit16_t&);
00025 std::string to_hex(const bit32_t&);
00026 std::string to_hex(const bit64_t&);
00027
00028 std::string to_bin(const Buffer& b,const std::size_t& begin=0,const std::size_t& end=-1);
00029 std::string to_bin(const bit8_t&);
00030 std::string to_bin(const bit16_t&);
00031 std::string to_bin(const bit32_t&);
00032 std::string to_bin(const bit64_t&);
00033
00034 std::string to_oct(const Buffer& b,const std::size_t& begin=0,const std::size_t& end=-1);
00035 std::string to_oct(const bit8_t&);
00036 std::string to_oct(const bit16_t&);
00037 std::string to_oct(const bit32_t&);
00038 std::string to_oct(const bit64_t&);
```

5.13 /home/runner/work/streamout/streamout/libs/core/include/↩ Interface.h File Reference

```
#include "Buffer.h"
#include <spdlog/logger.h>
#include <memory>
```

Classes

- class Interface

5.13.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Interface.h.

5.14 Interface.h

Go to the documentation of this file.

```
00001
00004 #pragma once
00005
00006 #include "Buffer.h"
00007
00008 #include <spdlog/logger.h>
00009 #include <memory>
00010
00011 class Interface
00012 {
00013 public:
00014   Interface(){}
00015   virtual ~Interface(){}
00016   std::shared_ptr<spdlog::logger>& log() {return m_Logger;}
00017   void setLogger(const std::shared_ptr<spdlog::logger>& logger)
00018   {
00019     m_Logger=logger;
00020   }
00021 private:
00022   std::shared_ptr<spdlog::logger> m_Logger{nullptr};
00023 };
```

5.15 /home/runner/work/streamout/streamout/libs/core/include/↩ SDHCAL_buffer_loop.h File Reference

```
#include "Buffer.h"
#include "Formatters.h"
#include "SDHCAL_RawBuffer_Navigator.h"
#include "SDHCAL_buffer_LoopCounter.h"
#include <cassert>
#include <memory>
#include <spdlog/sinks/null_sink.h>
#include <spdlog/spdlog.h>
#include <vector>
```

Classes

- class SDHCAL_buffer_loop< SOURCE, DESTINATION >

5.15.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file SDHCAL_buffer_loop.h.

5.16 SDHCAL_buffer_loop.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include "Buffer.h"
00008 #include "Formatters.h"
00009 #include "SDHCAL_RawBuffer_Navigator.h"
00010 #include "SDHCAL_buffer_LoopCounter.h"
00011
00012 #include <cassert>
00013 #include <memory>
00014 #include <spdlog/sinks/null_sink.h>
00015 #include <spdlog/spdlog.h>
00016 #include <vector>
00017
00018 // function to loop on buffers
00019 //
00020 // template class should implement
00021 // void SOURCE::start();
00022 // bool SOURCE::next();
00023 // void SOURCE::end();
00024 // SDHCAL_buffer SOURCE::getSDHCALBuffer();
00025 //
00026 // void DESTINATION::start();
00027 // void DESTINATION::processDIF(DIFPtr*);
00028 // void DESTINATION::processFrame(DIFPtr*,uint32_t frameIndex);
00029 // void DESTINATION::processPadInFrame(DIFPtr*,uint32_t frameIndex, uint32_t channelIndex);
00030 // void DESTINATION::processSlowControl(SDHCAL_buffer);
00031 // void DESTINATION::end();
00032 //
00033
00034 template<typename SOURCE, typename DESTINATION> class SDHCAL_buffer_loop
00035 {
00036 public:
00037   SDHCAL_buffer_loop(SOURCE& source, DESTINATION& dest, bool debug = false) : m_Source(source),
      m_Destination(dest), m_Debug(debug)
00038   {
00039     m_Logger = spdlog::create<spdlog::sinks::null_sink_mt>("streamout");
00040     if(!spdlog::get("streamout")) { spdlog::register_logger(m_Logger); }
00041     m_Source.setLogger(m_Logger);
00042     m_Destination.setLogger(m_Logger);
00043   }
00044
00045   void addSink(const spdlog::sink_ptr& sink, const spdlog::level::level_enum& level =
      spdlog::get_level())
00046   {
00047     sink->set_level(level);
00048     m_Sinks.push_back(sink);
00049     m_Logger = std::make_shared<spdlog::logger>("streamout", begin(m_Sinks), end(m_Sinks));
00050     m_Source.setLogger(m_Logger);
00051     m_Destination.setLogger(m_Logger);
00052   }
00053
00054   void loop(const std::int32_t& m_NbrEventsToProcess = 0)
00055   {
00056     m_Source.start();
00057     m_Destination.start();
00058     while(m_Source.nextEvent() && (m_NbrEventsToProcess == 0 || m_NbrEventsToProcess >= m_NbrEvents))
00059     {
00060       m_Logger->warn("===*** Event number {} ***===", m_NbrEvents);
00061       while(m_Source.nextDIFbuffer())
00062       {
00063         Buffer                    buffer           = m_Source.getSDHCALBuffer();
00064         unsigned char*            debug_variable_1 = buffer.end();
00065         SDHCAL_RawBuffer_Navigator bufferNavigator(buffer);
00066         unsigned char*            debug_variable_2 = bufferNavigator.getDIFBuffer().end();
00067         m_Logger->info("DIF BUFFER END {} {}", debug_variable_1, debug_variable_2);
```

```
00068         if(m_Debug) assert(debug_variable_1 == debug_variable_2);
00069         uint32_t idstart = bufferNavigator.getStartOfDIF();
00070         if(m_Debug && idstart == 0) m_Logger->info(to_hex(buffer));
00071         c.DIFStarter[idstart]++;
00072         if(!bufferNavigator.validBuffer()) continue;
00073         DIFPtr* d = bufferNavigator.getDIFPtr();
00074         if(m_Debug) assert(d != nullptr);
00075         if(d != nullptr)
00076         {
00077
      c.DIFPtrValueAtReturnedPos[bufferNavigator.getDIFBufferStart()[d->getGetFramePtrReturn()]]++;
00078           if(m_Debug) assert(bufferNavigator.getDIFBufferStart()[d->getGetFramePtrReturn()] == 0xa0);
00079         }
00080         c.SizeAfterDIFPtr[bufferNavigator.getSizeAfterDIFPtr()]++;
00081         m_Destination.processDIF(d);
00082         for(uint32_t i = 0; i < d->getNumberOfFrames(); i++)
00083         {
00084           m_Destination.processFrame(d, i);
00085           for(uint32_t j = 0; j < 64; j++) m_Destination.processPadInFrame(d, i, j);
00086         }
00087
00088         bool processSC = false;
00089         if(bufferNavigator.hasSlowControlData())
00090         {
00091           c.hasSlowControl++;
00092           processSC = true;
00093         }
00094         if(bufferNavigator.badSCData())
00095         {
00096           c.hasBadSlowControl++;
00097           processSC = false;
00098         }
00099         if(processSC) { m_Destination.processSlowControl(bufferNavigator.getSCBuffer()); }
00100
00101         Buffer eod = bufferNavigator.getEndOfAllData();
00102         c.SizeAfterAllData[eod.size()]++;
00103         unsigned char* debug_variable_3 = eod.end();
00104         m_Logger->info("END DATA BUFFER END {} {}", debug_variable_1, debug_variable_3);
00105         if(m_Debug) assert(debug_variable_1 == debug_variable_3);
00106         m_Logger->info("End of Data remaining stuff : {}", to_hex(eod));
00107
00108         int nonzeroCount = 0;
00109         for(unsigned char* it = eod.begin(); it != eod.end(); it++)
00110           if(static_cast<int>(*it) != 0) nonzeroCount++;
00111         c.NonZeroValusAtEndOfData[nonzeroCount]++;
00112       }  // end of DIF while loop
00113       m_Logger->warn("***=== Event number {} ===***", m_NbrEvents);
00114       m_NbrEvents++;
00115     }  // end of event while loop
00116     m_Destination.end();
00117     m_Source.end();
00118   }
00119   void                            printAllCounters() { c.printAllCounters(m_Logger); }
00120   std::shared_ptr<spdlog::logger> log() { return m_Logger; }
00121
00122 private:
00123   std::shared_ptr<spdlog::logger> m_Logger{nullptr};
00124   std::vector<spdlog::sink_ptr>   m_Sinks;
00125   SDHCAL_buffer_LoopCounter       c;
00126   SOURCE&                         m_Source{nullptr};
00127   DESTINATION&                    m_Destination{nullptr};
00128   bool                            m_Debug{false};
00129   std::uint32_t                   m_NbrEvents{1};
00130 };
```

5.17 /home/runner/work/streamout/streamout/libs/core/include/↩ SDHCAL_buffer_LoopCounter.h File Reference

```
#include <map>
#include <memory>
#include <spdlog/fwd.h>
#include <string>
```

Classes

- struct SDHCAL_buffer_LoopCounter

5.17.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file SDHCAL_buffer_LoopCounter.h.

5.18 SDHCAL_buffer_LoopCounter.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include <map>
00008 #include <memory>
00009 #include <spdlog/fwd.h>
00010 #include <string>
00011
00012 struct SDHCAL_buffer_LoopCounter
00013 {
00014 public:
00015   int                hasSlowControl    = 0;
00016   int                hasBadSlowControl = 0;
00017   std::map<int, int> DIFStarter;
00018   std::map<int, int> DIFPtrValueAtReturnedPos;
00019   std::map<int, int> SizeAfterDIFPtr;
00020   std::map<int, int> SizeAfterAllData;
00021   std::map<int, int> NonZeroValusAtEndOfData;
00022
00023   void printCounter(const std::string& description, const std::map<int, int>& m, const
       std::shared_ptr<spdlog::logger>& logger);
00024   void printAllCounters(const std::shared_ptr<spdlog::logger>& logger);
00025 };
```

5.19 /home/runner/work/streamout/streamout/libs/core/include/SDHCAL_RawBuffer_Navigator.h File Reference

```
#include "Buffer.h"
#include "DIFPtr.h"
```

Classes

- class SDHCAL_RawBuffer_Navigator

5.19.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file SDHCAL_RawBuffer_Navigator.h.

5.20 SDHCAL_RawBuffer_Navigator.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include "Buffer.h"
00008 #include "DIFPtr.h"
00009
00010 // class to navigate in the raw data buffer
00011 class SDHCAL_RawBuffer_Navigator
00012 {
00013 public:
00014   explicit SDHCAL_RawBuffer_Navigator(const Buffer& b, const int& start = -1);
00015   ~SDHCAL_RawBuffer_Navigator();
00016   bool           validBuffer();
00017   std::uint32_t  getStartOfDIF();
00018   unsigned char* getDIFBufferStart();
00019   std::uint32_t  getDIFBufferSize();
00020   Buffer         getDIFBuffer();
00021   DIFPtr*        getDIFPtr();
00022   std::uint32_t  getEndOfDIFData();
00023   std::uint32_t  getSizeAfterDIFPtr();
00024   std::uint32_t  getDIF_CRC();
00025   bool           hasSlowControlData();
00026   Buffer         getSCBuffer();
00027   bool           badSCData();
00028   Buffer         getEndOfAllData();
00029   static void    StartAt(const int& start);
00030
00031 private:
00032   void          setSCBuffer();
00033   Buffer        m_Buffer;
00034   Buffer        m_SCbuffer;
00035   std::uint32_t m_DIFstartIndex{0};
00036   DIFPtr*       m_TheDIFPtr{nullptr};
00037   bool          m_BadSCdata{false};
00038   static int    m_Start;
00039 };
```

5.21 /home/runner/work/streamout/streamout/libs/core/include/Words.h File Reference

Classes

- class DU

5.21.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Words.h.

5.22 Words.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 class DU
00008 {
```

```
00009 public:
00010   static const std::uint32_t START_OF_DIF{0xB0};
00011   static const std::uint32_t START_OF_DIF_TEMP{0xBB};
00012   static const std::uint32_t END_OF_DIF{0xA0};
00013   static const std::uint32_t START_OF_LINES{0xC4};
00014   static const std::uint32_t END_OF_LINES{0xD4};
00015
00016   static const std::uint32_t START_OF_FRAME{0xB4};
00017   static const std::uint32_t END_OF_FRAME{0xA3};
00018
00019   static const std::uint32_t ID_SHIFT{1};
00020   static const std::uint32_t DTC_SHIFT{2};
00021   static const std::uint32_t GTC_SHIFT{10};
00022   static const std::uint32_t ABCID_SHIFT{14};
00023   static const std::uint32_t BCID_SHIFT{20};
00024   static const std::uint32_t LINES_SHIFT{23};
00025   static const std::uint32_t TASU1_SHIFT{24};
00026   static const std::uint32_t TASU2_SHIFT{28};
00027   static const std::uint32_t TDIF_SHIFT{32};
00028
00029   static const std::uint32_t FRAME_ASIC_HEADER_SHIFT{0};
00030   static const std::uint32_t FRAME_BCID_SHIFT{1};
00031   static const std::uint32_t FRAME_DATA_SHIFT{4};
00032   static const std::uint32_t FRAME_SIZE{20};
00033 };
```

5.23 /home/runner/work/streamout/streamout/libs/core/src/Bits.cc File Reference

```
#include "Bits.h"
```

Functions

- std::ostream & operator<< (std::ostream &os, const bit8_t &c)

  Stream operator to print bit8_t aka std::uint8_t and not char or unsigned char.

5.23.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Bits.cc.

5.23.2 Function Documentation

5.23.2.1 operator<<()

```
std::ostream & operator<< (
            std::ostream & os,
            const bit8_t & c )
```

Stream operator to print bit8_t aka std::uint8_t and not char or unsigned char.

Definition at line 8 of file Bits.cc.

```
00008 { return os « c + 0; }
```

5.24 Bits.cc

Go to the documentation of this file.

```
00001
00006 #include "Bits.h"
00007
00008 std::ostream& operator«(std::ostream& os, const bit8_t& c) { return os « c + 0; }
```

5.25 /home/runner/work/streamout/streamout/libs/core/src/Buffer.cc File Reference

```
#include "Buffer.h"
```

5.26 Buffer.cc

Go to the documentation of this file.

```
00001
00006 #include "Buffer.h"
00007
00008 Buffer::~Buffer() {}
```

5.27 /home/runner/work/streamout/streamout/libs/core/src/DIFPtr.cc File Reference

```
#include "DIFPtr.h"
#include "spdlog/spdlog.h"
#include <string>
```

5.28 DIFPtr.cc

Go to the documentation of this file.

```
00001
00005 #include "DIFPtr.h"
00006
00007 #include "spdlog/spdlog.h"
00008
00009 #include <string>
00010
00011 DIFPtr::DIFPtr(unsigned char* p, const std::uint32_t& max_size) : theDIF_(p), theSize_(max_size)
00012 {
00013   theFrames_.clear();
00014   theLines_.clear();
00015   try
00016   {
00017     theGetFramePtrReturn_ = DIFUnpacker::getFramePtr(theFrames_, theLines_, theSize_, theDIF_);
00018   }
00019   catch(const std::string& e)
00020   {
00021     spdlog::get("streamout")->error(" DIF {} T ? {} {}", getID(), hasTemperature(), e);
00022   }
00023 }
```

5.29 /home/runner/work/streamout/streamout/libs/core/src/DIFSlowControl.cc File Reference

```
#include "DIFSlowControl.h"
#include <cstdint>
#include <iostream>
```

5.29.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file DIFSlowControl.cc.

5.30 DIFSlowControl.cc

Go to the documentation of this file.

```
00001
00005 #include "DIFSlowControl.h"
00006
00007 #include <cstdint>
00008 #include <iostream>
00009
00010 DIFSlowControl::DIFSlowControl(const std::uint8_t& version, const std::uint8_t& DIfId, unsigned char*
       cbuf) : m_Version(version), m_DIFId(DIfId), m_AsicType(2)
00011 {
00012   if(cbuf[0] != 0xb1) return;
00013   int header_shift{6};
00014   if(m_Version < 8) m_NbrAsic = cbuf[5];
00015   else
00016   {
00017     m_DIFId      = cbuf[1];
00018     m_NbrAsic    = cbuf[2];
00019     header_shift = 3;
00020   }
00021   int size_hardroc1 = m_NbrAsic * 72 + header_shift + 1;
00022   if(cbuf[size_hardroc1 - 1] != 0xa1) size_hardroc1 = 0;
00023
00024   int size_hardroc2 = m_NbrAsic * 109 + header_shift + 1;
00025   if(cbuf[size_hardroc2 - 1] != 0xa1) size_hardroc2 = 0;
00026   if(size_hardroc1 != 0)
00027   {
00028     FillHR1(header_shift, cbuf);
00029     m_AsicType = 1;
00030   }
00031   else if(size_hardroc2 != 0)
00032     FillHR2(header_shift, cbuf);
00033   else
00034     return;
00035 }
00036
00037 inline std::uint8_t DIFSlowControl::getDIFId() { return m_DIFId; }
00038
00039 inline std::map<int, std::map<std::string, int» DIFSlowControl::getChipsMap() { return m_MapSC; }
00040
00041 inline std::map<std::string, int> DIFSlowControl::getChipSlowControl(const int& asicid) { return
       m_MapSC[asicid]; }
00042
00043 inline int DIFSlowControl::getChipSlowControl(const std::int8_t& asicid, const std::string& param) {
       return getChipSlowControl(asicid)[param]; }
00044
00045 void DIFSlowControl::Dump()
00046 {
00047   for(std::map<int, std::map<std::string, int»::iterator it = m_MapSC.begin(); it != m_MapSC.end();
       it++)
00048   {
00049     std::cout « "ASIC " « it->first « std::endl;
```

```
00050     for(std::map<std::string, int>::iterator jt = (it->second).begin(); jt != (it->second).end();
       jt++) std::cout « jt->first « " : " « jt->second « std::endl;
00051   }
00052 }
00053
00054 void DIFSlowControl::FillHR1(const int& header_shift, unsigned char* cbuf)
00055 {
00056   int nasic{cbuf[header_shift - 1]};
00057   int idx{header_shift};
00058   for(int k = 0; k < nasic; k++)
00059   {
00060     std::bitset<72 * 8> bs;
00061     // printf("%x %x \n",cbuf[idx+k*72+69],cbuf[idx+k*72+70]);
00062     for(int l = 71; l >= 0; l--)
00063     {
00064       //  printf("%d %x : %d -->",l,cbuf[idx+k*72+l],(71-l)*8);
00065       for(int m = 0; m < 8; m++)
00066       {
00067         if(((1 « m) & cbuf[idx + k * 72 + l]) != 0) bs.set((71 - l) * 8 + m, 1);
00068         else
00069           bs.set((71 - l) * 8 + m, 0);
00070         // printf("%d",(int) bs[(71-l)*8+m]);
00071       }
00072       // printf("\n");
00073     }
00074     FillAsicHR1(bs);
00075   }
00076 }
00077
00078 void DIFSlowControl::FillHR2(const int& header_shift, unsigned char* cbuf)
00079 {
00080   // int scsize1=cbuf[header_shift-1]*109+(header_shift-1)+2;
00081   int nasic{cbuf[header_shift - 1]};
00082   int idx{header_shift};
00083   // std::cout«" DIFSlowControl::FillHR nasic "«nasic«std::endl;
00084   for(int k = 0; k < nasic; k++)
00085   {
00086     std::bitset<109 * 8> bs;
00087     // printf("%x %x \n",cbuf[idx+k*109+69],cbuf[idx+k*109+70]);
00088     for(int l = 108; l >= 0; l--)
00089     {
00090       //  printf("%d %x : %d -->",l,cbuf[idx+k*109+l],(71-l)*8);
00091       for(int m = 0; m < 8; m++)
00092       {
00093         if(((1 « m) & cbuf[idx + k * 109 + l]) != 0) bs.set((108 - l) * 8 + m, 1);
00094         else
00095           bs.set((108 - l) * 8 + m, 0);
00096         // printf("%d",(int) bs[(71-l)*8+m]);
00097       }
00098       // printf("\n");
00099     }
00100     FillAsicHR2(bs);
00101   }
00102 }
00103
00104 void DIFSlowControl::FillAsicHR1(const std::bitset<72 * 8>& bs)
00105 {
00106   // Asic Id
00107   int asicid{0};
00108   for(int j = 0; j < 8; j++)
00109     if(bs[j + 9] != 0) asicid += (1 « (7 - j));
00110   std::map<std::string, int> mAsic;
00111   // Slow Control
00112   mAsic["SSC0"]          = static_cast<int>(bs[575]);
00113   mAsic["SSC1"]          = static_cast<int>(bs[574]);
00114   mAsic["SSC2"]          = static_cast<int>(bs[573]);
00115   mAsic["Choix_caisson"] = static_cast<int>(bs[572]);
00116   mAsic["SW_50k"]        = static_cast<int>(bs[571]);
00117   mAsic["SW_100k"]       = static_cast<int>(bs[570]);
00118   mAsic["SW_100f"]       = static_cast<int>(bs[569]);
00119   mAsic["SW_50f"]        = static_cast<int>(bs[568]);
00120
00121   mAsic["Valid_DC"]  = static_cast<int>(bs[567]);
00122   mAsic["ON_Discri"] = static_cast<int>(bs[566]);
00123   mAsic["ON_Fsb"]    = static_cast<int>(bs[565]);
00124   mAsic["ON_Otaq"]   = static_cast<int>(bs[564]);
00125   mAsic["ON_W"]      = static_cast<int>(bs[563]);
00126   mAsic["ON_Ss"]     = static_cast<int>(bs[562]);
00127   mAsic["ON_Buf"]    = static_cast<int>(bs[561]);
00128   mAsic["ON_Paf"]    = static_cast<int>(bs[560]);
00129   // Gain
00130   for(int i = 0; i < 64; i++)
00131   {
00132     int gain{0};
00133     for(int j = 0; j < 6; j++)
00134       if(bs[176 + i * 6 + j] != 0) gain += (1 « j);
00135     mAsic["Channel_" + std::to_string(i) + "_" + "Gain"]       = gain;
```

```
00136     mAsic["Channel_" + std::to_string(i) + "_" + "cTest"]      = bs[112 + i];
00137     mAsic["Channel_" + std::to_string(i) + "_" + "Valid_trig"] = static_cast<int>(bs[25 + i]);
00138   }
00139
00140   mAsic["ON_Otabg"]  = static_cast<int>(bs[111]);
00141   mAsic["ON_Dac"]    = static_cast<int>(bs[110]);
00142   mAsic["ON_Otadac"] = static_cast<int>(bs[109]);
00143   // DAC
00144   int dac1{0};
00145   for(int j = 0; j < 10; j++)
00146     if(bs[j + 99] != 0) dac1 += (1 << j);
00147   mAsic["DAC1"] = dac1;
00148   int dac0{0};
00149   for(int j = 0; j < 10; j++)
00150     if(bs[j + 89] != 0) dac0 += (1 << j);
00151   mAsic["DAC0"]            = dac0;
00152   mAsic["EN_Raz_Ext"]      = static_cast<int>(bs[23]);
00153   mAsic["EN_Raz_Int"]      = static_cast<int>(bs[22]);
00154   mAsic["EN_Out_Raz_Int"]  = static_cast<int>(bs[21]);
00155   mAsic["EN_Trig_Ext"]     = static_cast<int>(bs[20]);
00156   mAsic["EN_Trig_Int"]     = static_cast<int>(bs[19]);
00157   mAsic["EN_Out_Trig_Int"] = static_cast<int>(bs[18]);
00158   mAsic["Bypass_Chip"]     = static_cast<int>(bs[17]);
00159   mAsic["HardrocHeader"]   = static_cast<int>(asicid);
00160   mAsic["EN_Out_Discri"]   = static_cast<int>(bs[8]);
00161   mAsic["EN_Transmit_On"]  = static_cast<int>(bs[7]);
00162   mAsic["EN_Dout"]         = static_cast<int>(bs[6]);
00163   mAsic["EN_RamFull"]      = static_cast<int>(bs[5]);
00164   m_MapSC[asicid]          = mAsic;
00165 }
00166
00167 void DIFSlowControl::FillAsicHR2(const std::bitset<109 * 8>& bs)
00168 {
00169   int asicid{0};
00170   for(int j = 0; j < 8; j++)
00171     if(bs[j + (108 - 7) * 8 + 2] != 0) asicid += (1 << (7 - j));
00172   std::map<std::string, int> mAsic;
00173   for(int i = 0; i < 64; i++)
00174   {
00175     int gain{0};
00176     int mask{0};
00177     mAsic["Channel_" + std::to_string(i) + "_" + "cTest"] = bs[i];
00178     for(int j = 0; j < 8; j++)
00179       if(bs[64 + i * 8 + j] != 0) gain += (1 << j);
00180     mAsic["Channel_" + std::to_string(i) + "_" + "Gain"] = gain;
00181     for(int j = 0; j < 3; j++)
00182       if(bs[8 * 77 + 2 + i * 3 + j] != 0) mask += (1 << j);
00183     mAsic["Channel_" + std::to_string(i) + "_" + "Mask"] = mask;
00184   }
00185   mAsic["PwrOnPA"] = static_cast<int>(bs[8 * 72]);
00186   mAsic["Cmdb3SS"] = static_cast<int>(bs[8 * 72 + 1]);
00187   mAsic["Cmdb2SS"] = static_cast<int>(bs[8 * 72 + 2]);
00188   mAsic["Cmdb1SS"] = static_cast<int>(bs[8 * 72 + 3]);
00189   mAsic["Cmdb0SS"] = static_cast<int>(bs[8 * 72 + 4]);
00190   mAsic["SwSsc0"]  = static_cast<int>(bs[8 * 72 + 5]);
00191   mAsic["SwSsc1"]  = static_cast<int>(bs[8 * 72 + 6]);
00192   mAsic["SwSsc2"]  = static_cast<int>(bs[8 * 72 + 7]);
00193
00194   mAsic["PwrOnBuff"] = static_cast<int>(bs[8 * 73]);
00195   mAsic["PwrOnSS"]   = static_cast<int>(bs[8 * 73 + 1]);
00196   mAsic["PwrOnW"]    = static_cast<int>(bs[8 * 73 + 2]);
00197   mAsic["Cmdb3Fsb2"] = static_cast<int>(bs[8 * 73 + 3]);
00198   mAsic["Cmdb2Fsb2"] = static_cast<int>(bs[8 * 73 + 4]);
00199   mAsic["Cmdb1Fsb2"] = static_cast<int>(bs[8 * 73 + 5]);
00200   mAsic["Cmdb0Fsb2"] = static_cast<int>(bs[8 * 73 + 6]);
00201   mAsic["Sw50k2"]    = static_cast<int>(bs[8 * 73 + 7]);
00202
00203   mAsic["Sw100k2"]   = static_cast<int>(bs[8 * 74]);
00204   mAsic["Sw100f2"]   = static_cast<int>(bs[8 * 74 + 1]);
00205   mAsic["Sw50f2"]    = static_cast<int>(bs[8 * 74 + 2]);
00206   mAsic["Cmdb3Fsb1"] = static_cast<int>(bs[8 * 74 + 3]);
00207   mAsic["Cmdb2Fsb1"] = static_cast<int>(bs[8 * 74 + 4]);
00208   mAsic["Cmdb1Fsb1"] = static_cast<int>(bs[8 * 74 + 5]);
00209   mAsic["Cmdb0Fsb1"] = static_cast<int>(bs[8 * 74 + 6]);
00210   mAsic["Sw50k1"]    = static_cast<int>(bs[8 * 74 + 7]);
00211
00212   mAsic["Sw100k1"]   = static_cast<int>(bs[8 * 75]);
00213   mAsic["Sw100f1"]   = static_cast<int>(bs[8 * 75 + 1]);
00214   mAsic["Sw50f1"]    = static_cast<int>(bs[8 * 75 + 2]);
00215   mAsic["Sel0"]      = static_cast<int>(bs[8 * 75 + 3]);
00216   mAsic["Sel11"]     = static_cast<int>(bs[8 * 75 + 4]);
00217   mAsic["PwrOnFsb"]  = static_cast<int>(bs[8 * 75 + 5]);
00218   mAsic["PwrOnFsb1"] = static_cast<int>(bs[8 * 75 + 6]);
00219   mAsic["PwrOnFsb2"] = static_cast<int>(bs[8 * 75 + 7]);
00220
00221   mAsic["Sw50k0"]      = static_cast<int>(bs[8 * 76]);
00222   mAsic["Sw100k0"]     = static_cast<int>(bs[8 * 76 + 1]);
```

```
00223   mAsic["Sw100f0"]     = static_cast<int>(bs[8 * 76 + 2]);
00224   mAsic["Sw50f0"]      = static_cast<int>(bs[8 * 76 + 3]);
00225   mAsic["EnOtaQ"]      = static_cast<int>(bs[8 * 76 + 4]);
00226   mAsic["OtaQ_PwrADC"] = static_cast<int>(bs[8 * 76 + 5]);
00227   mAsic["Discri_PwrA"] = static_cast<int>(bs[8 * 76 + 6]);
00228   mAsic["Discri2"]     = static_cast<int>(bs[8 * 76 + 7]);
00229
00230   mAsic["Discri1"]      = static_cast<int>(bs[8 * 77]);
00231   mAsic["RS_or_Discri"] = static_cast<int>(bs[8 * 77 + 1]);
00232
00233   mAsic["Header"] = asicid;
00234   for(int i = 0; i < 3; i++)
00235   {
00236     int B = 0;
00237     for(int j = 0; j < 10; j++)
00238       if(bs[8 * 102 + 2 + i * 10 + j] != 0) B += (1 << j);
00239     mAsic["B" + std::to_string(i)] = B;
00240   }
00241
00242   mAsic["Smalldac"]  = static_cast<int>(bs[8 * 106]);
00243   mAsic["DacSw"]     = static_cast<int>(bs[8 * 106 + 1]);
00244   mAsic["OtagBgSw"]  = static_cast<int>(bs[8 * 106 + 2]);
00245   mAsic["Trig2b"]    = static_cast<int>(bs[8 * 106 + 3]);
00246   mAsic["Trig1b"]    = static_cast<int>(bs[8 * 106 + 4]);
00247   mAsic["Trig0b"]    = static_cast<int>(bs[8 * 106 + 5]);
00248   mAsic["EnTrigOut"] = static_cast<int>(bs[8 * 106 + 6]);
00249   mAsic["DiscrOrOr"] = static_cast<int>(bs[8 * 106 + 7]);
00250
00251   mAsic["TrigExtVal"]   = static_cast<int>(bs[8 * 107]);
00252   mAsic["RazChnIntVal"] = static_cast<int>(bs[8 * 107 + 1]);
00253   mAsic["RazChnExtVal"] = static_cast<int>(bs[8 * 107 + 2]);
00254   mAsic["ScOn"]         = static_cast<int>(bs[8 * 107 + 3]);
00255   mAsic["CLKMux"]       = static_cast<int>(bs[8 * 107 + 4]);
00256
00257   // EnOCDout1b   EnOCDout2b   EnOCTransmitOn1b   EnOCTransmitOn2b   EnOCChipsatb   SelStartReadout
      SelEndReadout
00258   mAsic["SelEndReadout"]    = static_cast<int>(bs[8 * 108 + 1]);
00259   mAsic["SelStartReadout"]  = static_cast<int>(bs[8 * 108 + 2]);
00260   mAsic["EnOCChipsatb"]     = static_cast<int>(bs[8 * 108 + 3]);
00261   mAsic["EnOCTransmitOn2b"] = static_cast<int>(bs[8 * 108 + 4]);
00262   mAsic["EnOCTransmitOn1b"] = static_cast<int>(bs[8 * 108 + 5]);
00263   mAsic["EnOCDout2b"]       = static_cast<int>(bs[8 * 108 + 6]);
00264   mAsic["EnOCDout1b"]       = static_cast<int>(bs[8 * 108 + 7]);
00265   m_MapSC[asicid]           = mAsic;
00266 }
```

5.31 /home/runner/work/streamout/streamout/libs/core/src/DIFUnpacker.cc File Reference

```
#include "DIFUnpacker.h"
#include "Words.h"
#include <bitset>
#include <cstdint>
#include <iostream>
```

5.31.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file DIFUnpacker.cc.

5.32 DIFUnpacker.cc

Go to the documentation of this file.

```
00001
00005 #include "DIFUnpacker.h"
00006
00007 #include "Words.h"
00008
00009 #include <bitset>
00010 #include <cstdint>
00011 #include <iostream>
00012
00013 std::uint64_t DIFUnpacker::GrayToBin(const std::uint64_t& n)
00014 {
00015   std::uint64_t ish{1};
00016   std::uint64_t anss{n};
00017   std::uint64_t idiv{0};
00018   std::uint64_t ishmax{sizeof(std::uint64_t) * 8};
00019   while(true)
00020   {
00021     idiv = anss » ish;
00022     anss ^= idiv;
00023     if(idiv <= 1 || ish == ishmax) return anss;
00024     ish «= 1;
00025   }
00026 }
00027
00028 std::uint32_t DIFUnpacker::getStartOfDIF(const unsigned char* cbuf, const std::uint32_t& size_buf,
       const std::uint32_t& start)
00029 {
00030   std::uint32_t id0{0};
00031   for(std::uint32_t i = start; i < size_buf; i++)
00032   {
00033     if(cbuf[i] != DU::START_OF_DIF && cbuf[i] != DU::START_OF_DIF_TEMP) continue;
00034     id0 = i;
00035     // if (cbuf[id0+DU::ID_SHIFT]>0xFF) continue;
00036     break;
00037   }
00038   return id0;
00039 }
00040
00041 std::uint32_t DIFUnpacker::getID(const unsigned char* cb, const std::uint32_t& idx) { return cb[idx +
       DU::ID_SHIFT]; }
00042
00043 std::uint32_t DIFUnpacker::getDTC(const unsigned char* cb, const std::uint32_t& idx) { return (cb[idx
       + DU::DTC_SHIFT] « 24) + (cb[idx + DU::DTC_SHIFT + 1] « 16) + (cb[idx + DU::DTC_SHIFT + 2] « 8) +
       cb[idx + DU::DTC_SHIFT + 3]; }
00044
00045 std::uint32_t DIFUnpacker::getGTC(const unsigned char* cb, const std::uint32_t& idx) { return (cb[idx
       + DU::GTC_SHIFT] « 24) + (cb[idx + DU::GTC_SHIFT + 1] « 16) + (cb[idx + DU::GTC_SHIFT + 2] « 8) +
       cb[idx + DU::GTC_SHIFT + 3]; }
00046
00047 std::uint64_t DIFUnpacker::getAbsoluteBCID(const unsigned char* cb, const std::uint32_t& idx)
00048 {
00049   std::uint64_t Shift{16777216ULL};  // to shift the value from the 24 first bits
00050   std::uint64_t pos{idx + DU::ABCID_SHIFT};
00051   std::uint64_t LBC = ((cb[pos] « 16) | (cb[pos + 1] « 8) | (cb[pos + 2])) * Shift + ((cb[pos + 3] «
       16) | (cb[pos + 4] « 8) | (cb[pos + 5]));
00052   return LBC;
00053 }
00054
00055 std::uint32_t DIFUnpacker::getBCID(const unsigned char* cb, const std::uint32_t& idx) { return (cb[idx
       + DU::BCID_SHIFT] « 16) + (cb[idx + DU::BCID_SHIFT + 1] « 8) + cb[idx + DU::BCID_SHIFT + 2]; }
00056 std::uint32_t DIFUnpacker::getLines(const unsigned char* cb, const std::uint32_t& idx) { return
       (cb[idx + DU::LINES_SHIFT] » 4) & 0x5; }
00057
00058 bool DIFUnpacker::hasLine(const std::uint32_t& line, const unsigned char* cb, const std::uint32_t&
       idx) { return ((cb[idx + DU::LINES_SHIFT] » line) & 0x1); }
00059
00060 std::uint32_t DIFUnpacker::getTASU1(const unsigned char* cb, const std::uint32_t& idx) { return
       (cb[idx + DU::TASU1_SHIFT] « 24) + (cb[idx + DU::TASU1_SHIFT + 1] « 16) + (cb[idx + DU::TASU1_SHIFT +
       2] « 8) + cb[idx + DU::TASU1_SHIFT + 3]; }
00061
00062 std::uint32_t DIFUnpacker::getTASU2(const unsigned char* cb, const std::uint32_t& idx) { return
       (cb[idx + DU::TASU2_SHIFT] « 24) + (cb[idx + DU::TASU2_SHIFT + 1] « 16) + (cb[idx + DU::TASU2_SHIFT +
       2] « 8) + cb[idx + DU::TASU2_SHIFT + 3]; }
00063
00064 std::uint32_t DIFUnpacker::getTDIF(const unsigned char* cb, const std::uint32_t& idx) { return (cb[idx
       + DU::TDIF_SHIFT]); }
00065
00066 bool DIFUnpacker::hasTemperature(const unsigned char* cb, const std::uint32_t& idx) { return (cb[idx]
       == DU::START_OF_DIF_TEMP); }
00067
00068 bool DIFUnpacker::hasAnalogReadout(const unsigned char* cb, const std::uint32_t& idx) { return
       (DIFUnpacker::getLines(cb, idx) != 0); }
```

```
00069
00070 std::uint32_t DIFUnpacker::getFrameAsicHeader(const unsigned char* framePtr) { return
       (framePtr[DU::FRAME_ASIC_HEADER_SHIFT]); }
00071
00072 std::uint32_t DIFUnpacker::getFrameBCID(const unsigned char* framePtr)
00073 {
00074   std::uint32_t igray = (framePtr[DU::FRAME_BCID_SHIFT] << 16) + (framePtr[DU::FRAME_BCID_SHIFT + 1] <<
       8) + framePtr[DU::FRAME_BCID_SHIFT + 2];
00075   return DIFUnpacker::GrayToBin(igray);
00076 }
00077
00078 bool DIFUnpacker::getFramePAD(const unsigned char* framePtr, const std::uint32_t& ip)
00079 {
00080   std::uint32_t* iframe{(std::uint32_t*)&framePtr[DU::FRAME_DATA_SHIFT]};
00081   return ((iframe[3 - ip / 32] >> (ip % 32)) & 0x1);
00082 }
00083
00084 bool DIFUnpacker::getFrameLevel(const unsigned char* framePtr, const std::uint32_t& ip, const
       std::uint32_t& level) { return ((framePtr[DU::FRAME_DATA_SHIFT + ((3 - ip / 16) * 4 + (ip % 16) / 4)]
       >> (7 - (((ip % 16) % 4) * 2 + level))) & 0x1); }
00085
00086 std::uint32_t DIFUnpacker::getAnalogPtr(std::vector<unsigned char*>& vLines, unsigned char* cb, const
       std::uint32_t& idx)
00087 {
00088   std::uint32_t fshift{idx};
00089   if(cb[fshift] != DU::START_OF_LINES) return fshift;
00090   fshift++;
00091   while(cb[fshift] != DU::END_OF_LINES)
00092   {
00093     vLines.push_back(&cb[fshift]);
00094     std::uint32_t nchip{cb[fshift]};
00095     fshift += 1 + nchip * 64 * 2;
00096   }
00097   return fshift++;
00098 }
00099
00100 std::uint32_t DIFUnpacker::getFramePtr(std::vector<unsigned char*>& vFrame, std::vector<unsigned
       char*>& vLines, const std::uint32_t& max_size, unsigned char* cb, const std::uint32_t& idx)
00101 {
00102   std::uint32_t fshift{0};
00103   if(DATA_FORMAT_VERSION >= 13)
00104   {
00105     fshift = idx + DU::LINES_SHIFT + 1;
00106     if(DIFUnpacker::hasTemperature(cb, idx)) fshift = idx + DU::TDIF_SHIFT + 1;
      // jenlev 1
00107     if(DIFUnpacker::hasAnalogReadout(cb, idx)) fshift = DIFUnpacker::getAnalogPtr(vLines, cb, fshift);
      // to be implemented
00108   }
00109   else
00110     std::uint32_t fshift = idx + DU::BCID_SHIFT + 3;
00111   if(cb[fshift] != DU::START_OF_FRAME)
00112   {
00113     std::cout << "This is not a start of frame " << cb[fshift] << "\n";
00114     return fshift;
00115   }
00116   do {
00117     // printf("fshift %d and %d \n",fshift,max_size);
00118     if(cb[fshift] == DU::END_OF_DIF) return fshift;
00119     if(cb[fshift] == DU::START_OF_FRAME) fshift++;
00120     if(cb[fshift] == DU::END_OF_FRAME)
00121     {
00122       fshift++;
00123       continue;
00124     }
00125     std::uint32_t header = DIFUnpacker::getFrameAsicHeader(&cb[fshift]);
00126     if(header == DU::END_OF_FRAME) return (fshift + 2);
00127     // std::cout<<header<<" "<<fshift<<std::endl;
00128     if(header < 1 || header > 48) { throw header + " Header problem " + fshift; }
00129     vFrame.push_back(&cb[fshift]);
00130     fshift += DU::FRAME_SIZE;
00131     if(fshift > max_size)
00132     {
00133       std::cout << "fshift " << fshift << " exceed " << max_size << "\n";
00134       return fshift;
00135     }
00136     if(cb[fshift] == DU::END_OF_FRAME) fshift++;
00137   } while(true);
00138 }
00139
00140 void DIFUnpacker::dumpFrameOld(const unsigned char* buf)
00141 {
00142   bool         PAD[128];
00143   bool         l0[64];
00144   bool         l1[64];
00145   std::uint8_t un{1};
00146   for(std::size_t ip = 0; ip < 128; ip++) { PAD[ip] = false; }  // init PADs
00147   std::uint32_t idx1{4};
```

```
00148   for(int ik = 0; ik < 4; ik++)
00149   {
00150     std::uint32_t PadEtat{swap_bytes(&buf[idx1])};
00151     idx1 += 4;
00152     for(int e = 0; e < 32; e++)
00153     {
00154       PAD[((3 - ik) * 32) + (31 - e)] = PadEtat & un;  // binary operation
00155       PadEtat                         = PadEtat » 1;  // décalage des bit de 1
00156     }
00157   }
00158   // fill bool arrays
00159   for(int p = 0; p < 64; p++)
00160   {
00161     l0[p] = static_cast<bool>(PAD[(2 * p)]);      // _Lev0 (PAD paire)
00162     l1[p] = static_cast<bool>(PAD[(2 * p) + 1]);  // _Lev1 (PAD impaires)
00163   }
00164   std::bitset<64> bs0(0);
00165   std::bitset<64> bs1(0);
00166   for(std::uint32_t ip = 0; ip < 64; ip++)
00167   {
00168     bs0.set(ip, l0[ip]);
00169     bs1.set(ip, l1[ip]);
00170   }
00171   std::cout « "\t \t" « bs0 « std::endl;
00172   std::cout « "\t \t" « bs1 « std::endl;
00173 }
00174
00175 std::uint32_t DIFUnpacker::swap_bytes(const unsigned char* buf)
00176 {
00177   unsigned char Swapped[4];
00178   for(std::size_t i = 0; i < 4; i++) Swapped[i] = buf[4 - 1 - i];
00179   return *reinterpret_cast<std::uint32_t*>(&Swapped[0]);
00180 }
```

5.33 /home/runner/work/streamout/streamout/libs/core/src/↩ Formatters.cc File Reference

```
#include "Formatters.h"
#include "Bits.h"
#include "Buffer.h"
#include <fmt/format.h>
```

Functions

- std::string to_dec (const Buffer &b, const std::size_t &begin, const std::size_t &end)
- std::string to_dec (const bit8_t &b)
- std::string to_dec (const bit16_t &b)
- std::string to_dec (const bit32_t &b)
- std::string to_dec (const bit64_t &b)
- std::string to_hex (const Buffer &b, const std::size_t &begin, const std::size_t &end)
- std::string to_hex (const bit8_t &b)
- std::string to_hex (const bit16_t &b)
- std::string to_hex (const bit32_t &b)
- std::string to_hex (const bit64_t &b)
- std::string to_bin (const Buffer &b, const std::size_t &begin, const std::size_t &end)
- std::string to_bin (const bit8_t &b)
- std::string to_bin (const bit16_t &b)
- std::string to_bin (const bit32_t &b)
- std::string to_bin (const bit64_t &b)
- std::string to_oct (const Buffer &b, const std::size_t &begin, const std::size_t &end)
- std::string to_oct (const bit8_t &b)
- std::string to_oct (const bit16_t &b)
- std::string to_oct (const bit32_t &b)
- std::string to_oct (const bit64_t &b)

5.33.1 Function Documentation

5.33.1.1 to_bin() [1/5]

```
std::string to_bin (
            const bit16_t & b )
```

Definition at line 92 of file Formatters.cc.

```
00093 {
00094   return fmt::format("{:#016b}",b);
00095 }
```

5.33.1.2 to_bin() [2/5]

```
std::string to_bin (
            const bit32_t & b )
```

Definition at line 97 of file Formatters.cc.

```
00098 {
00099   return fmt::format("{:#032b}",b);
00100 }
```

5.33.1.3 to_bin() [3/5]

```
std::string to_bin (
            const bit64_t & b )
```

Definition at line 102 of file Formatters.cc.

```
00103 {
00104   return fmt::format("{:#064b}",b);
00105 }
```

5.33.1.4 to_bin() [4/5]

```
std::string to_bin (
            const bit8_t & b )
```

Definition at line 87 of file Formatters.cc.

```
00088 {
00089   return fmt::format("{:#08b}",b);
00090 }
```

5.33.1.5 to_bin() [5/5]

```
std::string to_bin (
              const Buffer & b,
              const std::size_t & begin,
              const std::size_t & end )
```

Definition at line 74 of file Formatters.cc.

```
00075 {
00076   std::size_t iend =end;
00077   if(iend==-1) iend=b.size();
00078   std::string ret;
00079   for(std::size_t k = begin; k < iend; k++)
00080   {
00081     ret+= to_bin(b[k]);
00082     ret+=" - ";
00083   }
00084   return ret;
00085 }
```

5.33.1.6 to_dec() [1/5]

```
std::string to_dec (
              const bit16_t & b )
```

Definition at line 26 of file Formatters.cc.

```
00027 {
00028   return fmt::format("{:#016d}",b);
00029 }
```

5.33.1.7 to_dec() [2/5]

```
std::string to_dec (
              const bit32_t & b )
```

Definition at line 31 of file Formatters.cc.

```
00032 {
00033   return fmt::format("{:#032d}",b);
00034 }
```

5.33.1.8 to_dec() [3/5]

```
std::string to_dec (
              const bit64_t & b )
```

Definition at line 36 of file Formatters.cc.

```
00037 {
00038   return fmt::format("{:#064d}",b);
00039 }
```

5.33.1.9 to_dec() [4/5]

```
std::string to_dec (
              const bit8_t & b )
```

Definition at line 21 of file Formatters.cc.

```
00022 {
00023     return fmt::format("{:#08d}",b);
00024 }
```

5.33.1.10 to_dec() [5/5]

```
std::string to_dec (
              const Buffer & b,
              const std::size_t & begin,
              const std::size_t & end )
```

Definition at line 8 of file Formatters.cc.

```
00009 {
00010   std::size_t iend =end;
00011   if(iend==-1) iend=b.size();
00012   std::string ret;
00013   for(std::size_t k = begin; k < iend; k++)
00014   {
00015     ret+= to_dec(b[k]);
00016     ret+=" - ";
00017   }
00018   return ret;
00019 }
```

5.33.1.11 to_hex() [1/5]

```
std::string to_hex (
              const bit16_t & b )
```

Definition at line 59 of file Formatters.cc.

```
00060 {
00061   return fmt::format("{:#016x}",b);
00062 }
```

5.33.1.12 to_hex() [2/5]

```
std::string to_hex (
              const bit32_t & b )
```

Definition at line 64 of file Formatters.cc.

```
00065 {
00066   return fmt::format("{:#032x}",b);
00067 }
```

5.33.1.13 to_hex() [3/5]

```
std::string to_hex (
            const bit64_t & b )
```

Definition at line 69 of file Formatters.cc.

```
00070 {
00071   return fmt::format("{:#064x}",b);
00072 }
```

5.33.1.14 to_hex() [4/5]

```
std::string to_hex (
            const bit8_t & b )
```

Definition at line 54 of file Formatters.cc.

```
00055 {
00056   return fmt::format("{:#08x}",b);
00057 }
```

5.33.1.15 to_hex() [5/5]

```
std::string to_hex (
            const Buffer & b,
            const std::size_t & begin,
            const std::size_t & end )
```

Definition at line 41 of file Formatters.cc.

```
00042 {
00043   std::size_t iend =end;
00044   if(iend==-1) iend=b.size();
00045   std::string ret;
00046   for(std::size_t k = begin; k < iend; k++)
00047   {
00048     ret+= to_hex(b[k]);
00049     ret+=" - ";
00050   }
00051   return ret;
00052 }
```

5.33.1.16 to_oct() [1/5]

```
std::string to_oct (
            const bit16_t & b )
```

Definition at line 125 of file Formatters.cc.

```
00126 {
00127   return fmt::format("{:#016o}",b);
00128 }
```

5.33.1.17 to_oct() [2/5]

```
std::string to_oct (
              const bit32_t & b )
```

Definition at line 130 of file Formatters.cc.

```
00131 {
00132   return fmt::format("{:#032o}",b);
00133 }
```

5.33.1.18 to_oct() [3/5]

```
std::string to_oct (
              const bit64_t & b )
```

Definition at line 135 of file Formatters.cc.

```
00136 {
00137   return fmt::format("{:#064o}",b);
00138 }
```

5.33.1.19 to_oct() [4/5]

```
std::string to_oct (
              const bit8_t & b )
```

Definition at line 120 of file Formatters.cc.

```
00121 {
00122   return fmt::format("{:#08o}",b);
00123 }
```

5.33.1.20 to_oct() [5/5]

```
std::string to_oct (
              const Buffer & b,
              const std::size_t & begin,
              const std::size_t & end )
```

Definition at line 107 of file Formatters.cc.

```
00108 {
00109   std::size_t iend =end;
00110   if(iend==-1) iend=b.size();
00111   std::string ret;
00112   for(std::size_t k = begin; k < iend; k++)
00113   {
00114     ret+= to_oct(b[k]);
00115     ret+=" - ";
00116   }
00117   return ret;
00118 }
```

5.34 Formatters.cc

Go to the documentation of this file.

```
00001 #include "Formatters.h"
00002
00003 #include "Bits.h"
00004 #include "Buffer.h"
00005
00006 #include <fmt/format.h>
00007
00008 std::string to_dec(const Buffer& b,const std::size_t& begin,const std::size_t& end)
00009 {
00010   std::size_t iend =end;
00011   if(iend==-1) iend=b.size();
00012   std::string ret;
00013   for(std::size_t k = begin; k < iend; k++)
00014   {
00015     ret+= to_dec(b[k]);
00016     ret+=" - ";
00017   }
00018   return ret;
00019 }
00020
00021 std::string to_dec(const bit8_t& b)
00022 {
00023     return fmt::format("{:#08d}",b);
00024 }
00025
00026 std::string to_dec(const bit16_t& b)
00027 {
00028   return fmt::format("{:#016d}",b);
00029 }
00030
00031 std::string to_dec(const bit32_t& b)
00032 {
00033   return fmt::format("{:#032d}",b);
00034 }
00035
00036 std::string to_dec(const bit64_t& b)
00037 {
00038   return fmt::format("{:#064d}",b);
00039 }
00040
00041 std::string to_hex(const Buffer& b,const std::size_t& begin,const std::size_t& end)
00042 {
00043   std::size_t iend =end;
00044   if(iend==-1) iend=b.size();
00045   std::string ret;
00046   for(std::size_t k = begin; k < iend; k++)
00047   {
00048     ret+= to_hex(b[k]);
00049     ret+=" - ";
00050   }
00051   return ret;
00052 }
00053
00054 std::string to_hex(const bit8_t& b)
00055 {
00056   return fmt::format("{:#08x}",b);
00057 }
00058
00059 std::string to_hex(const bit16_t& b)
00060 {
00061   return fmt::format("{:#016x}",b);
00062 }
00063
00064 std::string to_hex(const bit32_t& b)
00065 {
00066   return fmt::format("{:#032x}",b);
00067 }
00068
00069 std::string to_hex(const bit64_t& b)
00070 {
00071   return fmt::format("{:#064x}",b);
00072 }
00073
00074 std::string to_bin(const Buffer& b,const std::size_t& begin,const std::size_t& end)
00075 {
00076   std::size_t iend =end;
00077   if(iend==-1) iend=b.size();
00078   std::string ret;
00079   for(std::size_t k = begin; k < iend; k++)
00080   {
00081     ret+= to_bin(b[k]);
00082     ret+=" - ";
```

```
00083   }
00084   return ret;
00085 }
00086
00087 std::string to_bin(const bit8_t& b)
00088 {
00089   return fmt::format("{:#08b}",b);
00090 }
00091
00092 std::string to_bin(const bit16_t& b)
00093 {
00094   return fmt::format("{:#016b}",b);
00095 }
00096
00097 std::string to_bin(const bit32_t& b)
00098 {
00099   return fmt::format("{:#032b}",b);
00100 }
00101
00102 std::string to_bin(const bit64_t& b)
00103 {
00104   return fmt::format("{:#064b}",b);
00105 }
00106
00107 std::string to_oct(const Buffer& b,const std::size_t& begin,const std::size_t& end)
00108 {
00109   std::size_t iend =end;
00110   if(iend==-1) iend=b.size();
00111   std::string ret;
00112   for(std::size_t k = begin; k < iend; k++)
00113   {
00114     ret+= to_oct(b[k]);
00115     ret+=" - ";
00116   }
00117   return ret;
00118 }
00119
00120 std::string to_oct(const bit8_t& b)
00121 {
00122   return fmt::format("{:#08o}",b);
00123 }
00124
00125 std::string to_oct(const bit16_t& b)
00126 {
00127   return fmt::format("{:#016o}",b);
00128 }
00129
00130 std::string to_oct(const bit32_t& b)
00131 {
00132   return fmt::format("{:#032o}",b);
00133 }
00134
00135 std::string to_oct(const bit64_t& b)
00136 {
00137   return fmt::format("{:#064o}",b);
00138 }
```

5.35 /home/runner/work/streamout/streamout/libs/core/src/SDHCAL_↩ buffer_LoopCounter.cc File Reference

```
#include "SDHCAL_buffer_LoopCounter.h"
#include <spdlog/spdlog.h>
```

5.35.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file SDHCAL_buffer_LoopCounter.cc.

5.36 SDHCAL_buffer_LoopCounter.cc

Go to the documentation of this file.

```
00001
00005 #include "SDHCAL_buffer_LoopCounter.h"
00006
00007 #include <spdlog/spdlog.h>
00008
00009 void SDHCAL_buffer_LoopCounter::printAllCounters(const std::shared_ptr<spdlog::logger>& logger)
00010 {
00011   spdlog::level::level_enum level = logger->level();
00012   logger->set_level(spdlog::level::trace);
00013   logger->critical("BUFFER LOOP FINAL STATISTICS : ");
00014   printCounter("Start of DIF header", DIFStarter, logger);
00015   printCounter("Value after DIF data are processed", DIFPtrValueAtReturnedPos, logger);
00016   printCounter("Size remaining in buffer after end of DIF data", SizeAfterDIFPtr, logger);
00017   logger->critical("Number of Slow Control found {}  out of which {} are bad", hasSlowControl,
       hasBadSlowControl);
00018   printCounter("Size remaining after all of data have been processed", SizeAfterAllData, logger);
00019   printCounter("Number on non zero values in end of data buffer", NonZeroValusAtEndOfData, logger);
00020   logger->set_level(level);
00021 }
00022
00023 void SDHCAL_buffer_LoopCounter::printCounter(const std::string& description, const std::map<int, int>&
       m, const std::shared_ptr<spdlog::logger>& logger)
00024 {
00025   std::string out{"statistics for " + description + " : "};
00026   for(std::map<int, int>::const_iterator it = m.begin(); it != m.end(); it++)
00027   {
00028     if(it != m.begin()) out += ",";
00029     out += " [" + std::to_string(it->first) + "]=" + std::to_string(it->second);
00030   }
00031   logger->critical(out);
00032 }
```

5.37 /home/runner/work/streamout/streamout/libs/core/src/SDHCAL_RawBuffer_Navigator.cc File Reference

```
#include "SDHCAL_RawBuffer_Navigator.h"
#include <iostream>
```

5.37.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file SDHCAL_RawBuffer_Navigator.cc.

5.38 SDHCAL_RawBuffer_Navigator.cc

Go to the documentation of this file.

```
00001
00005 #include "SDHCAL_RawBuffer_Navigator.h"
00006
00007 #include <iostream>
00008 int SDHCAL_RawBuffer_Navigator::m_Start = 92;
00009
00010 void SDHCAL_RawBuffer_Navigator::StartAt(const int& start)
00011 {
00012   if(start >= 0) m_Start = start;
00013 }
```

```
00014
00015 SDHCAL_RawBuffer_Navigator::SDHCAL_RawBuffer_Navigator(const Buffer& b, const int& start) :
       m_Buffer(b)
00016 {
00017   StartAt(start);
00018   m_DIFstartIndex = DIFUnpacker::getStartOfDIF(m_Buffer.begin(), m_Buffer.size(), m_Start);
00019 }
00020
00021 SDHCAL_RawBuffer_Navigator::~SDHCAL_RawBuffer_Navigator()
00022 {
00023   if(m_TheDIFPtr != nullptr) delete m_TheDIFPtr;
00024 }
00025
00026 bool SDHCAL_RawBuffer_Navigator::validBuffer() { return m_DIFstartIndex != 0; }
00027
00028 std::uint32_t SDHCAL_RawBuffer_Navigator::getStartOfDIF() { return m_DIFstartIndex; }
00029
00030 unsigned char* SDHCAL_RawBuffer_Navigator::getDIFBufferStart() { return
       &(m_Buffer.begin()[m_DIFstartIndex]); }
00031
00032 std::uint32_t SDHCAL_RawBuffer_Navigator::getDIFBufferSize() { return m_Buffer.size() -
       m_DIFstartIndex; }
00033
00034 Buffer SDHCAL_RawBuffer_Navigator::getDIFBuffer() { return Buffer(getDIFBufferStart(),
       getDIFBufferSize()); }
00035
00036 DIFPtr* SDHCAL_RawBuffer_Navigator::getDIFPtr()
00037 {
00038   if(m_TheDIFPtr == nullptr) m_TheDIFPtr = new DIFPtr(getDIFBufferStart(), getDIFBufferSize());
00039   return m_TheDIFPtr;
00040 }
00041
00042 std::uint32_t SDHCAL_RawBuffer_Navigator::getEndOfDIFData() { return
       getDIFPtr()->getGetFramePtrReturn() + 3; }
00043
00044 std::uint32_t SDHCAL_RawBuffer_Navigator::getSizeAfterDIFPtr() { return getDIFBufferSize() -
       getDIFPtr()->getGetFramePtrReturn(); }
00045
00046 uint32_t SDHCAL_RawBuffer_Navigator::getDIF_CRC()
00047 {
00048   uint32_t i{getEndOfDIFData()};
00049   uint32_t ret{0};
00050   ret |= ((m_Buffer.begin()[i - 2]) « 8);
00051   ret |= m_Buffer.begin()[i - 1];
00052   return ret;
00053 }
00054
00055 bool SDHCAL_RawBuffer_Navigator::hasSlowControlData() { return getDIFBufferStart()[getEndOfDIFData()]
       == 0xb1; }
00056
00057 Buffer SDHCAL_RawBuffer_Navigator::getSCBuffer()
00058 {
00059   setSCBuffer();
00060   return m_SCbuffer;
00061 }
00062
00063 bool SDHCAL_RawBuffer_Navigator::badSCData()
00064 {
00065   setSCBuffer();
00066   return m_BadSCdata;
00067 }
00068
00069 void SDHCAL_RawBuffer_Navigator::setSCBuffer()
00070 {
00071   if(!hasSlowControlData()) return;
00072   if(m_SCbuffer.size() != 0) return;  // deja fait
00073   if(m_BadSCdata) return;
00074   m_SCbuffer.set(&(getDIFBufferStart()[getEndOfDIFData()]));
00075   // compute Slow Control size
00076   std::size_t maxsize{m_Buffer.size() - m_DIFstartIndex - getEndOfDIFData() + 1};  // should I +1 here
       ?
00077   uint32_t    k{1};                                                                 // SC Header
00078   uint32_t    dif_ID{m_SCbuffer[1]};
00079   uint32_t    chipSize{m_SCbuffer[3]};
00080   while((dif_ID != 0xa1 && m_SCbuffer[k] != 0xa1 && k < maxsize) || (dif_ID == 0xa1 && m_SCbuffer[k +
       2] == chipSize && k < maxsize))
00081   {
00082     k += 2;  // DIF ID + ASIC Header
00083     uint32_t scsize = m_SCbuffer[k];
00084     if(scsize != 74 && scsize != 109)
00085     {
00086       std::cout « "PROBLEM WITH SC SIZE " « scsize « std::endl;
00087       k           = 0;
00088       m_BadSCdata = true;
00089       break;
00090     }
00091     k++;          // skip size bit
```

```
00092     k += scsize;  // skip the data
00093   }
00094   if(m_SCbuffer[k] == 0xa1 && !m_BadSCdata) m_SCbuffer.setSize(k + 1);  // add the trailer
00095   else
00096   {
00097     m_BadSCdata = true;
00098     std::cout « "PROBLEM SC TRAILER NOT FOUND " « std::endl;
00099   }
00100 }
00101
00102 Buffer SDHCAL_RawBuffer_Navigator::getEndOfAllData()
00103 {
00104   setSCBuffer();
00105   if(hasSlowControlData() && !m_BadSCdata) { return Buffer(&(m_SCbuffer.begin()[m_SCbuffer.size()]),
      getSizeAfterDIFPtr() - 3 - m_SCbuffer.size()); }
00106   else
00107     return Buffer(&(getDIFBufferStart()[getEndOfDIFData()]), getSizeAfterDIFPtr() - 3);  // remove the
      2 bytes for CRC and the DIF trailer
00108 }
```

5.39 /home/runner/work/streamout/streamout/libs/interface/Dump/include/textDump.h File Reference

```
#include "DIFPtr.h"
#include "Interface.h"
#include "spdlog/sinks/stdout_color_sinks.h"
#include <memory>
#include <ostream>
#include <spdlog/logger.h>
```

Classes

- class textDump

5.39.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file textDump.h.

5.40 textDump.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include "DIFPtr.h"
00008 #include "Interface.h"
00009 #include "spdlog/sinks/stdout_color_sinks.h"
00010
00011 #include <memory>
00012 #include <ostream>
00013 #include <spdlog/logger.h>
00014
00015 class textDump : public Interface
00016 {
00017 public:
```

```
00018   textDump()
00019   {
00020     m_InternalLogger = std::make_shared<spdlog::logger>("textDump",
       std::make_shared<spdlog::sinks::stdout_color_sink_mt>());
00021     m_InternalLogger->set_level(spdlog::level::trace);
00022   }
00023   void                              start();
00024   void                              processDIF(DIFPtr*);
00025   void                              processFrame(DIFPtr*, uint32_t frameIndex);
00026   void                              processPadInFrame(DIFPtr*, uint32_t frameIndex, uint32_t
       channelIndex);
00027   void                              processSlowControl(Buffer);
00028   void                              end();
00029   std::shared_ptr<spdlog::logger>& print() { return m_InternalLogger; }
00030   void                              setLevel(const spdlog::level::level_enum& level) {
       m_InternalLogger->set_level(level); }
00031
00032 private:
00033   // This class is a dumb class to print on terminal so we need the logger + the standard one given by
       the interface.
00034   std::shared_ptr<spdlog::logger> m_InternalLogger{nullptr};
00035 };
```

5.41 /home/runner/work/streamout/streamout/libs/interface/↩ Dump/src/textDump.cc File Reference

```
#include "textDump.h"
```

5.41.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file textDump.cc.

5.42 textDump.cc

Go to the documentation of this file.

```
00001
00005 #include "textDump.h"
00006
00007 void textDump::start() { print()->info("Will dump bunch of DIF data"); }
00008
00009 void textDump::processDIF(DIFPtr* d)
00010 {
00011   if(nullptr == d)
00012   {
00013     print()->info("DIFPtr is nullptr");
00014     return;
00015   }
00016   print()->info("DIF number is {}", d->getDIFid());
00017   print()->info("DTC value is {}", d->getDTC());
00018   print()->info("GTC value is {}", d->getGTC());
00019   print()->info("DIF BCID is {}", d->getBCID());
00020   print()->info("Absolute BCID is {}", d->getAbsoluteBCID());
00021   print()->info("The number of frame is {}", d->getNumberOfFrames());
00022 }
00023
00024 void textDump::processFrame(DIFPtr* d, uint32_t frameIndex)
00025 {
00026   print()->info("Displaying frame number {}", frameIndex);
00027   print()->info("ASIC ID is {}", d->getASICid(frameIndex));
00028   print()->info("Frame BCID is {}", d->getFrameBCID(frameIndex));
```

```
00029   print()->info("Frame Time To Trigger (a.k.a timestamp) is {}",
       d->getFrameTimeToTrigger(frameIndex));
00030 }
00031
00032 void textDump::processPadInFrame(DIFPtr* d, uint32_t frameIndex, uint32_t channelIndex)
00033 {
00034   if(d->getThresholdStatus(frameIndex, channelIndex) > 0)
00035   {
00036     print()->info("Displaying channel number {}", channelIndex);
00037     print()->info("Threshold status is {}", d->getThresholdStatus(frameIndex, channelIndex));
00038   }
00039 }
00040
00041 void textDump::processSlowControl(Buffer) { print()->error("textDump::processSlowControl not
       implemented yet."); }
00042
00043 void textDump::end() { print()->info("textDump end of report"); }
```

5.43 /home/runner/work/streamout/streamout/libs/interface/RawDataReader/include/RawdataReader.h File Reference

```
#include "Interface.h"
#include <array>
#include <cstdint>
#include <fstream>
#include <string>
#include <vector>
```

Classes

- class RawdataReader

5.43.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file RawdataReader.h.

5.44 RawdataReader.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include "Interface.h"
00008
00009 #include <array>
00010 #include <cstdint>
00011 #include <fstream>
00012 #include <string>
00013 #include <vector>
00014
00015 class Buffer;
00016
00017 class RawdataReader : public Interface
00018 {
00019 public:
```

```
00020   explicit RawdataReader(const char* fileName);
00021   void   start();
00022   void   end();
00023   float  getFileSize();
00024   void   openFile(const std::string& fileName);
00025   void   closeFile();
00026   bool   nextEvent();
00027   bool   nextDIFbuffer();
00028   Buffer getSDHCALBuffer();
00029   virtual ~RawdataReader() { closeFile(); }
00030   static void setDefaultBufferSize(const std::size_t& size);
00031
00032 private:
00033   void               uncompress();
00034   std::ifstream      m_FileStream;
00035   void               setFileSize(const std::size_t& size);
00036   static std::size_t m_BufferSize;
00037   std::size_t        m_FileSize{0};
00038   std::uint32_t      m_NumberOfDIF{0};
00039   std::uint32_t      m_EventNumber{0};
00040   std::vector<bit8_t> m_buf;
00041   Buffer             m_Buffer;
00042   std::string        m_Filename;
00043 };
```

5.45 /home/runner/work/streamout/streamout/libs/interface/RawData↩ Reader/src/RawdataReader.cc File Reference

```
#include "RawdataReader.h"
#include <cstring>
#include <stdexcept>
#include <zlib.h>
```

5.45.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file RawdataReader.cc.

5.46 RawdataReader.cc

Go to the documentation of this file.

```
00001
00004 #include "RawdataReader.h"
00005
00006 #include <cstring>
00007 #include <stdexcept>
00008 #include <zlib.h>
00009
00011 std::size_t RawdataReader::m_BufferSize = 0x100000;
00012
00013 void RawdataReader::setDefaultBufferSize(const std::size_t& size) { m_BufferSize = size; }
00014
00015 RawdataReader::RawdataReader(const char* fileName)
00016 {
00017   m_buf.reserve(m_BufferSize);
00018   m_Filename = fileName;
00019 }
00020
00021 void RawdataReader::start() { openFile(m_Filename); }
00022
00023 void RawdataReader::end() { closeFile(); }
```

```
00024
00025 void RawdataReader::uncompress()
00026 {
00027   static const std::size_t size_buffer{0x20000};
00028   std::size_t              shift{3 * sizeof(std::uint32_t) + sizeof(std::uint64_t)};
00029   static bit8_t            obuf[size_buffer];
00030   std::size_t              size_buffer_end{0x20000};
00031   int                      rc = ::uncompress(obuf, &size_buffer_end, &m_Buffer[shift], m_Buffer.size()
      - shift);
00032   switch(rc)
00033   {
00034     case Z_OK: break;
00035     default: throw "decompress error"; break;
00036   }
00037   memcpy(&m_Buffer[shift], obuf, size_buffer_end);
00038   m_Buffer.setSize(size_buffer_end + shift);
00039 }
00040
00041 void RawdataReader::closeFile()
00042 {
00043   try
00044   {
00045     if(m_FileStream.is_open()) m_FileStream.close();
00046   }
00047   catch(const std::ios_base::failure& e)
00048   {
00049     log()->error("Caught an ios_base::failure in closeFile : {} {}", e.what(), e.code().value());
00050     throw;
00051   }
00052 }
00053
00054 void RawdataReader::openFile(const std::string& fileName)
00055 {
00056   try
00057   {
00058     m_FileStream.rdbuf()->pubsetbuf(0, 0);
00059     m_FileStream.exceptions(std::ifstream::failbit | std::ifstream::badbit);
00060     m_FileStream.open(fileName.c_str(), std::ios::in | std::ios::binary | std::ios::ate);  // Start at
      the end to directly calculate the size of the file then come back to beginning
00061     m_FileStream.rdbuf()->pubsetbuf(0, 0);
00062     if(m_FileStream.is_open())
00063     {
00064       setFileSize(m_FileStream.tellg());
00065       m_FileStream.seekg(0, std::ios::beg);
00066     }
00067   }
00068   catch(const std::ios_base::failure& e)
00069   {
00070     log()->error("Caught an ios_base::failure in openFile : {} {}", e.what(), e.code().value());
00071     throw;
00072   }
00073 }
00074
00075 bool RawdataReader::nextEvent()
00076 {
00077   try
00078   {
00079     m_FileStream.read(reinterpret_cast<char*>(&m_EventNumber), sizeof(std::uint32_t));
00080     m_FileStream.read(reinterpret_cast<char*>(&m_NumberOfDIF), sizeof(std::uint32_t));
00081   }
00082   catch(const std::ios_base::failure& e)
00083   {
00084     return false;
00085   }
00086   return true;
00087 }
00088
00089 bool RawdataReader::nextDIFbuffer()
00090 {
00091   try
00092   {
00093     static int DIF_processed{0};
00094     if(DIF_processed >= m_NumberOfDIF)
00095     {
00096       DIF_processed = 0;
00097       return false;
00098     }
00099     else
00100     {
00101       DIF_processed++;
00102       std::uint32_t bsize{0};
00103       m_FileStream.read(reinterpret_cast<char*>(&bsize), sizeof(std::uint32_t));
00104       m_FileStream.read(reinterpret_cast<char*>(&m_buf[0]), bsize);
00105       m_Buffer = Buffer(m_buf);
00106     }
00107   }
00108   catch(const std::ios_base::failure& e)
```

```
00109   {
00110     return false;
00111   }
00112   return true;
00113 }
00114
00115 Buffer RawdataReader::getSDHCALBuffer()
00116 {
00117   uncompress();
00118   return m_Buffer;
00119 }
00120
00121 void RawdataReader::setFileSize(const std::size_t& size) { m_FileSize = size; }
00122
00123 float RawdataReader::getFileSize() { return m_FileSize; }
```

5.47 /home/runner/work/streamout/streamout/libs/interface/↩ ROOT/include/ROOTtreeDest.h File Reference

```
#include "Buffer.h"
#include "DIFPtr.h"
#include "Interface.h"
#include "TTree.h"
```

Classes

- class ROOTtreeDest
- struct ROOTtreeDest::DATA

5.47.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file ROOTtreeDest.h.

5.48 ROOTtreeDest.h

Go to the documentation of this file.

```
00001
00006 #pragma once
00007
00008 #include "Buffer.h"
00009 #include "DIFPtr.h"
00010 #include "Interface.h"
00011 #include "TTree.h"
00012
00013 class ROOTtreeDest : public Interface
00014 {
00015 public:
00016   typedef struct
00017   {
00018     UInt_t    DIFid, ASICid, CHANNELid;
00019     UInt_t    Thresh;
00020     UInt_t    DTC, GTC, DIF_BCID, frame_BCID, timeStamp;
00021     ULong64_t AbsoluteBCID;
00022   } DATA;
00023
```

```
00024   ROOTtreeDest();
00025
00026   void start();
00027   void processDIF(DIFPtr*);
00028   void processFrame(DIFPtr*, std::uint32_t frameIndex);
00029   void processPadInFrame(DIFPtr*, std::uint32_t frameIndex, std::uint32_t channelIndex);
00030   void processSlowControl(const Buffer&) { ; }
00031   void end() { ; }
00032
00033 private:
00034   DATA   _data;
00035   TTree* _tree;
00036   void   dataReset();
00037 };
```

5.49 /home/runner/work/streamout/streamout/libs/interface/ROOT/src/↩ ROOTtreeDest.cc File Reference

```
#include "ROOTtreeDest.h"
```

5.49.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file ROOTtreeDest.cc.

5.50 ROOTtreeDest.cc

Go to the documentation of this file.

```
00001
00006 #include "ROOTtreeDest.h"
00007
00008 ROOTtreeDest::ROOTtreeDest()
00009 {
00010   dataReset();
00011   _tree = new TTree("RawData", "Raw SDHCAL data tree");
00012   _tree->Branch("data", &_data,
       "DIFid/i:ASICid:CHANNELid:Thresh:DTC:GTC:DIF_BCID:frame_BCID:timeStamp:AbsoluteBCID/l");
00013 }
00014
00015 void ROOTtreeDest::dataReset()
00016 {
00017   _data.DIFid = _data.ASICid = _data.CHANNELid = 0;
00018   _data.Thresh                                 = 0;
00019   _data.DTC = _data.GTC = _data.DIF_BCID = _data.frame_BCID = _data.timeStamp = 0;
00020   _data.AbsoluteBCID                                                          = 0;
00021 }
00022
00023 void ROOTtreeDest::start() { dataReset(); }
00024
00025 void ROOTtreeDest::processDIF(DIFPtr* d)
00026 {
00027   _data.DIFid        = d->getDIFid();
00028   _data.DTC          = d->getDTC();
00029   _data.GTC          = d->getGTC();
00030   _data.DIF_BCID     = d->getBCID();
00031   _data.AbsoluteBCID = d->getAbsoluteBCID();
00032 }
00033
00034 void ROOTtreeDest::processFrame(DIFPtr* d, std::uint32_t frameIndex)
00035 {
00036   _data.ASICid     = d->getASICid(frameIndex);
00037   _data.frame_BCID = d->getFrameBCID(frameIndex);
```

```
00038   _data.timeStamp  = d->getFrameTimeToTrigger(frameIndex);
00039 }
00040
00041 void ROOTtreeDest::processPadInFrame(DIFPtr* d, std::uint32_t frameIndex, std::uint32_t channelIndex)
00042 {
00043   _data.CHANNELid = channelIndex;
00044   _data.Thresh    = d->getThresholdStatus(frameIndex, channelIndex);
00045   if(_data.Thresh != 0) _tree->Fill();
00046 }
```