streamout

Generated by Doxygen 1.9.3

1 Hierarchical Index

1.1 Class Hierarchy

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

2 Class Index

2.1 Class List

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

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

4 Class Documentation

4.1 Buffer Class Reference

`#include <libs/core/include/Buffer.h>`

Public Member Functions

- Buffer ()
- virtual ~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 () const
- bit8_t * end () const
- bit8_t & operator[ ] (const std::size_t &pos)
- bit8_t & operator[ ] (const std::size_t &pos) const
- void setSize (const std::size_t &size)

4.1.1 Detailed Description

Definition at line 14 of file Buffer.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Buffer() [1/5] Buffer::Buffer ( ) [inline]

Definition at line 17 of file Buffer.h.

```
00017 :  m_Buffer(nullptr), m_Size(0), m_Capacity(0) {}
```

4.1.2.2 ~Buffer() virtual Buffer::~Buffer ( ) [inline], [virtual]

Definition at line 18 of file Buffer.h.

```
00018 {}
```

4.1.2.3 Buffer() [2/5] Buffer::Buffer (const bit8_t b[], const std::size_t & i) [inline]

Definition at line 19 of file Buffer.h.

```
00019 :  m_Buffer(const_cast<bit8_t*>(&b[0])), m_Size(i), m_Capacity(i) {}
```

4.1.2.4 Buffer() [3/5] Buffer::Buffer (const char b[], const std::size_t & i) [inline]

Definition at line 20 of file Buffer.h.

```
00020 :  m_Buffer(const_cast<bit8_t*>(reinterpret_cast<const bit8_t*>(&b[0]))), m_Size(i * sizeof(char)),
      m_Capacity(i * sizeof(char)) {}
```

4.1.2.5 Buffer() [4/5] template<typename T >

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

Definition at line 21 of file Buffer.h.

```
00021 :  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)) {}
```

4.1.2.6 Buffer() [5/5] template<typename T , std::size_t N>

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

Definition at line 22 of file Buffer.h.

```
00022 :  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)) {}
```

4.1.3 Member Function Documentation

4.1.3.1 begin() bit8_t * Buffer::begin ( ) const [inline]

Definition at line 28 of file Buffer.h.

```
00028 { return m_Buffer; }
```

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

Definition at line 25 of file Buffer.h.

```
00025 { return m_Capacity; }
```

4.1.3.3 end() bit8_t * Buffer::end ( ) const [inline]

Definition at line 29 of file Buffer.h.

```
00029 { 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 30 of file Buffer.h.

```
00030 { 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 31 of file Buffer.h.

```
00031 { return m_Buffer[pos]; }
```

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

Definition at line 27 of file Buffer.h.

```
00027 { m_Buffer = b; }
```

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

Definition at line 33 of file Buffer.h.

```
00033 { m_Size = size; }
```

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

Definition at line 24 of file Buffer.h.

```
00024 { return m_Size; }
```

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

- libs/core/include/Buffer.h

4.2 BufferLooper< SOURCE, DESTINATION > Class Template Reference

`#include <libs/core/include/BufferLooper.h>`

Public Member Functions

- BufferLooper (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::uint32_t &m_NbrEventsToProcess=0)
- void printAllCounters ()
- std::shared_ptr< spdlog::logger > log ()
- void setDetectorIDs (const std::vector< DetectorID > &detectorIDs)

4.2.1 Detailed Description

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

Definition at line 27 of file BufferLooper.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 BufferLooper() template<typename SOURCE , typename DESTINATION >

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

Definition at line 30 of file BufferLooper.h.

```
00030                                                                   :  m_Source(source),
      m_Destination(dest), m_Debug(debug)
00031   {
00032     m_Logger = spdlog::create<spdlog::sinks::null_sink_mt>("streamout");
00033     if(!spdlog::get("streamout")) { spdlog::register_logger(m_Logger); }
00034     m_Source.setLogger(m_Logger);
00035     m_Destination.setLogger(m_Logger);
00036   }
```

4.2.3 Member Function Documentation

4.2.3.1 addSink() template<typename SOURCE , typename DESTINATION >

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

Definition at line 38 of file BufferLooper.h.

```
00039   {
00040     sink->set_level(level);
00041     m_Sinks.push_back(sink);
00042     m_Logger = std::make_shared<spdlog::logger>("streamout", begin(m_Sinks), end(m_Sinks));
00043     m_Source.setLogger(m_Logger);
00044     m_Destination.setLogger(m_Logger);
00045   }
```

4.2.3.2 log() template<typename SOURCE , typename DESTINATION >

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

Definition at line 203 of file BufferLooper.h.

```
00203 { return m_Logger; }
```

4.2.3.3 loop() template<typename SOURCE , typename DESTINATION >

```
void BufferLooper< SOURCE, DESTINATION >::loop (
            const std::uint32_t & m_NbrEventsToProcess = 0 )  [inline]
```

START EVENT ///

START DIF ///

START FRAME ///

START FRAME ///

START DIF ///

START EVENT ///

Definition at line 47 of file BufferLooper.h.

```
00048   {
00049     // clang-format off
00050     fmt::print(fg(fmt::color::medium_orchid) | fmt::emphasis::bold,
00051                "\n"
00052 " SSSSSSSSSSSSSSS      tttt
      tttt\n"
00053 "SS:::::::::::::::S  ttt:::t
      ttt:::t\n"
00054 "S:::::SSSSSS::::::S  t:::::t
      t:::::t\n"
00055 "S:::::S     SSSSSSS  t:::::t
      t:::::t\n"
00056 "S:::::S        ttttttt:::::ttttttt    rrrrr   rrrrrrrrr       eeeeeeeeeeee    aaaaaaaaaaaaa
      mmmmmmm    mmmmmmm      ooooooooooo   uuuuuu    uuuuuuttttttt:::::ttttttt\n"
00057 "S:::::S        t:::::::::::::::::t   r::::rrr:::::::::r    ee::::::::::::ee  a::::::::::::a
      mm:::::::m  m:::::::mm  oo:::::::::::oo u::::u    u::::ut:::::::::::::::::t\n"
00058 " S::::SSSS     t:::::::::::::::::t   r:::::::::::::::::r  e::::::eeeee:::::eeaaaaaaaaa:::::a
      m::::::::::mm::::::::::mo:::::::::::::::ou::::u    u::::ut:::::::::::::::::t\n"
00059 "  SS::::::SSSSStttttt:::::::tttttt   rr::::::rrrrr::::::re::::::e     e:::::e         a::::a
      m::::::::::::::::::::::mo::::ooooo::::ou::::u    u::::utttttt:::::::tttttt\n"
00060 "    SSS::::::::SS    t:::::t          r:::::r     r:::::re:::::::eeeee::::::e  aaaaaaa:::::a
      m::::mmm::::::mmm:::::mo::::o     o::::ou::::u    u::::u      t:::::t\n"
00061 "       SSSSSS::::S   t:::::t          r:::::r     rrrrrrre:::::::::::::::::e aa::::::::::::a m::::m
      m::::m   m::::mo::::o     o::::ou::::u    u::::u      t:::::t\n"
00062 "            S:::::S  t:::::t          r:::::r          e::::::eeeeeeeeeee a::::aaaa::::::a m::::m
      m::::m   m::::mo::::o     o::::ou::::u    u::::u      t:::::t\n"
00063 "            S:::::S  t:::::t    ttttttr:::::r          e:::::::e         a::::a    a:::::a m::::m
      m::::m   m::::mo::::o     o::::ou::::uuuu:::::u      t:::::t    ttttttt\n"
00064 "SSSSSSS     S:::::S  t:::::::ttttt:::::tr:::::r          e::::::::e        a::::a    a:::::a m::::m
      m::::m   m::::mo::::ooooo::::ou:::::::::::::::uu    t:::::::ttttt:::::t\n"
00065 "S::::::SSSSSS:::::S  tt:::::::::::::::tr:::::r           e::::::::eeeeeeeea:::::aaaa::::::a m::::m
      m::::m   m::::mo:::::::::::::::o u:::::::::::::::u    tt:::::::::::::::t\n"
00066 "S:::::::::::::::SS     tt:::::::::::::ttr:::::r            ee::::::::::::e a::::::::::aa:::am::::m
      m::::m   m::::m oo:::::::::::oo   uu::::::::uu:::u      tt:::::::::::tt\n"
00067 " SSSSSSSSSSSSSSS        ttttttttttt  rrrrrrr                eeeeeeeeeeeeee  aaaaaaaaaa  aaaammmmmm
      mmmmmm   mmmmmm   oooooooooo       uuuuuuuu  uuuu        ttttttttttt {}\n"
00068 "\n",
00069 fmt::format(fg(fmt::color::red) | fmt::emphasis::bold, "v{}", streamout_version.to_string()));
00070     // clang-format on
00071     log()->info("*********************************************************************");
00072     log()->info("Streamout Version :  {}", streamout_version.to_string());
00073     log()->info("Using InterfaceReader {} version {}", m_Source.getName(),
      m_Source.getVersion().to_string());
00074     log()->info("Using InterfaceWriter {} version {}", m_Destination.getName(),
      m_Destination.getVersion().to_string());
00075
00076     if(!m_Destination.checkCompatibility(m_Source.getName(), m_Source.getVersion().to_string()))
00077     {
00078       log()->critical("{} version {} is not compatible with {} version {} !  ", m_Source.getName(),
      m_Source.getVersion().to_string(), m_Destination.getName(), m_Destination.getVersion().to_string());
00079       log()->info("Compatible Interfaces for {} are", m_Destination.getName());
00080       for(std::map<std::string, std::string>::iterator it = m_Destination.getCompatibility().begin();
      it != m_Destination.getCompatibility().end(); ++it) { log()->info("{} version {}", it->first,
      it->second); }
00081       std::exit(-1);
00082     }
00083     if(!m_DetectorIDs.empty())
00084     {
00085       std::string ids;
00086       for(std::vector<DetectorID>::const_iterator it = m_DetectorIDs.cbegin(); it !=
      m_DetectorIDs.cend(); ++it) ids += std::to_string(static_cast<std::uint16_t>(*it)) + ";";
00087       log()->info("Detector ID(s) other than {} will be ignored", ids);
00088     }
```

```
00089     log()->info("**********************************************************************");
00090     RawBufferNavigator bufferNavigator(m_Logger);
00091     Timer              timer;
00092     timer.start();
00093     m_Source.start();
00094     m_Destination.start();
00095     while(m_Source.nextEvent() && m_NbrEventsToProcess >= m_NbrEvents)
00096     {
00098       m_Source.startEvent();
00099       m_Destination.startEvent();
00101
00102       m_Logger->warn("===*** Event {} ***===", m_NbrEvents);
00103       while(m_Source.nextDIFbuffer())
00104       {
00105         const Buffer& buffer = m_Source.getBuffer();
00106         bufferNavigator.setBuffer(buffer);
00107         if(std::find(m_DetectorIDs.begin(), m_DetectorIDs.end(),
      static_cast<DetectorID>(bufferNavigator.getDetectorID())) == m_DetectorIDs.end())
00108         {
00109           m_Logger->debug("Ignoring detector ID : {}", bufferNavigator.getDetectorID());
00110           continue;
00111         }
00112
00113         bit8_t* debug_variable_1 = buffer.end();
00114         bit8_t* debug_variable_2 = bufferNavigator.getDIFBuffer().end();
00115         if(debug_variable_1 != debug_variable_2) m_Logger->info("DIF BUFFER END {} {}",
      fmt::ptr(debug_variable_1), fmt::ptr(debug_variable_2));
00116         if(m_Debug) assert(debug_variable_1 == debug_variable_2);
00117
00118         std::int32_t idstart = bufferNavigator.getStartOfPayload();
00119         if(m_Debug && idstart == -1) m_Logger->info(to_hex(buffer));
00120         c.DIFStarter[idstart]++;
00121         if(!bufferNavigator.validBuffer())
00122         {
00123           m_Logger->error("!bufferNavigator.validBuffer()");
00124           continue;
00125         }
00126
00128         m_Source.startDIF();
00129         m_Destination.startDIF();
00131
00132         DIFPtr& d = bufferNavigator.getDIFPtr();
00133         c.DIFPtrValueAtReturnedPos[bufferNavigator.getDIFBufferStart()[d.getGetFramePtrReturn()]]++;
00134         if(m_Debug) assert(bufferNavigator.getDIFBufferStart()[d.getGetFramePtrReturn()] == 0xa0);
00135         c.SizeAfterDIFPtr[bufferNavigator.getSizeAfterDIFPtr()]++;
00136         m_Destination.processDIF(d);
00137         for(std::size_t i = 0; i < d.getNumberOfFrames(); ++i)
00138         {
00140           m_Source.startFrame();
00141           m_Destination.startFrame();
00143           m_Destination.processFrame(d, i);
00144           for(std::size_t j = 0; j < DU::NUMBER_PAD; ++j)
00145           {
00146             if(d.getThresholdStatus(i, j) != 0)
00147             {
00148               m_Source.startPad();
00149               m_Destination.startPad();
00150               m_Destination.processPadInFrame(d, i, j);
00151               m_Source.endPad();
00152               m_Destination.endPad();
00153             }
00154           }
00156           m_Source.endFrame();
00157           m_Destination.endFrame();
00159         }
00160
00161         bool processSC = false;
00162         if(bufferNavigator.hasSlowControlData())
00163         {
00164           c.hasSlowControl++;
00165           processSC = true;
00166         }
00167         if(bufferNavigator.badSCData())
00168         {
00169           c.hasBadSlowControl++;
00170           processSC = false;
00171         }
00172         if(processSC) { m_Destination.processSlowControl(bufferNavigator.getSCBuffer()); }
00173
00174         Buffer eod = bufferNavigator.getEndOfAllData();
00175         c.SizeAfterAllData[eod.size()]++;
00176         bit8_t* debug_variable_3 = eod.end();
00177         if(debug_variable_1 != debug_variable_3) m_Logger->info("END DATA BUFFER END {} {}",
      fmt::ptr(debug_variable_1), fmt::ptr(debug_variable_3));
00178         if(m_Debug) assert(debug_variable_1 == debug_variable_3);
00179         if(eod.size() != 0) m_Logger->info("End of Data remaining stuff :  {}", to_hex(eod));
00180
```

```
00181         int nonzeroCount = 0;
00182         for(bit8_t* it = eod.begin(); it != eod.end(); it++)
00183           if(static_cast<int>(*it) != 0) nonzeroCount++;
00184         c.NonZeroValusAtEndOfData[nonzeroCount]++;
00186         m_Source.endDIF();
00187         m_Destination.endDIF();
00189       }  // end of DIF while loop
00190       m_Logger->warn("===*** Event {} ***===", m_NbrEvents);
00191       m_NbrEvents++;
00193       m_Source.endEvent();
00194       m_Destination.endEvent();
00196     }  // end of event while loop
00197     m_Destination.end();
00198     m_Source.end();
00199     timer.stop();
00200     fmt::print("=== elapsed time {}ms ({}ms/event) ===\n", timer.getElapsedTime() / 1000,
      timer.getElapsedTime() / (1000 * m_NbrEvents));
00201   }
```

4.2.3.4 printAllCounters() template<typename SOURCE , typename DESTINATION >

void BufferLooper< SOURCE, DESTINATION >::printAllCounters ( ) [inline]

Definition at line 202 of file BufferLooper.h.

```
00202 { c.printAllCounters(); }
```

4.2.3.5 setDetectorIDs() template<typename SOURCE , typename DESTINATION >

void BufferLooper< SOURCE, DESTINATION >::setDetectorIDs (
const std::vector< DetectorID > & *detectorIDs*) [inline]

Definition at line 205 of file BufferLooper.h.

```
00205 { m_DetectorIDs = detectorIDs; }
```

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

- libs/core/include/BufferLooper.h

4.3 BufferLooperCounter Struct Reference

#include <libs/core/include/BufferLooperCounter.h>

Public Member Functions

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

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

4.3.1 Detailed Description

Definition at line 11 of file BufferLooperCounter.h.

4.3.2 Member Function Documentation

4.3.2.1 printAllCounters() `void BufferLooperCounter::printAllCounters ( )`

Definition at line 9 of file BufferLooperCounter.cc.

```
00010 {
00011   fmt::print("BUFFER LOOP FINAL STATISTICS : \n");
00012   printCounter("Start of DIF header", DIFStarter);
00013   printCounter("Value after DIF data are processed", DIFPtrValueAtReturnedPos);
00014   printCounter("Size remaining in buffer after end of DIF data", SizeAfterDIFPtr);
00015   fmt::print("Number of Slow Control found {}  out of which {} are bad\n", hasSlowControl,
      hasBadSlowControl);
00016   printCounter("Size remaining after all of data have been processed", SizeAfterAllData);
00017   printCounter("Number on non zero values in end of data buffer", NonZeroValusAtEndOfData);
00018 }
```

4.3.2.2 printCounter() `void BufferLooperCounter::printCounter (`

`const std::string & description,`

`const std::map< int, int > & m )`

Definition at line 20 of file BufferLooperCounter.cc.

```
00021 {
00022   std::string out{"statistics for " + description + " :  \n"};
00023   for(std::map<int, int>::const_iterator it = m.begin(); it != m.end(); it++)
00024   {
00025     if(it != m.begin()) out += ",";
00026     out += " [" + std::to_string(it->first) + "]=" + std::to_string(it->second);
00027   }
00028   out += "\n";
00029   fmt::print(out);
00030 }
```

4.3.3 Member Data Documentation

4.3.3.1 DIFPtrValueAtReturnedPos `std::map<int, int> BufferLooperCounter::DIFPtrValueAt↩ ReturnedPos`

Definition at line 17 of file BufferLooperCounter.h.

4.3.3.2 DIFStarter `std::map<int, int> BufferLooperCounter::DIFStarter`

Definition at line 16 of file BufferLooperCounter.h.

4.3.3.3 hasBadSlowControl `int BufferLooperCounter::hasBadSlowControl = 0`

Definition at line 15 of file BufferLooperCounter.h.

4.3.3.4 hasSlowControl `int BufferLooperCounter::hasSlowControl = 0`

Definition at line 14 of file BufferLooperCounter.h.

4.3.3.5 NonZeroValusAtEndOfData `std::map<int, int> BufferLooperCounter::NonZeroValusAtEnd↩OfData`

Definition at line 20 of file BufferLooperCounter.h.

4.3.3.6 SizeAfterAllData `std::map<int, int> BufferLooperCounter::SizeAfterAllData`

Definition at line 19 of file BufferLooperCounter.h.

4.3.3.7 SizeAfterDIFPtr `std::map<int, int> BufferLooperCounter::SizeAfterDIFPtr`

Definition at line 18 of file BufferLooperCounter.h.

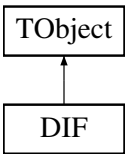
The documentation for this struct was generated from the following files:

- libs/core/include/BufferLooperCounter.h
- libs/core/src/BufferLooperCounter.cc

4.4 DIF Class Reference

`#include <libs/interface/ROOT/include/DIF.h>`

Inheritance diagram for DIF:



Public Member Functions

- void clear ()
- void addHit (const Hit &)
- void setID (const std::uint8_t &)
- std::uint8_t getID () const
- void setDTC (const std::uint32_t &)
- std::uint32_t getDTC () const
- void setGTC (const std::uint32_t &)
- std::uint32_t getGTC () const
- void setDIFBCID (const std::uint32_t &)
- std::uint32_t getDIFBCID () const
- void setAbsoluteBCID (const std::uint64_t &)
- std::uint64_t getAbsoluteBCID () const
- std::vector< Hit >::const_iterator cbegin () const
- std::vector< Hit >::const_iterator cend () const

4.4.1 Detailed Description

Definition at line 16 of file DIF.h.

4.4.2 Member Function Documentation

4.4.2.1 addHit() `void DIF::addHit ( const Hit & hit )`

Definition at line 10 of file DIF.cc.

```
00010 { m_Hits.push_back(hit); }
```

4.4.2.2 cbegin() `std::vector< Hit >::const_iterator DIF::cbegin ( ) const`

Definition at line 32 of file DIF.cc.

```
00032 { return m_Hits.cbegin(); }
```

4.4.2.3 cend() `std::vector< Hit >::const_iterator DIF::cend ( ) const`

Definition at line 34 of file DIF.cc.

```
00034 { return m_Hits.cend(); }
```

4.4.2.4 clear() `void DIF::clear ( )`

Definition at line 36 of file DIF.cc.

```
00036 { m_Hits.clear(); }
```

4.4.2.5 getAbsoluteBCID() `std::uint64_t DIF::getAbsoluteBCID ( ) const`

Definition at line 30 of file DIF.cc.

```
00030 { return m_AbsoluteBCID; }
```

4.4.2.6 getDIFBCID() `std::uint32_t DIF::getDIFBCID ( ) const`

Definition at line 26 of file DIF.cc.

```
00026 { return m_DIFBCID; }
```

4.4.2.7 getDTC() `std::uint32_t DIF::getDTC ( ) const`

Definition at line 18 of file DIF.cc.

```
00018 { return m_DTC; }
```

4.4.2.8 getGTC() `std::uint32_t DIF::getGTC ( ) const`

Definition at line 22 of file DIF.cc.

```
00022 { return m_GTC; }
```

4.4.2.9 getID() `std::uint8_t DIF::getID ( ) const`

Definition at line 14 of file DIF.cc.

```
00014 { return m_ID; }
```

4.4.2.10 setAbsoluteBCID() `void DIF::setAbsoluteBCID ( const std::uint64_t & absolutebcid )`

Definition at line 28 of file DIF.cc.

```
00028 { m_AbsoluteBCID = absolutebcid; }
```

4.4.2.11 setDIFBCID() `void DIF::setDIFBCID (`

```
const std::uint32_t & difbcid )
```

Definition at line 24 of file DIF.cc.

```
00024 { m_DIFBCID = difbcid; }
```

4.4.2.12 setDTC() `void DIF::setDTC (`

```
const std::uint32_t & dtc )
```

Definition at line 16 of file DIF.cc.

```
00016 { m_DTC = dtc; }
```

4.4.2.13 setGTC() `void DIF::setGTC (`

```
const std::uint32_t & gtc )
```

Definition at line 20 of file DIF.cc.

```
00020 { m_GTC = gtc; }
```

4.4.2.14 setID() `void DIF::setID (`

```
const std::uint8_t & id )
```

Definition at line 12 of file DIF.cc.

```
00012 { m_ID = id; }
```

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

- libs/interface/ROOT/include/DIF.h
- libs/interface/ROOT/src/DIF.cc

4.5 DIFPtr Class Reference

`#include <libs/core/include/DIFPtr.h>`

Public Member Functions

- void setBuffer (unsigned char *, const std::uint32_t &)
- bit8_t * getPtr () const
- std::uint32_t getGetFramePtrReturn () const
- std::vector< bit8_t * > & getFramesVector ()
- std::vector< bit8_t * > & getLinesVector ()
- std::uint32_t getID () const
- std::uint32_t getDTC () const
- std::uint32_t getGTC () const
- std::uint64_t getAbsoluteBCID () const
- std::uint32_t getBCID () const
- std::uint32_t getLines () const
- bool hasLine (const std::uint32_t &) const
- std::uint32_t getTASU1 () const
- std::uint32_t getTASU2 () const
- std::uint32_t getTDIF () const
- float getTemperatureDIF () const
- float getTemperatureASU1 () const
- float getTemperatureASU2 () const
- bool hasTemperature () const
- bool hasAnalogReadout () const
- std::uint32_t getNumberOfFrames () const
- bit8_t * getFramePtr (const std::uint32_t &) const
- std::uint32_t getFrameAsicHeader (const std::uint32_t &) const
- std::uint32_t getFrameBCID (const std::uint32_t &) const
- std::uint32_t getFrameTimeToTrigger (const std::uint32_t &) const
- bool getFrameLevel (const std::uint32_t &, const std::uint32_t &, const std::uint32_t &) const
- std::uint32_t getDIFid () const
- std::uint32_t getASICid (const std::uint32_t &) const
- std::uint32_t getThresholdStatus (const std::uint32_t &, const std::uint32_t &) const

4.5.1 Detailed Description

Definition at line 18 of file DIFPtr.h.

4.5.2 Member Function Documentation

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

Definition at line 95 of file DIFPtr.h.

```
00096 {
00097   std::uint64_t LBC = ((theDIF_[DU::ABCID_SHIFT] << 16) | (theDIF_[DU::ABCID_SHIFT + 1] << 8) |
      (theDIF_[DU::ABCID_SHIFT + 2])) * 16777216ULL /* to shift the value from the 24 first bits*/
00098                     + ((theDIF_[DU::ABCID_SHIFT + 3] << 16) | (theDIF_[DU::ABCID_SHIFT + 4] << 8) |
      (theDIF_[DU::ABCID_SHIFT + 5]));
00099   return LBC;
00100 }
```

4.5.2.2 getASICid() uint32_t DIFPtr::getASICid (const std::uint32_t & *i*) const [inline]

Definition at line 141 of file DIFPtr.h.

```
00141 { return getFrameAsicHeader(i) & 0xFF; }
```

4.5.2.3 getBCID() std::uint32_t DIFPtr::getBCID ( ) const [inline]

Definition at line 102 of file DIFPtr.h.

```
00102 { return (theDIF_[DU::BCID_SHIFT] << 16) + (theDIF_[DU::BCID_SHIFT + 1] << 8) + theDIF_[DU::BCID_SHIFT +
      2]; }
```

4.5.2.4 getDIFid() uint32_t DIFPtr::getDIFid ( ) const [inline]

Definition at line 139 of file DIFPtr.h.

```
00139 { return getID() & 0xFF; }
```

4.5.2.5 getDTC() std::uint32_t DIFPtr::getDTC ( ) const [inline]

Definition at line 91 of file DIFPtr.h.

```
00091 { return (theDIF_[DU::DTC_SHIFT] << 24) + (theDIF_[DU::DTC_SHIFT + 1] << 16) + (theDIF_[DU::DTC_SHIFT +
      2] << 8) + theDIF_[DU::DTC_SHIFT + 3]; }
```

4.5.2.6 getFrameAsicHeader() std::uint32_t DIFPtr::getFrameAsicHeader (const std::uint32_t & *i*) const [inline]

Definition at line 128 of file DIFPtr.h.

```
00128 { return getFrameAsicHeaderInternal(theFrames_[i]); }
```

4.5.2.7 getFrameBCID() std::uint32_t DIFPtr::getFrameBCID (const std::uint32_t & *i*) const [inline]

Definition at line 130 of file DIFPtr.h.

```
00130 { return GrayToBin((theFrames_[i][DU::FRAME_BCID_SHIFT] << 16) + (theFrames_[i][DU::FRAME_BCID_SHIFT +
      1] << 8) + theFrames_[i][DU::FRAME_BCID_SHIFT + 2]); }
```

4.5.2.8 getFrameLevel() bool DIFPtr::getFrameLevel (const std::uint32_t & *i*, const std::uint32_t & *ipad*, const std::uint32_t & *ilevel*) const [inline]

Definition at line 134 of file DIFPtr.h.

```
00135 {
00136   return ((theFrames_[i][DU::FRAME_DATA_SHIFT + ((3 - ipad / 16) * 4 + (ipad % 16) / 4)] >> (7 -
      (((ipad % 16) % 4) * 2 + ilevel))) & 0x1);
00137 }
```

4.5.2.9 getFramePtr()

```
bit8_t * DIFPtr::getFramePtr (
        const std::uint32_t & i ) const  [inline]
```

Definition at line 126 of file DIFPtr.h.

```
00126 { return theFrames_[i]; }
```

4.5.2.10 getFramesVector()

```
std::vector< bit8_t * > & DIFPtr::getFramesVector ( )  [inline]
```

Definition at line 85 of file DIFPtr.h.

```
00085 { return theFrames_; }
```

4.5.2.11 getFrameTimeToTrigger()

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

Definition at line 132 of file DIFPtr.h.

```
00132 { return getBCID() - getFrameBCID(i); }
```

4.5.2.12 getGetFramePtrReturn()

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

Definition at line 83 of file DIFPtr.h.

```
00083 { return theGetFramePtrReturn_; }
```

4.5.2.13 getGTC()

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

Definition at line 93 of file DIFPtr.h.

```
00093 { return (theDIF_[DU::GTC_SHIFT] << 24) + (theDIF_[DU::GTC_SHIFT + 1] << 16) + (theDIF_[DU::GTC_SHIFT +
      2] << 8) + theDIF_[DU::GTC_SHIFT + 3]; }
```

4.5.2.14 getID()

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

Definition at line 89 of file DIFPtr.h.

```
00089 { return theDIF_[DU::ID_SHIFT]; }
```

4.5.2.15 getLines()

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

Definition at line 104 of file DIFPtr.h.

```
00104 { return (theDIF_[DU::LINES_SHIFT] >> 4) & 0x5; }
```

4.5.2.16 getLinesVector() `std::vector< bit8_t * > & DIFPtr::getLinesVector ( ) [inline]`

Definition at line 87 of file DIFPtr.h.

```
00087 { return theLines_; }
```

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

Definition at line 124 of file DIFPtr.h.

```
00124 { return theFrames_.size(); }
```

4.5.2.18 getPtr() `bit8_t * DIFPtr::getPtr ( ) const [inline]`

Definition at line 81 of file DIFPtr.h.

```
00081 { return theDIF_; }
```

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

Definition at line 108 of file DIFPtr.h.

```
00108 { return (theDIF_[DU::TASU1_SHIFT] << 24) + (theDIF_[DU::TASU1_SHIFT + 1] << 16) +
      (theDIF_[DU::TASU1_SHIFT + 2] << 8) + theDIF_[DU::TASU1_SHIFT + 3]; }
```

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

Definition at line 110 of file DIFPtr.h.

```
00110 { return (theDIF_[DU::TASU2_SHIFT] << 24) + (theDIF_[DU::TASU2_SHIFT + 1] << 16) +
      (theDIF_[DU::TASU2_SHIFT + 2] << 8) + theDIF_[DU::TASU2_SHIFT + 3]; }
```

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

Definition at line 112 of file DIFPtr.h.

```
00112 { return theDIF_[DU::TDIF_SHIFT]; }
```

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

Definition at line 116 of file DIFPtr.h.

```
00116 { return (getTASU1() >> 3) * 0.0625; }
```

4.5.2.23 getTemperatureASU2() float DIFPtr::getTemperatureASU2 ( ) const [inline]

Definition at line 118 of file DIFPtr.h.

```
00118 { return (getTASU2() » 3) * 0.0625; }
```

4.5.2.24 getTemperatureDIF() float DIFPtr::getTemperatureDIF ( ) const [inline]

Definition at line 114 of file DIFPtr.h.

```
00114 { return 0.508 * getTDIF() - 9.659; }
```

4.5.2.25 getThresholdStatus() uint32_t DIFPtr::getThresholdStatus (const std::uint32_t & *i*, const std::uint32_t & *ipad*) const [inline]

Definition at line 143 of file DIFPtr.h.

```
00143 { return (((std::uint32_t)getFrameLevel(i, ipad, 1)) « 1) | ((std::uint32_t)getFrameLevel(i, ipad,
      0)); }
```

4.5.2.26 hasAnalogReadout() bool DIFPtr::hasAnalogReadout ( ) const [inline]

Definition at line 122 of file DIFPtr.h.

```
00122 { return getLines() != 0; }
```

4.5.2.27 hasLine() bool DIFPtr::hasLine (const std::uint32_t & *line*) const [inline]

Definition at line 106 of file DIFPtr.h.

```
00106 { return ((theDIF_[DU::LINES_SHIFT] » line) & 0x1); }
```

4.5.2.28 hasTemperature() bool DIFPtr::hasTemperature ( ) const [inline]

Definition at line 120 of file DIFPtr.h.

```
00120 { return (theDIF_[0] == DU::START_OF_DIF_TEMP); }
```

4.5.2.29 setBuffer() `void DIFPtr::setBuffer (`

```
unsigned char * ,
const std::uint32_t &  )  [inline]
```

Definition at line 65 of file DIFPtr.h.

```
00066 {
00067   theFrames_.clear();
00068   theLines_.clear();
00069   theSize_ = max_size;
00070   theDIF_  = p;
00071   try
00072   {
00073     theGetFramePtrReturn_ = getFramePtr();
00074   }
00075   catch(const std::string& e)
00076   {
00077     spdlog::get("streamout")->error(" DIF {} T ? {} {}", getID(), hasTemperature(), e);
00078   }
00079 }
```

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

- libs/core/include/DIFPtr.h

4.6 DIFSlowControl Class Reference

`#include <libs/core/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.
- std::map< int, std::map< std::string, int > >::const_iterator cbegin () const
- std::map< int, std::map< std::string, int > >::const_iterator cend () const

4.6.1 Detailed Description

Definition at line 13 of file DIFSlowControl.h.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 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 7 of file DIFSlowControl.cc.

```
00007
      :  m_Version(version), m_DIFId(DIfId), m_AsicType(2)
00008 {
00009   if(cbuf[0] != 0xb1) return;
00010   int header_shift{6};
00011   if(m_Version < 8) m_NbrAsic = cbuf[5];
00012   else
00013   {
00014     m_DIFId      = cbuf[1];
00015     m_NbrAsic    = cbuf[2];
00016     header_shift = 3;
00017   }
00018   int size_hardroc1 = m_NbrAsic * 72 + header_shift + 1;
00019   if(cbuf[size_hardroc1 - 1] != 0xa1) size_hardroc1 = 0;
00020
00021   int size_hardroc2 = m_NbrAsic * 109 + header_shift + 1;
00022   if(cbuf[size_hardroc2 - 1] != 0xa1) size_hardroc2 = 0;
00023   if(size_hardroc1 != 0)
00024   {
00025     FillHR1(header_shift, cbuf);
00026     m_AsicType = 1;
00027   }
00028   else if(size_hardroc2 != 0)
00029     FillHR2(header_shift, cbuf);
00030   else
00031     return;
00032 }
```

4.6.3 Member Function Documentation

4.6.3.1 cbegin() std::map< int, std::map< std::string, int > >::const_iterator DIFSlowControl::cbegin ( ) const [inline]

Definition at line 47 of file DIFSlowControl.h.

```
00047 { return m_MapSC.cbegin(); }
```

4.6.3.2 cend() std::map< int, std::map< std::string, int > >::const_iterator DIFSlowControl::cend ( ) const [inline]

Definition at line 49 of file DIFSlowControl.h.

```
00049 { return m_MapSC.cend(); }
```

4.6.3.3 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 38 of file DIFSlowControl.cc.

```
00038 { return m_MapSC[asicid]; }
```

4.6.3.4 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 40 of file DIFSlowControl.cc.

```
00040 { return getChipSlowControl(asicid)[param]; }
```

4.6.3.5 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 36 of file DIFSlowControl.cc.

```
00036 { return m_MapSC; }
```

4.6.3.6 getDIFId()

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

get DIF id

Definition at line 34 of file DIFSlowControl.cc.

```
00034 { return m_DIFId; }
```

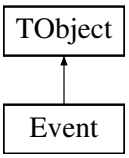
The documentation for this class was generated from the following files:

- libs/core/include/DIFSlowControl.h
- libs/core/src/DIFSlowControl.cc

4.7 Event Class Reference

`#include <libs/interface/ROOT/include/Event.h>`

Inheritance diagram for Event:



Public Member Functions

- void clear ()
- void addDIF (const DIF &dif)
- std::map< std::uint8_t, DIF >::const_iterator cbegin () const
- std::map< std::uint8_t, DIF >::const_iterator cend () const

4.7.1 Detailed Description

Definition at line 15 of file Event.h.

4.7.2 Member Function Documentation

4.7.2.1 addDIF() `void Event::addDIF ( const DIF & dif )`

Definition at line 10 of file Event.cc.

```
00010 { DIFs[dif.getID()] = dif; }
```

4.7.2.2 cbegin() `std::map< std::uint8_t, DIF >::const_iterator Event::cbegin ( ) const`

Definition at line 12 of file Event.cc.

```
00012 { return DIFs.cbegin(); }
```

4.7.2.3 cend() `std::map< std::uint8_t, DIF >::const_iterator Event::cend ( ) const`

Definition at line 14 of file Event.cc.

```
00014 { return DIFs.cend(); }
```

4.7.2.4 clear() `void Event::clear ( )`

Definition at line 8 of file Event.cc.

```
00008 { DIFs.clear(); }
```

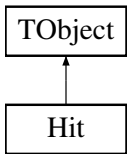
The documentation for this class was generated from the following files:

- libs/interface/ROOT/include/Event.h
- libs/interface/ROOT/src/Event.cc

4.8 Hit Class Reference

`#include <libs/interface/ROOT/include/Hit.h>`

Inheritance diagram for Hit:



Public Member Functions

- void clear ()
- void setDIF (const std::uint8_t &)
- void setASIC (const std::uint8_t &)
- void setChannel (const std::uint8_t &)
- void setThreshold (const std::uint8_t &)
- void setDTC (const std::uint32_t &)
- void setGTC (const std::uint32_t &)
- void setDIFBCID (const std::uint32_t &)
- void setFrameBCID (const std::uint32_t &)
- void setTimestamp (const std::uint32_t &)
- void setAbsoluteBCID (const std::uint64_t &)
- std::uint8_t getDIFid () const
- std::uint8_t getASICid () const
- std::uint8_t getChannel () const
- std::uint8_t getThreshold () const
- std::uint32_t getDTC () const
- std::uint32_t getGTC () const
- std::uint32_t getDIFBCID () const
- std::uint32_t getFrameBCID () const
- std::uint32_t getTimestamp () const
- std::uint64_t getAbsoluteBCID () const

4.8.1 Detailed Description

Definition at line 10 of file Hit.h.

4.8.2 Member Function Documentation

4.8.2.1 clear() `void Hit::clear ( )`

Definition at line 7 of file Hit.cc.

```
00008 {
00009   m_DIF          = 0;
00010   m_ASIC         = 0;
00011   m_Channel      = 0;
00012   m_Threshold    = 0;
00013   m_DTC          = 0;
00014   m_GTC          = 0;
00015   m_DIFBCID      = 0;
00016   m_FrameBCID    = 0;
00017   m_Timestamp    = 0;
00018   m_AbsoluteBCID = 0;
00019 }
```

4.8.2.2 getAbsoluteBCID() `std::uint64_t Hit::getAbsoluteBCID ( ) const`

Definition at line 59 of file Hit.cc.

```
00059 { return m_AbsoluteBCID; }
```

4.8.2.3 getASICid() `std::uint8_t Hit::getASICid ( ) const`

Definition at line 43 of file Hit.cc.

```
00043 { return m_ASIC; }
```

4.8.2.4 getChannel() `std::uint8_t Hit::getChannel ( ) const`

Definition at line 45 of file Hit.cc.

```
00045 { return m_Channel; }
```

4.8.2.5 getDIFBCID() `std::uint32_t Hit::getDIFBCID ( ) const`

Definition at line 53 of file Hit.cc.

```
00053 { return m_DIFBCID; }
```

4.8.2.6 getDIFid() `std::uint8_t Hit::getDIFid ( ) const`

Definition at line 41 of file Hit.cc.

```
00041 { return m_DIF; }
```

4.8.2.7 getDTC() std::uint32_t Hit::getDTC ( ) const

Definition at line 49 of file Hit.cc.
```
00049 { return m_DTC; }
```

4.8.2.8 getFrameBCID() std::uint32_t Hit::getFrameBCID ( ) const

Definition at line 55 of file Hit.cc.
```
00055 { return m_FrameBCID; }
```

4.8.2.9 getGTC() std::uint32_t Hit::getGTC ( ) const

Definition at line 51 of file Hit.cc.
```
00051 { return m_GTC; }
```

4.8.2.10 getThreshold() std::uint8_t Hit::getThreshold ( ) const

Definition at line 47 of file Hit.cc.
```
00047 { return m_Threshold; }
```

4.8.2.11 getTimestamp() std::uint32_t Hit::getTimestamp ( ) const

Definition at line 57 of file Hit.cc.
```
00057 { return m_Timestamp; }
```

4.8.2.12 setAbsoluteBCID() void Hit::setAbsoluteBCID (const std::uint64_t & *absolutebcid*)

Definition at line 39 of file Hit.cc.
```
00039 { m_AbsoluteBCID = absolutebcid; }
```

4.8.2.13 setASIC() void Hit::setASIC (const std::uint8_t & *asic*)

Definition at line 23 of file Hit.cc.
```
00023 { m_ASIC = asic; }
```

4.8.2.14 setChannel() void Hit::setChannel (const std::uint8_t & *channel*)

Definition at line 25 of file Hit.cc.

```
00025 { m_Channel = channel; }
```

4.8.2.15 setDIF() void Hit::setDIF (const std::uint8_t & *dif*)

Definition at line 21 of file Hit.cc.

```
00021 { m_DIF = dif; }
```

4.8.2.16 setDIFBCID() void Hit::setDIFBCID (const std::uint32_t & *difbcid*)

Definition at line 33 of file Hit.cc.

```
00033 { m_DIFBCID = difbcid; }
```

4.8.2.17 setDTC() void Hit::setDTC (const std::uint32_t & *dtc*)

Definition at line 29 of file Hit.cc.

```
00029 { m_DTC = dtc; }
```

4.8.2.18 setFrameBCID() void Hit::setFrameBCID (const std::uint32_t & *framebcid*)

Definition at line 35 of file Hit.cc.

```
00035 { m_FrameBCID = framebcid; }
```

4.8.2.19 setGTC() void Hit::setGTC (const std::uint32_t & *gtc*)

Definition at line 31 of file Hit.cc.

```
00031 { m_GTC = gtc; }
```

4.8.2.20 setThreshold() void Hit::setThreshold (const std::uint8_t & *threshold*)

Definition at line 27 of file Hit.cc.

```
00027 { m_Threshold = threshold; }
```

4.8.2.21 setTimestamp() `void Hit::setTimestamp ( const std::uint32_t & timestamp )`

Definition at line 37 of file Hit.cc.

```
00037 { m_Timestamp = timestamp; }
```

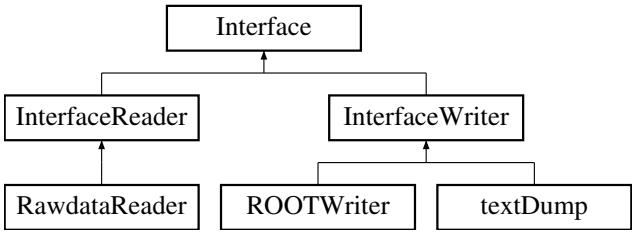
The documentation for this class was generated from the following files:

- libs/interface/ROOT/include/Hit.h
- libs/interface/ROOT/src/Hit.cc

4.9 Interface Class Reference

`#include <libs/core/include/Interface.h>`

Inheritance diagram for Interface:



Public Member Functions

- Interface (const std::string &name, const std::string &version, const InterfaceType &type)
- virtual ~Interface ()=default
- virtual void startEvent ()
- virtual void endEvent ()
- virtual void startDIF ()
- virtual void endDIF ()
- virtual void startFrame ()
- virtual void endFrame ()
- virtual void startPad ()
- virtual void endPad ()
- std::shared_ptr< spdlog::logger > & log ()
- void setLogger (const std::shared_ptr< spdlog::logger > &logger)
- std::string getName ()
- Version getVersion ()

4.9.1 Detailed Description

Definition at line 39 of file Interface.h.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 Interface()

```
Interface::Interface (
        const std::string & name,
        const std::string & version,
        const InterfaceType & type )  [inline]
```

Definition at line 42 of file Interface.h.

```
00042 :  m_Name(name), m_Version(version) {}
```

4.9.2.2 ~Interface()

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

4.9.3 Member Function Documentation

4.9.3.1 endDIF()

```
virtual void Interface::endDIF ( )  [inline], [virtual]
```

Reimplemented in ROOTWriter.

Definition at line 47 of file Interface.h.

```
00047 {}
```

4.9.3.2 endEvent()

```
virtual void Interface::endEvent ( )  [inline], [virtual]
```

Reimplemented in ROOTWriter.

Definition at line 45 of file Interface.h.

```
00045 {}
```

4.9.3.3 endFrame()

```
virtual void Interface::endFrame ( )  [inline], [virtual]
```

Reimplemented in ROOTWriter.

Definition at line 49 of file Interface.h.

```
00049 {}
```

4.9.3.4 endPad()

```
virtual void Interface::endPad ( )  [inline], [virtual]
```

Reimplemented in ROOTWriter.

Definition at line 51 of file Interface.h.

```
00051 {}
```

4.9.3.5 getName() `std::string Interface::getName ( ) [inline]`

Definition at line 54 of file Interface.h.

```
00054 { return m_Name; }
```

4.9.3.6 getVersion() `Version Interface::getVersion ( ) [inline]`

Definition at line 55 of file Interface.h.

```
00055 { return m_Version; }
```

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

Definition at line 52 of file Interface.h.

```
00052 { return m_Logger; }
```

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

Definition at line 53 of file Interface.h.

```
00053 { m_Logger = logger; }
```

4.9.3.9 startDIF() `virtual void Interface::startDIF ( ) [inline], [virtual]`

Reimplemented in ROOTWriter.

Definition at line 46 of file Interface.h.

```
00046 {}
```

4.9.3.10 startEvent() `virtual void Interface::startEvent ( ) [inline], [virtual]`

Reimplemented in ROOTWriter.

Definition at line 44 of file Interface.h.

```
00044 {}
```

4.9.3.11 startFrame() `virtual void Interface::startFrame ( ) [inline], [virtual]`

Reimplemented in ROOTWriter.

Definition at line 48 of file Interface.h.

```
00048 {}
```

4.9.3.12 startPad() `virtual void Interface::startPad ( ) [inline], [virtual]`

Reimplemented in ROOTWriter.

Definition at line 50 of file Interface.h.
```
00050 {}
```

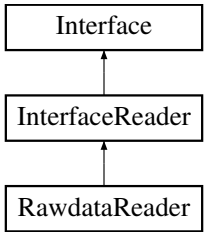
The documentation for this class was generated from the following file:

- libs/core/include/Interface.h

4.10 InterfaceReader Class Reference

`#include <libs/core/include/Interface.h>`

Inheritance diagram for InterfaceReader:



Public Member Functions

- InterfaceReader (const std::string &name, const std::string &version)
- virtual ~InterfaceReader ()=default

Protected Attributes

- Buffer m_Buffer

4.10.1 Detailed Description

Definition at line 64 of file Interface.h.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 InterfaceReader() `InterfaceReader::InterfaceReader ( const std::string & name, const std::string & version ) [inline]`

Definition at line 67 of file Interface.h.
```
00067 : Interface(name, version, InterfaceType::Reader) {}
```

4.10.2.2 ~InterfaceReader() `virtual InterfaceReader::~InterfaceReader ( ) [virtual], [default]`

4.10.3 Member Data Documentation

4.10.3.1 m_Buffer `Buffer InterfaceReader::m_Buffer [protected]`

Definition at line 71 of file Interface.h.

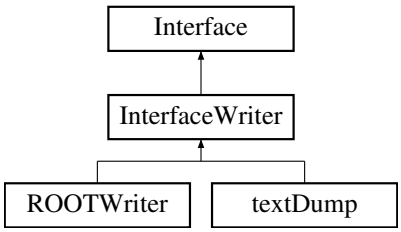
The documentation for this class was generated from the following file:

- libs/core/include/Interface.h

4.11 InterfaceWriter Class Reference

`#include <libs/core/include/Interface.h>`

Inheritance diagram for InterfaceWriter:



Public Member Functions

- InterfaceWriter (const std::string &name, const std::string &version)
- void addCompatibility (const std::string &name, const std::string &version)
- std::map< std::string, std::string > getCompatibility ()
- bool checkCompatibility (const std::string &name, const std::string &version)
- virtual ~InterfaceWriter ()=default

4.11.1 Detailed Description

Definition at line 74 of file Interface.h.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 InterfaceWriter()

```
InterfaceWriter::InterfaceWriter (
        const std::string & name,
        const std::string & version )  [inline]
```

Definition at line 77 of file Interface.h.

```
00077 :  Interface(name, version, InterfaceType::Writer) {}
```

4.11.2.2 ~InterfaceWriter()

```
virtual InterfaceWriter::~InterfaceWriter ( )  [virtual], [default]
```

4.11.3 Member Function Documentation

4.11.3.1 addCompatibility()

```
void InterfaceWriter::addCompatibility (
        const std::string & name,
        const std::string & version )  [inline]
```

Definition at line 79 of file Interface.h.

```
00079 { m_Compatible[name] = version; }
```

4.11.3.2 checkCompatibility()

```
bool InterfaceWriter::checkCompatibility (
        const std::string & name,
        const std::string & version )  [inline]
```

Definition at line 83 of file Interface.h.

```
00084   {
00085     if(m_Compatible.find(name) != m_Compatible.end())
00086     {
00087       auto            ran = semver::range::detail::range(m_Compatible[name]);
00088       semver::version ver = semver::version(version);
00089       if(ran.satisfies(ver, false)) return true;
00090       else
00091         return false;
00092     }
00093     else
00094       return false;
00095   }
```

4.11.3.3 getCompatibility()

```
std::map< std::string, std::string > InterfaceWriter::getCompatibility
( )  [inline]
```

Definition at line 81 of file Interface.h.

```
00081 { return m_Compatible; }
```

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

- libs/core/include/Interface.h

4.12 RawBufferNavigator Class Reference

`#include <libs/core/include/RawBufferNavigator.h>`

Public Member Functions

- RawBufferNavigator (const std::shared_ptr< spdlog::logger > &)
- ~RawBufferNavigator ()=default
- RawBufferNavigator (const Buffer &b)
- void setBuffer (const Buffer &b)
- std::uint8_t getDetectorID ()
- bool validBuffer ()
- bit8_t * 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 ()
- std::int32_t getStartOfPayload ()

Static Public Member Functions

- static void StartAt (const int &start)

4.12.1 Detailed Description

Definition at line 14 of file RawBufferNavigator.h.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 RawBufferNavigator() [1/2] `RawBufferNavigator::RawBufferNavigator ( const std::shared_ptr< spdlog::logger > & logger ) [explicit]`

Definition at line 10 of file RawBufferNavigator.cc.

```
00010 { m_Logger = logger; }
```

4.12.2.2 ~RawBufferNavigator() `RawBufferNavigator::~RawBufferNavigator ( ) [default]`

4.12.2.3 RawBufferNavigator() [2/2] RawBufferNavigator::RawBufferNavigator (const Buffer & b) [explicit]

Definition at line 40 of file RawBufferNavigator.cc.

```
00040 :  m_Buffer(b) { setBuffer(b); }
```

4.12.3 Member Function Documentation

4.12.3.1 badSCData() bool RawBufferNavigator::badSCData ( )

Definition at line 79 of file RawBufferNavigator.cc.

```
00080 {
00081   setSCBuffer();
00082   return m_BadSCdata;
00083 }
```

4.12.3.2 getDetectorID() std::uint8_t RawBufferNavigator::getDetectorID ( )

Definition at line 42 of file RawBufferNavigator.cc.

```
00042 { return m_Buffer[0]; }
```

4.12.3.3 getDIF_CRC() std::uint32_t RawBufferNavigator::getDIF_CRC ( )

Definition at line 62 of file RawBufferNavigator.cc.

```
00063 {
00064   uint32_t i{getEndOfDIFData()};
00065   uint32_t ret{0};
00066   ret |= ((m_Buffer.begin()[i - 2]) « 8);
00067   ret |= m_Buffer.begin()[i - 1];
00068   return ret;
00069 }
```

4.12.3.4 getDIFBuffer() Buffer RawBufferNavigator::getDIFBuffer ( )

Definition at line 50 of file RawBufferNavigator.cc.

```
00050 { return Buffer(getDIFBufferStart(), getDIFBufferSize()); }
```

4.12.3.5 getDIFBufferSize() std::uint32_t RawBufferNavigator::getDIFBufferSize ( )

Definition at line 48 of file RawBufferNavigator.cc.

```
00048 { return m_Buffer.size() - m_StartPayload; }
```

4.12.3.6 getDIFBufferStart() `bit8_t * RawBufferNavigator::getDIFBufferStart ( )`

Definition at line 46 of file RawBufferNavigator.cc.

```
00046 { return &(m_Buffer.begin()[m_StartPayload]); }
```

4.12.3.7 getDIFPtr() `DIFPtr & RawBufferNavigator::getDIFPtr ( )`

Definition at line 52 of file RawBufferNavigator.cc.

```
00053 {
00054   m_TheDIFPtr.setBuffer(getDIFBufferStart(), getDIFBufferSize());
00055   return m_TheDIFPtr;
00056 }
```

4.12.3.8 getEndOfAllData() `Buffer RawBufferNavigator::getEndOfAllData ( )`

Definition at line 118 of file RawBufferNavigator.cc.

```
00119 {
00120   setSCBuffer();
00121   if(hasSlowControlData() && !m_BadSCdata) { return Buffer(&(m_SCbuffer.begin()[m_SCbuffer.size()]),
      getSizeAfterDIFPtr() - 3 - m_SCbuffer.size()); }
00122   else
00123     return Buffer(&(getDIFBufferStart()[getEndOfDIFData()]), getSizeAfterDIFPtr() - 3);  // remove the
      2 bytes for CRC and the DIF trailer
00124 }
```

4.12.3.9 getEndOfDIFData() `std::uint32_t RawBufferNavigator::getEndOfDIFData ( )`

Definition at line 58 of file RawBufferNavigator.cc.

```
00058 { return getDIFPtr().getGetFramePtrReturn() + 3; }
```

4.12.3.10 getSCBuffer() `Buffer RawBufferNavigator::getSCBuffer ( )`

Definition at line 73 of file RawBufferNavigator.cc.

```
00074 {
00075   setSCBuffer();
00076   return m_SCbuffer;
00077 }
```

4.12.3.11 getSizeAfterDIFPtr() `std::uint32_t RawBufferNavigator::getSizeAfterDIFPtr ( )`

Definition at line 60 of file RawBufferNavigator.cc.

```
00060 { return getDIFBufferSize() - getDIFPtr().getGetFramePtrReturn(); }
```

4.12.3.12 getStartOfPayload() std::int32_t RawBufferNavigator::getStartOfPayload ( )

Definition at line 12 of file RawBufferNavigator.cc.

```
00013 {
00014   for(std::size_t i = m_Start; i < m_Buffer.size(); i++)
00015   {
00016     if(m_Buffer[i] == DU::START_OF_DIF || m_Buffer[i] == DU::START_OF_DIF_TEMP)
00017     {
00018       m_StartPayload = i;
00019       return m_StartPayload;
00020     }
00021   }
00022   m_StartPayload = -1;
00023   return m_StartPayload;
00024 }
```

4.12.3.13 hasSlowControlData() bool RawBufferNavigator::hasSlowControlData ( )

Definition at line 71 of file RawBufferNavigator.cc.

```
00071 { return getDIFBufferStart()[getEndOfDIFData()] == 0xb1; }
```

4.12.3.14 setBuffer() void RawBufferNavigator::setBuffer (const Buffer & *b*)

Definition at line 33 of file RawBufferNavigator.cc.

```
00034 {
00035   m_BadSCdata = false;
00036   m_Buffer    = b;
00037   // m_DIFstartIndex = getStartOfPayload();
00038 }
```

4.12.3.15 StartAt() void RawBufferNavigator::StartAt (const int & *start*) [static]

Definition at line 28 of file RawBufferNavigator.cc.

```
00029 {
00030   if(start >= 0) m_Start = start;
00031 }
```

4.12.3.16 validBuffer() bool RawBufferNavigator::validBuffer ( )

Definition at line 44 of file RawBufferNavigator.cc.

```
00044 { return m_StartPayload != -1; }
```

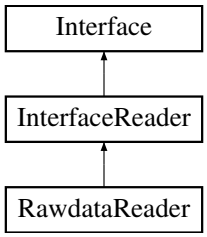
The documentation for this class was generated from the following files:

- libs/core/include/RawBufferNavigator.h
- libs/core/src/RawBufferNavigator.cc

4.13 RawdataReader Class Reference

`#include <libs/interface/RawDataReader/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 ()
- const Buffer & getBuffer ()
- virtual ~RawdataReader ()

Static Public Member Functions

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

Additional Inherited Members

4.13.1 Detailed Description

Definition at line 17 of file RawdataReader.h.

4.13.2 Constructor & Destructor Documentation

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

Definition at line 16 of file RawdataReader.cc.

```
00016                                                 : InterfaceReader("RawdataReader", "1.0.0")
00017 {
00018   m_buf.reserve(m_BufferSize);
00019   m_Filename = fileName;
00020 }
```

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

Definition at line 29 of file RawdataReader.h.

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

4.13.3 Member Function Documentation

4.13.3.1 closeFile() void RawdataReader::closeFile ( )

Definition at line 42 of file RawdataReader.cc.

```
00043 {
00044   try
00045   {
00046     if(m_FileStream.is_open()) m_FileStream.close();
00047   }
00048   catch(const std::ios_base::failure& e)
00049   {
00050     log()->error("Caught an ios_base::failure in closeFile :  {} {}", e.what(), e.code().value());
00051     throw;
00052   }
00053 }
```

4.13.3.2 end() void RawdataReader::end ( )

Definition at line 24 of file RawdataReader.cc.

```
00024 { closeFile(); }
```

4.13.3.3 getBuffer() const Buffer & RawdataReader::getBuffer ( )

Definition at line 117 of file RawdataReader.cc.

```
00118 {
00119   uncompress();
00120   return m_Buffer;
00121 }
```

4.13.3.4 getFileSize() float RawdataReader::getFileSize ( )

Definition at line 125 of file RawdataReader.cc.

```
00125 { return m_FileSize; }
```

4.13.3.5 nextDIFbuffer() bool RawdataReader::nextDIFbuffer ( )

Definition at line 90 of file RawdataReader.cc.

```
00091 {
00092   try
00093   {
00094     static int DIF_processed{0};
00095     if(DIF_processed >= m_NumberOfDIF)
00096     {
00097       DIF_processed = 0;
00098       return false;
00099     }
00100     else
00101     {
00102       DIF_processed++;
00103       std::uint32_t bsize{0};
00104       m_FileStream.read(reinterpret_cast<char*>(&bsize), sizeof(std::uint32_t));
00105       m_FileStream.read(reinterpret_cast<char*>(&m_buf[0]), bsize);
00106       m_Buffer = Buffer(m_buf);
00107     }
00108   }
00109   catch(const std::ios_base::failure& e)
00110   {
00111     log()->error("Caught an ios_base::failure in openFile :  {}", e.what());
00112     return false;
00113   }
00114   return true;
00115 }
```

4.13.3.6 nextEvent() bool RawdataReader::nextEvent ( )

Definition at line 76 of file RawdataReader.cc.

```
00077 {
00078   try
00079   {
00080     m_FileStream.read(reinterpret_cast<char*>(&m_EventNumber), sizeof(std::uint32_t));
00081     m_FileStream.read(reinterpret_cast<char*>(&m_NumberOfDIF), sizeof(std::uint32_t));
00082   }
00083   catch(const std::ios_base::failure& e)
00084   {
00085     return false;
00086   }
00087   return true;
00088 }
```

4.13.3.7 openFile() void RawdataReader::openFile (const std::string & *fileName*)

Definition at line 55 of file RawdataReader.cc.

```
00056 {
00057   try
00058   {
00059     m_FileStream.rdbuf()->pubsetbuf(0, 0);
00060     m_FileStream.exceptions(std::ifstream::failbit | std::ifstream::badbit);
00061     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
00062     m_FileStream.rdbuf()->pubsetbuf(0, 0);
00063     if(m_FileStream.is_open())
00064     {
00065       setFileSize(m_FileStream.tellg());
00066       m_FileStream.seekg(0, std::ios::beg);
00067     }
00068   }
00069   catch(const std::ios_base::failure& e)
00070   {
00071     log()->error("Caught an ios_base::failure in openFile :  {}", e.what());
00072     throw;
00073   }
00074 }
```

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

Definition at line 14 of file RawdataReader.cc.

```
00014 { m_BufferSize = size; }
```

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

Definition at line 22 of file RawdataReader.cc.

```
00022 { openFile(m_Filename); }
```

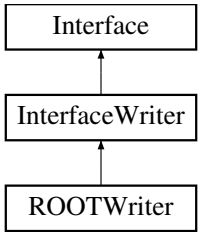
The documentation for this class was generated from the following files:

- libs/interface/RawDataReader/include/RawdataReader.h
- libs/interface/RawDataReader/src/RawdataReader.cc

4.14 ROOTWriter Class Reference

`#include <libs/interface/ROOT/include/ROOTWriter.h>`

Inheritance diagram for ROOTWriter:



Public Member Functions

- ROOTWriter ()
- void setFilename (const std::string &)
- void start ()
- void processDIF (const DIFPtr &)
- void processFrame (const DIFPtr &, const std::uint32_t &frameIndex)
- void processPadInFrame (const DIFPtr &, const std::uint32_t &frameIndex, const std::uint32_t &channel↩ Index)
- void processSlowControl (const Buffer &)
- void end ()
- virtual void startEvent ()
- virtual void endEvent ()
- virtual void startDIF ()
- virtual void endDIF ()
- virtual void startFrame ()
- virtual void endFrame ()
- virtual void startPad ()
- virtual void endPad ()

4.14.1 Detailed Description

Definition at line 18 of file ROOTWriter.h.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 ROOTWriter() `ROOTWriter::ROOTWriter ( )`

Definition at line 10 of file ROOTWriter.cc.

```
00010 :  InterfaceWriter("ROOTWriter", "1.0.0") { addCompatibility("RawdataReader", ">=1.0.0"); }
```

4.14.3 Member Function Documentation

4.14.3.1 end() `void ROOTWriter::end ( )`

Definition at line 19 of file ROOTWriter.cc.

```
00020 {
00021   if(m_Tree) m_Tree->Write();
00022   if(m_File)
00023   {
00024     m_File->Write();
00025     m_File->Close();
00026   }
00027   if(m_File) delete m_File;
00028 }
```

4.14.3.2 endDIF() `void ROOTWriter::endDIF ( ) [virtual]`

Reimplemented from Interface.

Definition at line 75 of file ROOTWriter.cc.

```
00076 {
00077   m_Event->addDIF(*m_DIF);
00078   delete m_DIF;
00079 }
```

4.14.3.3 endEvent() `void ROOTWriter::endEvent ( ) [virtual]`

Reimplemented from Interface.

Definition at line 63 of file ROOTWriter.cc.

```
00064 {
00065   m_Tree->Fill();
00066   if(m_Event) delete m_Event;
00067 }
```

4.14.3.4 endFrame() void ROOTWriter::endFrame ( ) [virtual]

Reimplemented from Interface.

Definition at line 87 of file ROOTWriter.cc.

```
00088 {
00089   m_DIF->addHit(*m_Hit);
00090   delete m_Hit;
00091 }
```

4.14.3.5 endPad() void ROOTWriter::endPad ( ) [virtual]

Reimplemented from Interface.

Definition at line 95 of file ROOTWriter.cc.

```
00095 {}
```

4.14.3.6 processDIF() void ROOTWriter::processDIF (const DIFPtr & d)

Definition at line 30 of file ROOTWriter.cc.

```
00031 {
00032   m_DIF->setID(d.getDIFid());
00033   m_DIF->setDTC(d.getDTC());
00034   m_DIF->setGTC(d.getGTC());
00035   m_DIF->setDIFBCID(d.getBCID());
00036   m_DIF->setAbsoluteBCID(d.getAbsoluteBCID());
00037 }
```

4.14.3.7 processFrame() void ROOTWriter::processFrame (const DIFPtr & d, const std::uint32_t & frameIndex)

Definition at line 39 of file ROOTWriter.cc.

```
00040 {
00041   m_Hit->setDIF(d.getDIFid());
00042   m_Hit->setASIC(d.getASICid(frameIndex));
00043   m_Hit->setDTC(d.getDTC());
00044   m_Hit->setGTC(d.getGTC());
00045   m_Hit->setDIFBCID(d.getBCID());
00046   m_Hit->setAbsoluteBCID(d.getAbsoluteBCID());
00047   m_Hit->setFrameBCID(d.getFrameBCID(frameIndex));
00048   m_Hit->setTimestamp(d.getFrameTimeToTrigger(frameIndex));
00049 }
```

4.14.3.8 processPadInFrame() void ROOTWriter::processPadInFrame (const DIFPtr & d, const std::uint32_t & frameIndex, const std::uint32_t & channelIndex)

Definition at line 51 of file ROOTWriter.cc.

```
00052 {
00053   m_Hit->setChannel(channelIndex);
00054   m_Hit->setThreshold(static_cast<std::uint8_t>(d.getThresholdStatus(frameIndex, channelIndex)));
00055 }
```

4.14.3.9 processSlowControl() void ROOTWriter::processSlowControl (const Buffer &) [inline]

Definition at line 29 of file ROOTWriter.h.

```
00029 { ; }
```

4.14.3.10 setFilename() void ROOTWriter::setFilename (const std::string & *filename*)

Definition at line 8 of file ROOTWriter.cc.

```
00008 { m_Filename = filename; }
```

4.14.3.11 start() void ROOTWriter::start ( )

Definition at line 12 of file ROOTWriter.cc.

```
00013 {
00014   m_File = TFile::Open(m_Filename.c_str(), "RECREATE", m_Filename.c_str(),
      ROOT::CompressionSettings(ROOT::kZLIB, 5));
00015   m_Tree = new TTree("RawData", "Raw SDHCAL data tree");
00016   m_Tree->Branch("Events", &m_Event, 512000, 99);
00017 }
```

4.14.3.12 startDIF() void ROOTWriter::startDIF ( ) [virtual]

Reimplemented from Interface.

Definition at line 69 of file ROOTWriter.cc.

```
00070 {
00071   m_DIF = new DIF();
00072   // m_DIF->clear();
00073 }
```

4.14.3.13 startEvent() void ROOTWriter::startEvent ( ) [virtual]

Reimplemented from Interface.

Definition at line 57 of file ROOTWriter.cc.

```
00058 {
00059   m_Event = new Event();
00060   // m_Event->clear();
00061 }
```

4.14.3.14 startFrame() void ROOTWriter::startFrame ( ) [virtual]

Reimplemented from Interface.

Definition at line 81 of file ROOTWriter.cc.

```
00082 {
00083   m_Hit = new Hit();
00084   // m_Hit->clear();
00085 }
```

4.14.3.15 startPad() `void ROOTWriter::startPad ( ) [virtual]`

Reimplemented from Interface.

Definition at line 93 of file ROOTWriter.cc.
```
00093 {}
```

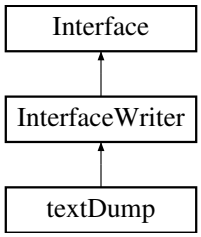
The documentation for this class was generated from the following files:

- libs/interface/ROOT/include/ROOTWriter.h
- libs/interface/ROOT/src/ROOTWriter.cc

4.15 textDump Class Reference

`#include <libs/interface/Dump/include/textDump.h>`

Inheritance diagram for textDump:



Public Member Functions

- textDump ()
- void start ()
- void processDIF (const DIFPtr &)
- void processFrame (const DIFPtr &, uint32_t frameIndex)
- void processPadInFrame (const 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)

4.15.1 Detailed Description

Definition at line 14 of file textDump.h.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 textDump() textDump::textDump ( )

Definition at line 9 of file textDump.cc.

```
00009                     :  InterfaceWriter("textDump", "1.0.0")
00010 {
00011   m_InternalLogger = std::make_shared<spdlog::logger>("textDump",
      std::make_shared<spdlog::sinks::stdout_color_sink_mt>());
00012   m_InternalLogger->set_level(spdlog::level::trace);
00013   addCompatibility("RawdataReader", ">=1.0.0");
00014   addCompatibility("DIFdataExample", ">=1.0.0");
00015 }
```

4.15.3 Member Function Documentation

4.15.3.1 end() void textDump::end ( )

Definition at line 33 of file textDump.cc.

```
00033 { print()->info("textDump end of report"); }
```

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

Definition at line 24 of file textDump.h.

```
00024 { return m_InternalLogger; }
```

4.15.3.3 processDIF() void textDump::processDIF (const DIFPtr & *d*)

Definition at line 19 of file textDump.cc.

```
00019 { print()->info("DIF_ID : {}, DTC : {}, GTC : {}, DIF BCID {}, Absolute BCID : {}, Nbr frames {}",
      d.getDIFid(), d.getDTC(), d.getGTC(), d.getBCID(), d.getAbsoluteBCID(), d.getNumberOfFrames()); }
```

4.15.3.4 processFrame() void textDump::processFrame (const DIFPtr & *d*, uint32_t *frameIndex*)

Definition at line 21 of file textDump.cc.

```
00022 {
00023   print()->info("\tDisplaying frame number {} :  ASIC ID {}, Frame BCID {}, Frame Time To Trigger
      (a.k.a timestamp) is {}", frameIndex, d.getASICid(frameIndex), d.getFrameBCID(frameIndex),
      d.getFrameTimeToTrigger(frameIndex));
00024 }
```

4.15.3.5 processPadInFrame() void textDump::processPadInFrame (const DIFPtr & d, uint32_t frameIndex, uint32_t channelIndex)

Definition at line 26 of file textDump.cc.

```
00027 {
00028   if(d.getThresholdStatus(frameIndex, channelIndex) > 0) { print()->info("\t\tChannel {}, Threshold
      {}", channelIndex, d.getThresholdStatus(frameIndex, channelIndex)); }
00029 }
```

4.15.3.6 processSlowControl() void textDump::processSlowControl (Buffer)

Definition at line 31 of file textDump.cc.

```
00031 { print()->error("textDump::processSlowControl not implemented yet."); }
```

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

Definition at line 25 of file textDump.h.

```
00025 { m_InternalLogger->set_level(level); }
```

4.15.3.8 start() void textDump::start ( )

Definition at line 17 of file textDump.cc.

```
00017 { print()->info("Will dump bunch of DIF data"); }
```

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

- libs/interface/Dump/include/textDump.h
- libs/interface/Dump/src/textDump.cc

4.16 Timer Class Reference

`#include <libs/core/include/Timer.h>`

Public Member Functions

- void start ()
- void stop ()
- float getElapsedTime ()

4.16.1 Detailed Description

Definition at line 9 of file Timer.h.

4.16.2 Member Function Documentation

4.16.2.1 getElapsedTime() `float Timer::getElapsedTime ( ) [inline]`

Definition at line 14 of file Timer.h.

```
00014 { return std::chrono::duration_cast<std::chrono::microseconds>(m_StopTime - m_StartTime).count(); }
```

4.16.2.2 start() `void Timer::start ( ) [inline]`

Definition at line 12 of file Timer.h.

```
00012 { m_StartTime = std::chrono::high_resolution_clock::now(); }
```

4.16.2.3 stop() `void Timer::stop ( ) [inline]`

Definition at line 13 of file Timer.h.

```
00013 { m_StopTime = std::chrono::high_resolution_clock::now(); }
```

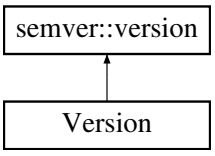
The documentation for this class was generated from the following file:

- libs/core/include/Timer.h

4.17 Version Class Reference

`#include <libs/core/include/Version.h>`

Inheritance diagram for Version:



Public Member Functions

- Version (const std::uint8_t &mj, const std::uint8_t &mn, const std::uint8_t &pt, const semver::prerelease &prt=semver::prerelease::none, const std::uint8_t &prn=0) noexcept
- Version (const std::string_view &str)
- Version ()=default
- std::uint8_t getMajor ()
- std::uint8_t getMinor ()
- std::uint8_t getPatch ()
- std::string getPreRelease ()
- std::uint8_t getPreReleaseNumber ()

4.17.1 Detailed Description

Definition at line 11 of file Version.h.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 Version() [1/3] Version::Version (

```
const std::uint8_t & mj,
const std::uint8_t & mn,
const std::uint8_t & pt,
const semver::prerelease & prt = semver::prerelease::none,
const std::uint8_t & prn = 0 )  [inline], [noexcept]
```

Definition at line 14 of file Version.h.

```
00014 :  semver::version(mj, mn, pt, prt, prn) {}
```

4.17.2.2 Version() [2/3] Version::Version (

```
const std::string_view & str )  [inline], [explicit]
```

Definition at line 15 of file Version.h.

```
00015 :  semver::version(str) {}
```

4.17.2.3 Version() [3/3] Version::Version ( ) [default]

4.17.3 Member Function Documentation

4.17.3.1 getMajor() std::uint8_t Version::getMajor ( )

Definition at line 9 of file Version.cc.

```
00009 { return major; }
```

4.17.3.2 getMinor() std::uint8_t Version::getMinor ( )

Definition at line 11 of file Version.cc.

```
00011 { return minor; }
```

4.17.3.3 getPatch() std::uint8_t Version::getPatch ( )

Definition at line 13 of file Version.cc.

```
00013 { return patch; }
```

4.17.3.4 getPreRelease() std::string Version::getPreRelease ( )

Definition at line 15 of file Version.cc.

```
00016 {
00017   switch(prerelease_type)
00018   {
00019     case semver::prerelease::alpha:  return "alpha";
00020     case semver::prerelease::beta:  return "beta";
00021     case semver::prerelease::rc:  return "rc";
00022     case semver::prerelease::none:  return "";
00023     default:  return "";
00024   }
00025 }
```

4.17.3.5 getPreReleaseNumber() std::uint8_t Version::getPreReleaseNumber ( )

Definition at line 27 of file Version.cc.

```
00027 { return prerelease_number; }
```

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

- libs/core/include/Version.h
- libs/core/src/Version.cc

5 File Documentation

5.1 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 libs/core/include/Buffer.h File Reference

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

Classes

- class Buffer

5.3.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde A.Pingault L.Mirabito

See also

https://github.com/apingault/Trivent4HEP

Definition in file Buffer.h.

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 <string>
00012 #include <vector>
00013
00014 class Buffer
00015 {
00016 public:
00017   Buffer() :  m_Buffer(nullptr), m_Size(0), m_Capacity(0) {}
00018   virtual ~Buffer() {}
00019   Buffer(const bit8_t b[], const std::size_t& i) :  m_Buffer(const_cast<bit8_t*>(&b[0])), m_Size(i),
      m_Capacity(i) {}
```

```
00020   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 * sizeof(char)), m_Capacity(i * sizeof(char)) {}
00021   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)) {}
00022   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)) {}
00023
00024   std::size_t size()const { return m_Size; }
00025   std::size_t capacity()const { return m_Capacity; }
00026
00027   void    set(unsigned char* b) { m_Buffer = b; }
00028   bit8_t* begin()const { return m_Buffer; }
00029   bit8_t* end()const { return m_Buffer + m_Size; }
00030   bit8_t& operator[](const std::size_t& pos) { return m_Buffer[pos]; }
00031   bit8_t& operator[](const std::size_t& pos)const { return m_Buffer[pos]; }
00032
00033   void setSize(const std::size_t& size) { m_Size = size; }
00034
00035 private:
00036   bit8_t*     m_Buffer{nullptr};
00037   std::size_t m_Size{0};
00038   std::size_t m_Capacity{0};
00039 };
```

5.5 libs/core/include/BufferLooper.h File Reference

```
#include "AppVersion.h"
#include "Buffer.h"
#include "BufferLooperCounter.h"
#include "DetectorId.h"
#include "Formatters.h"
#include "RawBufferNavigator.h"
#include "Timer.h"
#include "Words.h"
#include <algorithm>
#include <cassert>
#include <fmt/color.h>
#include <map>
#include <memory>
#include <spdlog/sinks/null_sink.h>
#include <spdlog/spdlog.h>
#include <string>
#include <vector>
```

Classes

- class BufferLooper< SOURCE, DESTINATION >

5.5.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file BufferLooper.h.

5.6 BufferLooper.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include "AppVersion.h"
00008 #include "Buffer.h"
00009 #include "BufferLooperCounter.h"
00010 #include "DetectorId.h"
00011 #include "Formatters.h"
00012 #include "RawBufferNavigator.h"
00013 #include "Timer.h"
00014 #include "Words.h"
00015
00016 #include <algorithm>
00017 #include <cassert>
00018 #include <fmt/color.h>
00019 #include <map>
00020 #include <memory>
00021 #include <spdlog/sinks/null_sink.h>
00022 #include <spdlog/spdlog.h>
00023 #include <string>
00024 #include <vector>
00025 // function to loop on buffers
00026
00027 template<typename SOURCE, typename DESTINATION> class BufferLooper
00028 {
00029 public:
00030   BufferLooper(SOURCE& source, DESTINATION& dest, bool debug = false) :  m_Source(source),
      m_Destination(dest), m_Debug(debug)
00031   {
00032     m_Logger = spdlog::create<spdlog::sinks::null_sink_mt>("streamout");
00033     if(!spdlog::get("streamout")) { spdlog::register_logger(m_Logger); }
00034     m_Source.setLogger(m_Logger);
00035     m_Destination.setLogger(m_Logger);
00036   }
00037
00038   void addSink(const spdlog::sink_ptr& sink, const spdlog::level::level_enum& level =
      spdlog::get_level())
00039   {
00040     sink->set_level(level);
00041     m_Sinks.push_back(sink);
00042     m_Logger = std::make_shared<spdlog::logger>("streamout", begin(m_Sinks), end(m_Sinks));
00043     m_Source.setLogger(m_Logger);
00044     m_Destination.setLogger(m_Logger);
00045   }
00046
00047   void loop(const std::uint32_t& m_NbrEventsToProcess = 0)
00048   {
00049     // clang-format off
00050     fmt::print(fg(fmt::color::medium_orchid) | fmt::emphasis::bold,
00051           "\n"
00052 " SSSSSSSSSSSSSSS      tttt
      tttt\n"
00053 "SS:::::::::::::::S  ttt:::t
      ttt:::t\n"
00054 "S:::::SSSSSS::::::S  t:::::t
      t:::::t\n"
00055 "S:::::S     SSSSSSS  t:::::t
      t:::::t\n"
00056 "S:::::S        ttttttt:::::ttttttt   rrrrr   rrrrrrrrr       eeeeeeeeeeee    aaaaaaaaaaaaa
      mmmmmmm    mmmmmmm      ooooooooooo   uuuuuu    uuuuuuttttttttt:::::ttttttt\n"
00057 "S:::::S        t:::::::::::::::::t   r::::rrr:::::::::r    ee::::::::::::ee  a::::::::::::a
      mm:::::::m  m:::::::mm  oo:::::::::::oo u:::::u    u:::::ut:::::::::::::::::t\n"
00058 " S::::SSSS     t:::::::::::::::::t   r:::::::::::::::::r  e::::::eeeee:::::eeaaaaaaaaa:::::a
      m::::::::::mm::::::::::mo:::::::::::::::ou::::u    u::::ut:::::::::::::::::t\n"
00059 "  SS::::::SSSSStttttt:::::::tttttt   rr::::::rrrrr::::::re::::::e     e:::::e         a::::a
      m::::::::::::::::::::::mo:::::ooooo:::::ou::::u    u::::utttttt:::::::tttttt\n"
00060 "    SSS::::::::SS    t:::::t          r:::::r     r:::::re:::::::eeeee:::::e  aaaaaaa:::::a
      m:::::mmm::::::mmm:::::mo::::o     o::::ou::::u    u::::u      t:::::t\n"
00061 "       SSSSSS::::S   t:::::t          r:::::r     rrrrrrre:::::::::::::::::e aa::::::::::::a m::::m
      m::::m   m::::mo::::o     o::::ou::::u    u::::u      t:::::t\n"
00062 "            S:::::S  t:::::t          r:::::r          e::::::eeeeeeeeeee a::::aaaa::::::a m::::m
      m::::m   m::::mo::::o     o::::ou::::u    u::::u      t:::::t\n"
00063 "            S:::::S  t:::::t    ttttttr:::::r          e:::::::e         a::::a    a:::::a m::::m
      m::::m   m::::mo::::o     o::::ou:::::uuuu:::::u      t:::::t    ttttttt\n"
00064 "SSSSSSS     S:::::S  t:::::::ttttt:::::tr:::::r          e::::::::e        a::::a    a:::::a m::::m
      m::::m   m::::mo:::::ooooo:::::ou::::::::::::::uu    t::::::tttt:::::t\n"
00065 "S::::::SSSSSS:::::S  tt::::::::::::::tr::::::r           e::::::::eeeeeeeea:::::aaaa::::::a m::::m
      m::::m   m::::mo::::::::::::::::o u::::::::::::::u    tt::::::::::::::t\n"
00066 "S:::::::::::::::SS     tt:::::::::::::ttr:::::::r            ee::::::::::::e a:::::::::::aa:::am::::m
      m::::m   m::::m oo:::::::::::oo   uu::::::::uu:::u      tt:::::::::::tt\n"
00067 " SSSSSSSSSSSSSSS         ttttttttttt  rrrrrrr               eeeeeeeeeeeeee  aaaaaaaaaa  aaaammmmmm
      mmmmmm   mmmmmm   ooooooooooo       uuuuuuuu  uuuu        ttttttttttt {}\n"
00068 "\n",
```

```
00069 fmt::format(fg(fmt::color::red) | fmt::emphasis::bold, "v{}", streamout_version.to_string()));
00070     // clang-format on
00071     log()->info("*****************************************************************");
00072     log()->info("Streamout Version :  {}", streamout_version.to_string());
00073     log()->info("Using InterfaceReader {} version {}", m_Source.getName(),
      m_Source.getVersion().to_string());
00074     log()->info("Using InterfaceWriter {} version {}", m_Destination.getName(),
      m_Destination.getVersion().to_string());
00075
00076     if(!m_Destination.checkCompatibility(m_Source.getName(), m_Source.getVersion().to_string()))
00077     {
00078       log()->critical("{} version {} is not compatible with {} version {} !  ", m_Source.getName(),
      m_Source.getVersion().to_string(), m_Destination.getName(), m_Destination.getVersion().to_string());
00079       log()->info("Compatible Interfaces for {} are", m_Destination.getName());
00080       for(std::map<std::string, std::string>::iterator it = m_Destination.getCompatibility().begin();
      it != m_Destination.getCompatibility().end(); ++it) { log()->info("{} version {}", it->first,
      it->second); }
00081       std::exit(-1);
00082     }
00083     if(!m_DetectorIDs.empty())
00084     {
00085       std::string ids;
00086       for(std::vector<DetectorID>::const_iterator it = m_DetectorIDs.cbegin(); it !=
      m_DetectorIDs.cend(); ++it) ids += std::to_string(static_cast<std::uint16_t>(*it)) + ";";
00087       log()->info("Detector ID(s) other than {} will be ignored", ids);
00088     }
00089     log()->info("*****************************************************************");
00090     RawBufferNavigator bufferNavigator(m_Logger);
00091     Timer              timer;
00092     timer.start();
00093     m_Source.start();
00094     m_Destination.start();
00095     while(m_Source.nextEvent() && m_NbrEventsToProcess >= m_NbrEvents)
00096     {
00098       m_Source.startEvent();
00099       m_Destination.startEvent();
00101
00102       m_Logger->warn("===*** Event {} ***===", m_NbrEvents);
00103       while(m_Source.nextDIFbuffer())
00104       {
00105         const Buffer& buffer = m_Source.getBuffer();
00106         bufferNavigator.setBuffer(buffer);
00107         if(std::find(m_DetectorIDs.begin(), m_DetectorIDs.end(),
      static_cast<DetectorID>(bufferNavigator.getDetectorID())) == m_DetectorIDs.end())
00108         {
00109           m_Logger->debug("Ignoring detector ID : {}", bufferNavigator.getDetectorID());
00110           continue;
00111         }
00112
00113         bit8_t* debug_variable_1 = buffer.end();
00114         bit8_t* debug_variable_2 = bufferNavigator.getDIFBuffer().end();
00115         if(debug_variable_1 != debug_variable_2) m_Logger->info("DIF BUFFER END {} {}",
      fmt::ptr(debug_variable_1), fmt::ptr(debug_variable_2));
00116         if(m_Debug) assert(debug_variable_1 == debug_variable_2);
00117
00118         std::int32_t idstart = bufferNavigator.getStartOfPayload();
00119         if(m_Debug && idstart == -1) m_Logger->info(to_hex(buffer));
00120         c.DIFStarter[idstart]++;
00121         if(!bufferNavigator.validBuffer())
00122         {
00123           m_Logger->error("!bufferNavigator.validBuffer()");
00124           continue;
00125         }
00126
00128         m_Source.startDIF();
00129         m_Destination.startDIF();
00131
00132         DIFPtr& d = bufferNavigator.getDIFPtr();
00133         c.DIFPtrValueAtReturnedPos[bufferNavigator.getDIFBufferStart()[d.getGetFramePtrReturn()]]++;
00134         if(m_Debug) assert(bufferNavigator.getDIFBufferStart()[d.getGetFramePtrReturn()] == 0xa0);
00135         c.SizeAfterDIFPtr[bufferNavigator.getSizeAfterDIFPtr()]++;
00136         m_Destination.processDIF(d);
00137         for(std::size_t i = 0; i < d.getNumberOfFrames(); ++i)
00138         {
00140           m_Source.startFrame();
00141           m_Destination.startFrame();
00143           m_Destination.processFrame(d, i);
00144           for(std::size_t j = 0; j < DU::NUMBER_PAD; ++j)
00145           {
00146             if(d.getThresholdStatus(i, j) != 0)
00147             {
00148               m_Source.startPad();
00149               m_Destination.startPad();
00150               m_Destination.processPadInFrame(d, i, j);
00151               m_Source.endPad();
00152               m_Destination.endPad();
00153             }
```

```
00154           }
00156           m_Source.endFrame();
00157           m_Destination.endFrame();
00159         }
00160
00161         bool processSC = false;
00162         if(bufferNavigator.hasSlowControlData())
00163         {
00164           c.hasSlowControl++;
00165           processSC = true;
00166         }
00167         if(bufferNavigator.badSCData())
00168         {
00169           c.hasBadSlowControl++;
00170           processSC = false;
00171         }
00172         if(processSC) { m_Destination.processSlowControl(bufferNavigator.getSCBuffer()); }
00173
00174         Buffer eod = bufferNavigator.getEndOfAllData();
00175         c.SizeAfterAllData[eod.size()]++;
00176         bit8_t* debug_variable_3 = eod.end();
00177         if(debug_variable_1 != debug_variable_3) m_Logger->info("END DATA BUFFER END {} {}",
      fmt::ptr(debug_variable_1), fmt::ptr(debug_variable_3));
00178         if(m_Debug) assert(debug_variable_1 == debug_variable_3);
00179         if(eod.size() != 0) m_Logger->info("End of Data remaining stuff :  {}", to_hex(eod));
00180
00181         int nonzeroCount = 0;
00182         for(bit8_t* it = eod.begin(); it != eod.end(); it++)
00183           if(static_cast<int>(*it) != 0) nonzeroCount++;
00184         c.NonZeroValusAtEndOfData[nonzeroCount]++;
00186         m_Source.endDIF();
00187         m_Destination.endDIF();
00189       }  // end of DIF while loop
00190       m_Logger->warn("===*** Event {} ***===", m_NbrEvents);
00191       m_NbrEvents++;
00193       m_Source.endEvent();
00194       m_Destination.endEvent();
00196     }  // end of event while loop
00197     m_Destination.end();
00198     m_Source.end();
00199     timer.stop();
00200     fmt::print("=== elapsed time {}ms ({}ms/event) ===\n", timer.getElapsedTime() / 1000,
      timer.getElapsedTime() / (1000 * m_NbrEvents));
00201   }
00202   void                              printAllCounters() { c.printAllCounters(); }
00203   std::shared_ptr<spdlog::logger> log() { return m_Logger; }
00204
00205   void setDetectorIDs(const std::vector<DetectorID>& detectorIDs) { m_DetectorIDs = detectorIDs; }
00206
00207 private:
00208   std::vector<DetectorID>         m_DetectorIDs;
00209   std::shared_ptr<spdlog::logger> m_Logger{nullptr};
00210   std::vector<spdlog::sink_ptr>   m_Sinks;
00211   BufferLooperCounter             c;
00212   SOURCE&                         m_Source{nullptr};
00213   DESTINATION&                    m_Destination{nullptr};
00214   bool                            m_Debug{false};
00215   std::uint32_t                   m_NbrEvents{1};
00216 };
```

5.7 libs/core/include/BufferLooperCounter.h File Reference

```
#include <map>
#include <memory>
#include <string>
```

Classes

- struct BufferLooperCounter

5.7.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file BufferLooperCounter.h.

5.8 BufferLooperCounter.h

Go to the documentation of this file.

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

5.9 libs/core/include/DetectorId.h File Reference

```
#include <cstdint>
```

Enumerations

- enum class DetectorID : std::uint16_t { HARDROC = 100 , HARDROC_NEW = 150 , RUNHEADER = 255 }

5.9.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file DetectorId.h.

5.9.2 Enumeration Type Documentation

5.9.2.1 DetectorID `enum class DetectorID : std::uint16_t [strong]`

Enumerator

HARDROC	
HARDROC_NEW	
RUNHEADER	

Definition at line 9 of file DetectorId.h.

```
00010 {
00011   HARDROC     = 100,
00012   HARDROC_NEW = 150,
00013   RUNHEADER   = 255
00014 };
```

5.10 DetectorId.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include <cstdint>
00008
00009 enum class DetectorID :  std::uint16_t
00010 {
00011   HARDROC     = 100,
00012   HARDROC_NEW = 150,
00013   RUNHEADER   = 255
00014 };
```

5.11 libs/core/include/DIFPtr.h File Reference

```
#include "Bits.h"
#include "Formatters.h"
#include "Utilities.h"
#include "Words.h"
#include <cstdint>
#include <iostream>
#include <spdlog/spdlog.h>
#include <string>
#include <vector>
```

Classes

- class DIFPtr

5.11.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file DIFPtr.h.

5.12 DIFPtr.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include "Bits.h"
00008 #include "Formatters.h"
00009 #include "Utilities.h"
00010 #include "Words.h"
00011
00012 #include <cstdint>
00013 #include <iostream>
00014 #include <spdlog/spdlog.h>
00015 #include <string>
00016 #include <vector>
00017
00018 class DIFPtr
00019 {
00020 public:
00021   void                 setBuffer(unsigned char*, const std::uint32_t&);
00022   bit8_t*              getPtr() const;
00023   std::uint32_t        getGetFramePtrReturn() const;
00024   std::vector<bit8_t*>& getFramesVector();
00025   std::vector<bit8_t*>& getLinesVector();
00026   std::uint32_t        getID() const;
00027   std::uint32_t        getDTC() const;
00028   std::uint32_t        getGTC() const;
00029   std::uint64_t        getAbsoluteBCID() const;
00030   std::uint32_t        getBCID() const;
00031   std::uint32_t        getLines() const;
00032   bool                 hasLine(const std::uint32_t&) const;
00033   std::uint32_t        getTASU1() const;
00034   std::uint32_t        getTASU2() const;
00035   std::uint32_t        getTDIF() const;
00036   float                getTemperatureDIF() const;
00037   float                getTemperatureASU1() const;
00038   float                getTemperatureASU2() const;
00039   bool                 hasTemperature() const;
00040   bool                 hasAnalogReadout() const;
00041   std::uint32_t        getNumberOfFrames() const;
00042   bit8_t*              getFramePtr(const std::uint32_t&) const;
00043   std::uint32_t        getFrameAsicHeader(const std::uint32_t&) const;
00044   std::uint32_t        getFrameBCID(const std::uint32_t&) const;
00045   std::uint32_t        getFrameTimeToTrigger(const std::uint32_t&) const;
00046   bool                 getFrameLevel(const std::uint32_t&, const std::uint32_t&, const
      std::uint32_t&) const;
00047   // Addition by GG
00048   std::uint32_t        getDIFid() const;
00049   std::uint32_t        getASICid(const std::uint32_t&) const;
00050   std::uint32_t        getThresholdStatus(const std::uint32_t&, const std::uint32_t&) const;
00051
00052 private:
00053   std::uint32_t       getAnalogPtr(const std::uint32_t& idx = 0);
00054   std::uint32_t       getFrameAsicHeaderInternal(const unsigned char* framePtr) const;
00055   std::uint32_t       getFramePtr();
00056   std::uint32_t       theSize_{0};
00057   std::uint32_t       theGetFramePtrReturn_{0};
00058   bit8_t*             theDIF_{nullptr};
00059   std::vector<bit8_t*> theFrames_;
00060   std::vector<bit8_t*> theLines_;
00061 };
00062
00063 inline std::uint32_t DIFPtr::getFrameAsicHeaderInternal(const bit8_t* framePtr)const { return
      (framePtr[DU::FRAME_ASIC_HEADER_SHIFT]); }
00064
00065 inline void DIFPtr::setBuffer(bit8_t* p, const std::uint32_t& max_size)
00066 {
00067   theFrames_.clear();
00068   theLines_.clear();
00069   theSize_ = max_size;
00070   theDIF_  = p;
00071   try
00072   {
00073     theGetFramePtrReturn_ = getFramePtr();
00074   }
00075   catch(const std::string& e)
00076   {
00077     spdlog::get("streamout")->error(" DIF {} T ? {} {}", getID(), hasTemperature(), e);
00078   }
00079 }
00080
00081 inline bit8_t* DIFPtr::getPtr()const { return theDIF_; }
00082
00083 inline std::uint32_t DIFPtr::getGetFramePtrReturn()const { return theGetFramePtrReturn_; }
00084
```

```
00085 inline std::vector<bit8_t*>& DIFPtr::getFramesVector() { return theFrames_; }
00086
00087 inline std::vector<bit8_t*>& DIFPtr::getLinesVector() { return theLines_; }
00088
00089 inline std::uint32_t DIFPtr::getID()const { return theDIF_[DU::ID_SHIFT]; }
00090
00091 inline std::uint32_t DIFPtr::getDTC()const { return (theDIF_[DU::DTC_SHIFT] << 24) +
      (theDIF_[DU::DTC_SHIFT + 1] << 16) + (theDIF_[DU::DTC_SHIFT + 2] << 8) + theDIF_[DU::DTC_SHIFT + 3]; }
00092
00093 inline std::uint32_t DIFPtr::getGTC()const { return (theDIF_[DU::GTC_SHIFT] << 24) +
      (theDIF_[DU::GTC_SHIFT + 1] << 16) + (theDIF_[DU::GTC_SHIFT + 2] << 8) + theDIF_[DU::GTC_SHIFT + 3]; }
00094
00095 inline std::uint64_t DIFPtr::getAbsoluteBCID()const
00096 {
00097   std::uint64_t LBC = ((theDIF_[DU::ABCID_SHIFT] << 16) | (theDIF_[DU::ABCID_SHIFT + 1] << 8) |
      (theDIF_[DU::ABCID_SHIFT + 2])) * 16777216ULL /* to shift the value from the 24 first bits*/
00098                     + ((theDIF_[DU::ABCID_SHIFT + 3] << 16) | (theDIF_[DU::ABCID_SHIFT + 4] << 8) |
      (theDIF_[DU::ABCID_SHIFT + 5]));
00099   return LBC;
00100 }
00101
00102 inline std::uint32_t DIFPtr::getBCID()const { return (theDIF_[DU::BCID_SHIFT] << 16) +
      (theDIF_[DU::BCID_SHIFT + 1] << 8) + theDIF_[DU::BCID_SHIFT + 2]; }
00103
00104 inline std::uint32_t DIFPtr::getLines()const { return (theDIF_[DU::LINES_SHIFT] >> 4) & 0x5; }
00105
00106 inline bool DIFPtr::hasLine(const std::uint32_t& line)const { return ((theDIF_[DU::LINES_SHIFT] >>
      line) & 0x1); }
00107
00108 inline std::uint32_t DIFPtr::getTASU1()const { return (theDIF_[DU::TASU1_SHIFT] << 24) +
      (theDIF_[DU::TASU1_SHIFT + 1] << 16) + (theDIF_[DU::TASU1_SHIFT + 2] << 8) + theDIF_[DU::TASU1_SHIFT +
      3]; }
00109
00110 inline std::uint32_t DIFPtr::getTASU2()const { return (theDIF_[DU::TASU2_SHIFT] << 24) +
      (theDIF_[DU::TASU2_SHIFT + 1] << 16) + (theDIF_[DU::TASU2_SHIFT + 2] << 8) + theDIF_[DU::TASU2_SHIFT +
      3]; }
00111
00112 inline std::uint32_t DIFPtr::getTDIF()const { return theDIF_[DU::TDIF_SHIFT]; }
00113
00114 inline float DIFPtr::getTemperatureDIF()const { return 0.508 * getTDIF() - 9.659; }
00115
00116 inline float DIFPtr::getTemperatureASU1()const { return (getTASU1() >> 3) * 0.0625; }
00117
00118 inline float DIFPtr::getTemperatureASU2()const { return (getTASU2() >> 3) * 0.0625; }
00119
00120 inline bool DIFPtr::hasTemperature()const { return (theDIF_[0] == DU::START_OF_DIF_TEMP); }
00121
00122 inline bool DIFPtr::hasAnalogReadout()const { return getLines() != 0; }
00123
00124 inline std::uint32_t DIFPtr::getNumberOfFrames()const { return theFrames_.size(); }
00125
00126 inline bit8_t* DIFPtr::getFramePtr(const std::uint32_t& i)const { return theFrames_[i]; }
00127
00128 inline std::uint32_t DIFPtr::getFrameAsicHeader(const std::uint32_t& i)const { return
      getFrameAsicHeaderInternal(theFrames_[i]); }
00129
00130 inline std::uint32_t DIFPtr::getFrameBCID(const std::uint32_t& i)const { return
      GrayToBin((theFrames_[i][DU::FRAME_BCID_SHIFT] << 16) + (theFrames_[i][DU::FRAME_BCID_SHIFT + 1] << 8) +
      theFrames_[i][DU::FRAME_BCID_SHIFT + 2]); }
00131
00132 inline std::uint32_t DIFPtr::getFrameTimeToTrigger(const std::uint32_t& i)const { return getBCID() -
      getFrameBCID(i); }
00133
00134 inline bool DIFPtr::getFrameLevel(const std::uint32_t& i, const std::uint32_t& ipad, const
      std::uint32_t& ilevel)const
00135 {
00136   return ((theFrames_[i][DU::FRAME_DATA_SHIFT + ((3 - ipad / 16) * 4 + (ipad % 16) / 4)] >> (7 -
      (((ipad % 16) % 4) * 2 + ilevel))) & 0x1);
00137 }
00138 // Addition by GG
00139 inline uint32_t DIFPtr::getDIFid()const { return getID() & 0xFF; }
00140
00141 inline uint32_t DIFPtr::getASICid(const std::uint32_t& i)const { return getFrameAsicHeader(i) & 0xFF;
      }
00142
00143 inline uint32_t DIFPtr::getThresholdStatus(const std::uint32_t& i, const std::uint32_t& ipad)const {
      return (((std::uint32_t)getFrameLevel(i, ipad, 1)) << 1) | ((std::uint32_t)getFrameLevel(i, ipad, 0));
      }
00144
00145 inline std::uint32_t DIFPtr::getFramePtr()
00146 {
00147   std::uint32_t fshift{0};
00148   if(DATA_FORMAT_VERSION >= 13)
00149   {
00150     fshift = DU::LINES_SHIFT + 1;
00151     if(hasTemperature()) fshift = DU::TDIF_SHIFT + 1;       // jenlev 1
00152     if(hasAnalogReadout()) fshift = getAnalogPtr(fshift);  // to be implemented
```

```
00153   }
00154   else
00155     fshift = DU::BCID_SHIFT + 3;
00156   if(theDIF_[fshift] != DU::START_OF_FRAME)
00157   {
00158     std::cout « "This is not a start of frame " « to_hex(theDIF_[fshift]) « " \n";
00159     return fshift;
00160   }
00161   do {
00162     if(theDIF_[fshift] == DU::END_OF_DIF) return fshift;
00163     if(theDIF_[fshift] == DU::START_OF_FRAME) fshift++;
00164     if(theDIF_[fshift] == DU::END_OF_FRAME)
00165     {
00166       fshift++;
00167       continue;
00168     }
00169     std::uint32_t header = getFrameAsicHeaderInternal(&theDIF_[fshift]);
00170     if(header == DU::END_OF_FRAME) return (fshift + 2);
00171     if(header < 1 || header > 48) { throw header + " Header problem " + fshift; }
00172     theFrames_.push_back(&theDIF_[fshift]);
00173     fshift += DU::FRAME_SIZE;
00174     if(fshift > theSize_)
00175     {
00176       std::cout « "fshift " « fshift « " exceed " « theSize_ « "\n";
00177       return fshift;
00178     }
00179     if(theDIF_[fshift] == DU::END_OF_FRAME) fshift++;
00180   } while(true);
00181 }
00182
00183 inline std::uint32_t DIFPtr::getAnalogPtr(const std::uint32_t& idx)
00184 {
00185   std::uint32_t fshift{idx};
00186   if(theDIF_[fshift] != DU::START_OF_LINES) return fshift;
00187   fshift++;
00188   while(theDIF_[fshift] != DU::END_OF_LINES)
00189   {
00190     theLines_.push_back(&theDIF_[fshift]);
00191     std::uint32_t nchip{theDIF_[fshift]};
00192     fshift += 1 + nchip * 64 * 2;
00193   }
00194   return fshift++;
00195 }
```

5.13 libs/core/include/DIFSlowControl.h File Reference

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

Classes

- class DIFSlowControl

Functions

- std::string to_string (const DIFSlowControl &c)

5.13.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file DIFSlowControl.h.

5.13.2 Function Documentation

5.13.2.1 to_string() std::string to_string (const DIFSlowControl & c)

Definition at line 256 of file DIFSlowControl.cc.

```
00257 {
00258   std::string ret;
00259   for(std::map<int, std::map<std::string, int»::const_iterator it = c.cbegin(); it != c.cend(); it++)
00260   {
00261     ret += "ASIC " + std::to_string(it->first) + " :\n";
00262     for(std::map<std::string, int>::const_iterator jt = (it->second).begin(); jt !=
      (it->second).end(); jt++) ret += jt->first + " :  " + std::to_string(jt->second) + "\n";
00263   }
00264   return ret;
00265 }
```

5.14 DIFSlowControl.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include <bitset>
00008 #include <cstdint>
00009 #include <iosfwd>
00010 #include <map>
00011 #include <string>
00012
00013 class DIFSlowControl
00014 {
00015 public:
00017
00022   DIFSlowControl(const std::uint8_t& version, const std::uint8_t& DIFid, unsigned char* buf);
00023
00025   std::uint8_t getDIFId();
00026
00028
00031   std::map<int, std::map<std::string, int» getChipsMap();
00032
00034
00038   std::map<std::string, int> getChipSlowControl(const int& asicid);
00039
00041
00045   int getChipSlowControl(const std::int8_t& asicid, const std::string& param);
00046
00047   std::map<int, std::map<std::string, int»::const_iterator cbegin()const { return m_MapSC.cbegin(); }
00048
00049   std::map<int, std::map<std::string, int»::const_iterator cend()const { return m_MapSC.cend(); }
00050
00051 private:
00053   DIFSlowControl() = delete;
00055   void FillHR1(const int& header_shift, unsigned char* cbuf);
00057   void FillHR2(const int& header_shift, unsigned char* cbuf);
00059   void FillAsicHR1(const std::bitset<72 * 8>& bs);
00061   void FillAsicHR2(const std::bitset<109 * 8>& bs);
00062
00063   unsigned int                              m_DIFId{0};
00064   unsigned int                              m_Version{0};
00065   unsigned int                              m_AsicType{0};  // asicType_
00066   unsigned int                              m_NbrAsic{0};
00067   std::map<int, std::map<std::string, int» m_MapSC;
00068 };
00069
00070 std::string to_string(const DIFSlowControl& c);
```

5.15 libs/core/include/Filesystem.h File Reference

```
#include <string>
```

Functions

- std::string path (const std::string &)
- std::string extension (const std::string &)
- std::string filename (const std::string &)

5.15.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Filesystem.h.

5.15.2 Function Documentation

5.15.2.1 extension()

```
std::string extension (
        const std::string & file )
```

Definition at line 13 of file Filesystem.cc.

```
00014 {
00015   std::size_t position = file.find_last_of(".");
00016   return (std::string::npos == position || position == 0) ?  "" :  file.substr(position + 1);
00017 }
```

5.15.2.2 filename()

```
std::string filename (
        const std::string & file )
```

Definition at line 19 of file Filesystem.cc.

```
00020 {
00021   std::size_t position = file.find_last_of(".");
00022   std::size_t pos      = file.find_last_of("\\/");
00023   return (std::string::npos == pos) ?  file.substr(0, position) :  file.substr(pos + 1, position - pos
      - 1);
00024 }
```

5.15.2.3 path()

```
std::string path (
        const std::string & file )
```

Definition at line 7 of file Filesystem.cc.

```
00008 {
00009   std::size_t pos = file.find_last_of("\\/");
00010   return (std::string::npos == pos) ?  "" :  file.substr(0, pos);
00011 }
```

5.16 Filesystem.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include <string>
00008
00009 std::string path(const std::string&);
00010 std::string extension(const std::string&);
00011 std::string filename(const std::string&);
```

5.17 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 &)

5.17.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Formatters.h.

5.17.2 Function Documentation

5.17.2.1 to_bin() [1/5] std::string to_bin (

```
const bit16_t & b )
```

Definition at line 71 of file Formatters.cc.

```
00071 { return fmt::format("{:#016b}", b); }
```

5.17.2.2 to_bin() [2/5] std::string to_bin (

```
const bit32_t & b )
```

Definition at line 73 of file Formatters.cc.

```
00073 { return fmt::format("{:#032b}", b); }
```

5.17.2.3 to_bin() [3/5] std::string to_bin (

```
const bit64_t & b )
```

Definition at line 75 of file Formatters.cc.

```
00075 { return fmt::format("{:#064b}", b); }
```

5.17.2.4 to_bin() [4/5] std::string to_bin (

```
const bit8_t & b )
```

Definition at line 69 of file Formatters.cc.

```
00069 { return fmt::format("{:#08b}", b); }
```

5.17.2.5 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 56 of file Formatters.cc.

```
00057 {
00058   std::size_t iend = end;
00059   if(iend == -1) iend = b.size();
00060   std::string ret;
00061   for(std::size_t k = begin; k < iend; k++)
00062   {
00063     ret += to_bin(b[k]);
00064     ret += " - ";
00065   }
00066   return ret;
00067 }
```

5.17.2.6 to_dec() [1/5] std::string to_dec (

```
const bit16_t & b )
```

Definition at line 29 of file Formatters.cc.

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

5.17.2.7 to_dec() [2/5] std::string to_dec (

```
const bit32_t & b )
```

Definition at line 31 of file Formatters.cc.

```
00031 { return fmt::format("{:#d}", b); }
```

5.17.2.8 to_dec() [3/5] std::string to_dec (

```
const bit64_t & b )
```

Definition at line 33 of file Formatters.cc.

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

5.17.2.9 to_dec() [4/5] std::string to_dec (

```
const bit8_t & b )
```

Definition at line 27 of file Formatters.cc.

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

5.17.2.10 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 14 of file Formatters.cc.

```
00015 {
00016   std::size_t iend = end;
00017   if(iend == -1) iend = b.size();
00018   std::string ret;
00019   for(std::size_t k = begin; k < iend; k++)
00020   {
00021     ret += to_dec(b[k]);
00022     ret += " - ";
00023   }
00024   return ret;
00025 }
```

5.17.2.11 to_hex() [1/5] std::string to_hex (

```
const bit16_t & b )
```

Definition at line 50 of file Formatters.cc.

```
00050 { return fmt::format("{:#04x}", b); }
```

5.17.2.12 to_hex() [2/5] std::string to_hex (

```
const bit32_t & b )
```

Definition at line 52 of file Formatters.cc.

```
00052 { return fmt::format("{:#08x}", b); }
```

5.17.2.13 to_hex() [3/5] std::string to_hex (

```
const bit64_t & b )
```

Definition at line 54 of file Formatters.cc.

```
00054 { return fmt::format("{:#016x}", b); }
```

5.17.2.14 to_hex() [4/5] std::string to_hex (

```
const bit8_t & b )
```

Definition at line 48 of file Formatters.cc.

```
00048 { return fmt::format("{:#02x}", b); }
```

5.17.2.15 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 35 of file Formatters.cc.

```
00036 {
00037   std::size_t iend = end;
00038   if(iend == -1) iend = b.size();
00039   std::string ret;
00040   for(std::size_t k = begin; k < iend; k++)
00041   {
00042     ret += to_hex(b[k]);
00043     ret += " - ";
00044   }
00045   return ret;
00046 }
```

5.17.2.16 to_oct() [1/5] std::string to_oct (

```
const bit16_t & b )
```

Definition at line 92 of file Formatters.cc.

```
00092 { return fmt::format("{:#08o}", b); }
```

5.17.2.17 to_oct() [2/5] std::string to_oct (

```
const bit32_t & b )
```

Definition at line 94 of file Formatters.cc.

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

5.17.2.18 to_oct() [3/5] std::string to_oct (

```
const bit64_t & b )
```

Definition at line 96 of file Formatters.cc.

```
00096 { return fmt::format("{:#032o}", b); }
```

5.17.2.19 to_oct() [4/5] std::string to_oct (

```
const bit8_t & b )
```

Definition at line 90 of file Formatters.cc.

```
00090 { return fmt::format("{:#04o}", b); }
```

5.17.2.20 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 77 of file Formatters.cc.

```
00078 {
00079   std::size_t iend = end;
00080   if(iend == -1) iend = b.size();
00081   std::string ret;
00082   for(std::size_t k = begin; k < iend; k++)
00083   {
00084     ret += to_oct(b[k]);
00085     ret += " - ";
00086   }
00087   return ret;
00088 }
```

5.18 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 class Buffer;
00013
00014 std::string to_dec(const Buffer& b, const std::size_t& begin = 0, const std::size_t& end = -1);
00015 std::string to_dec(const bit8_t&);
00016 std::string to_dec(const bit16_t&);
00017 std::string to_dec(const bit32_t&);
00018 std::string to_dec(const bit64_t&);
00019
00020 std::string to_hex(const Buffer& b, const std::size_t& begin = 0, const std::size_t& end = -1);
00021 std::string to_hex(const bit8_t&);
00022 std::string to_hex(const bit16_t&);
00023 std::string to_hex(const bit32_t&);
00024 std::string to_hex(const bit64_t&);
00025
00026 std::string to_bin(const Buffer& b, const std::size_t& begin = 0, const std::size_t& end = -1);
00027 std::string to_bin(const bit8_t&);
00028 std::string to_bin(const bit16_t&);
00029 std::string to_bin(const bit32_t&);
00030 std::string to_bin(const bit64_t&);
00031
00032 std::string to_oct(const Buffer& b, const std::size_t& begin = 0, const std::size_t& end = -1);
00033 std::string to_oct(const bit8_t&);
00034 std::string to_oct(const bit16_t&);
00035 std::string to_oct(const bit32_t&);
00036 std::string to_oct(const bit64_t&);
```

5.19 libs/core/include/Interface.h File Reference

```
#include "AppVersion.h"
#include "Buffer.h"
#include "Version.h"
```

```
#include <iostream>
#include <map>
#include <memory>
#include <semver.hpp>
#include <spdlog/logger.h>
#include <string>
```

Classes

- class Interface
- class InterfaceReader
- class InterfaceWriter

Enumerations

- enum class InterfaceType { Unknown = 0 , Reader = 1 , Writer = 2 }

  template class should implement void SOURCE::start(); bool SOURCE::next(); void SOURCE::end(); const Buffer& SOURCE::getBuffer();

5.19.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Interface.h.

5.19.2 Enumeration Type Documentation

5.19.2.1 InterfaceType `enum class InterfaceType [strong]`

template class should implement void SOURCE::start(); bool SOURCE::next(); void SOURCE::end(); const Buffer& SOURCE::getBuffer();

void DESTINATION::begin(); void DESTINATION::processDIF(const DIFPtr&); void DESTINATION::process↩ Frame(const DIFPtr&,const std::uint32_t& frameIndex); void DESTINATION::processPadInFrame(const DIFPtr&,const std::uint32_t& frameIndex,const std::uint32_t& channelIndex); void DESTINATION::processSlowControl(const Buffer&); void DESTINATION::end();

Enumerator

Unknown	
Reader	
Writer	

Definition at line 32 of file Interface.h.

```
00033 {
00034   Unknown = 0,
00035   Reader  = 1,
00036   Writer  = 2
00037 };
```

5.20 Interface.h

Go to the documentation of this file.

```
00001
00004 #pragma once
00005
00006 #include "AppVersion.h"
00007 #include "Buffer.h"
00008 #include "Version.h"
00009
00010 #include <iostream>
00011 #include <map>
00012 #include <memory>
00013 #include <semver.hpp>
00014 #include <spdlog/logger.h>
00015 #include <string>
00016
00032 enum class InterfaceType
00033 {
00034   Unknown = 0,
00035   Reader  = 1,
00036   Writer  = 2
00037 };
00038
00039 class Interface
00040 {
00041 public:
00042   Interface(const std::string& name, const std::string& version, const InterfaceType& type) :
      m_Name(name), m_Version(version) {}
00043   virtual ~Interface() = default;
00044   virtual void                      startEvent() {}
00045   virtual void                      endEvent() {}
00046   virtual void                      startDIF() {}
00047   virtual void                      endDIF() {}
00048   virtual void                      startFrame() {}
00049   virtual void                      endFrame() {}
00050   virtual void                      startPad() {}
00051   virtual void                      endPad() {}
00052   std::shared_ptr<spdlog::logger>& log() { return m_Logger; }
00053   void                              setLogger(const std::shared_ptr<spdlog::logger>& logger) { m_Logger
      = logger; }
00054   std::string                       getName() { return m_Name; }
00055   Version                           getVersion() { return m_Version; }
00056
00057 private:
00058   std::shared_ptr<spdlog::logger> m_Logger{nullptr};
00059   std::string                     m_Name;
00060   Version                         m_Version;
00061   InterfaceType                   m_Type{InterfaceType::Unknown};
00062 };
00063
00064 class InterfaceReader :  public Interface
00065 {
00066 public:
00067   InterfaceReader(const std::string& name, const std::string& version) :  Interface(name, version,
      InterfaceType::Reader) {}
00068   virtual ~InterfaceReader() = default;
00069
00070 protected:
00071   Buffer m_Buffer;
00072 };
00073
00074 class InterfaceWriter :  public Interface
00075 {
00076 public:
00077   InterfaceWriter(const std::string& name, const std::string& version) :  Interface(name, version,
      InterfaceType::Writer) {}
00078
00079   void addCompatibility(const std::string& name, const std::string& version) { m_Compatible[name] =
      version; }
00080
00081   std::map<std::string, std::string> getCompatibility() { return m_Compatible; }
00082
00083   bool checkCompatibility(const std::string& name, const std::string& version)
00084   {
00085     if(m_Compatible.find(name) != m_Compatible.end())
```

```
00086     {
00087       auto            ran = semver::range::detail::range(m_Compatible[name]);
00088       semver::version ver = semver::version(version);
00089       if(ran.satisfies(ver, false)) return true;
00090       else
00091         return false;
00092     }
00093     else
00094       return false;
00095   }
00096
00097   virtual ~InterfaceWriter() = default;
00098
00099 private:
00100   std::map<std::string, std::string> m_Compatible;
00101 };
```

5.21 libs/core/include/RawBufferNavigator.h File Reference

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

Classes

- class RawBufferNavigator

5.21.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file RawBufferNavigator.h.

5.22 RawBufferNavigator.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include "Buffer.h"
00008 #include "DIFPtr.h"
00009 #include "spdlog/spdlog.h"
00010
00011 #include <memory>
00012
00013 // class to navigate in the raw data buffer
00014 class RawBufferNavigator
00015 {
00016 public:
00017   explicit RawBufferNavigator(const std::shared_ptr<spdlog::logger>&);
00018   ~RawBufferNavigator() = default;
00019   explicit RawBufferNavigator(const Buffer& b);
00020   void          setBuffer(const Buffer& b);
00021   std::uint8_t  getDetectorID();
00022   bool          validBuffer();
00023   bit8_t*       getDIFBufferStart();
00024   std::uint32_t getDIFBufferSize();
00025   Buffer        getDIFBuffer();
00026   DIFPtr&       getDIFPtr();
00027   std::uint32_t getEndOfDIFData();
00028   std::uint32_t getSizeAfterDIFPtr();
```

```
00029   std::uint32_t getDIF_CRC();
00030   bool          hasSlowControlData();
00031   Buffer        getSCBuffer();
00032   bool          badSCData();
00033   Buffer        getEndOfAllData();
00034   static void   StartAt(const int& start);
00035   std::int32_t  getStartOfPayload();
00036
00037 private:
00038   std::shared_ptr<spdlog::logger> m_Logger{nullptr};
00039   void                            setSCBuffer();
00040   Buffer                          m_Buffer;
00041   Buffer                          m_SCbuffer;
00042   std::int32_t                    m_StartPayload{-1};
00043   DIFPtr                          m_TheDIFPtr;
00044   bool                            m_BadSCdata{false};
00045   static int                      m_Start;
00046 };
```

5.23 libs/core/include/Timer.h File Reference

```
#include <chrono>
```

Classes

- class Timer

5.23.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Timer.h.

5.24 Timer.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include <chrono>
00008
00009 class Timer
00010 {
00011 public:
00012   void  start() { m_StartTime = std::chrono::high_resolution_clock::now(); }
00013   void  stop() { m_StopTime = std::chrono::high_resolution_clock::now(); }
00014   float getElapsedTime() { return std::chrono::duration_cast<std::chrono::microseconds>(m_StopTime -
      m_StartTime).count(); }
00015
00016 private:
00017   std::chrono::time_point<std::chrono::high_resolution_clock> m_StartTime;
00018   std::chrono::time_point<std::chrono::high_resolution_clock> m_StopTime;
00019 };
```

5.25 libs/core/include/Utilities.h File Reference

```
#include <cstdint>
```

Functions

- std::uint64_t GrayToBin (const std::uint64_t &n)

5.25.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Utilities.h.

5.25.2 Function Documentation

5.25.2.1 GrayToBin() `std::uint64_t GrayToBin ( const std::uint64_t & n ) [inline]`

Definition at line 9 of file Utilities.h.

```
00010 {
00011   std::uint64_t ish{1};
00012   std::uint64_t anss{n};
00013   std::uint64_t idiv{0};
00014   std::uint64_t ishmax{sizeof(std::uint64_t) * 8};
00015   while(true)
00016   {
00017     idiv = anss » ish;
00018     anss ^= idiv;
00019     if(idiv <= 1 || ish == ishmax) return anss;
00020     ish «= 1;
00021   }
00022 }
```

5.26 Utilities.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include <cstdint>
00008
00009 inline std::uint64_t GrayToBin(const std::uint64_t& n)
00010 {
00011   std::uint64_t ish{1};
00012   std::uint64_t anss{n};
00013   std::uint64_t idiv{0};
00014   std::uint64_t ishmax{sizeof(std::uint64_t) * 8};
00015   while(true)
00016   {
00017     idiv = anss » ish;
00018     anss ^= idiv;
00019     if(idiv <= 1 || ish == ishmax) return anss;
00020     ish «= 1;
00021   }
00022 }
```

5.27 libs/core/include/Version.h File Reference

```
#include <cstdint>
#include <semver.hpp>
#include <string>
```

Classes

- class Version

5.27.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Version.h.

5.28 Version.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include <cstdint>
00008 #include <semver.hpp>
00009 #include <string>
00010
00011 class Version :  public semver::version
00012 {
00013 public:
00014   Version(const std::uint8_t& mj, const std::uint8_t& mn, const std::uint8_t& pt, const
      semver::prerelease& prt = semver::prerelease::none, const std::uint8_t& prn = 0) noexcept :
      semver::version(mj, mn, pt, prt, prn) {}
00015   explicit Version(const std::string_view& str) :  semver::version(str) {}
00016   Version() = default;
00017   std::uint8_t getMajor();
00018   std::uint8_t getMinor();
00019   std::uint8_t getPatch();
00020   std::string  getPreRelease();
00021   std::uint8_t getPreReleaseNumber();
00022 };
```

5.29 libs/core/include/Words.h File Reference

```
#include <cstdint>
```

Enumerations

- enum DU : std::uint8_t {
  START_OF_DIF = 0xB0 , START_OF_DIF_TEMP = 0xBB , END_OF_DIF = 0xA0 , START_OF_LINES = 0xC4 ,
  END_OF_LINES = 0xD4 , START_OF_FRAME = 0xB4 , END_OF_FRAME = 0xA3 , ID_SHIFT = 1 ,
  DTC_SHIFT = 2 , GTC_SHIFT = 10 , ABCID_SHIFT = 14 , BCID_SHIFT = 20 ,
  LINES_SHIFT = 23 , TASU1_SHIFT = 24 , TASU2_SHIFT = 28 , TDIF_SHIFT = 32 ,
  FRAME_ASIC_HEADER_SHIFT = 0 , FRAME_BCID_SHIFT = 1 , FRAME_DATA_SHIFT = 4 , FRAME_SIZE = 20 ,
  NUMBER_PAD = 64 }

5.29.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Words.h.

5.29.2 Enumeration Type Documentation

5.29.2.1 DU `enum DU : std::uint8_t`

Enumerator

START_OF_DIF	
START_OF_DIF_TEMP	
END_OF_DIF	
START_OF_LINES	
END_OF_LINES	
START_OF_FRAME	
END_OF_FRAME	
ID_SHIFT	
DTC_SHIFT	
GTC_SHIFT	
ABCID_SHIFT	
BCID_SHIFT	
LINES_SHIFT	
TASU1_SHIFT	
TASU2_SHIFT	
TDIF_SHIFT	
FRAME_ASIC_HEADER_SHIFT	
FRAME_BCID_SHIFT	
FRAME_DATA_SHIFT	
FRAME_SIZE	
NUMBER_PAD	

Definition at line 9 of file Words.h.

```
00010 {
00011   START_OF_DIF      = 0xB0,
00012   START_OF_DIF_TEMP = 0xBB,
00013   END_OF_DIF        = 0xA0,
00014   START_OF_LINES    = 0xC4,
00015   END_OF_LINES      = 0xD4,
00016
00017   START_OF_FRAME = 0xB4,
00018   END_OF_FRAME   = 0xA3,
00019
00020   ID_SHIFT    = 1,
00021   DTC_SHIFT   = 2,
00022   GTC_SHIFT   = 10,
00023   ABCID_SHIFT = 14,
00024   BCID_SHIFT  = 20,
00025   LINES_SHIFT = 23,
00026   TASU1_SHIFT = 24,
```

```
00027   TASU2_SHIFT = 28,
00028   TDIF_SHIFT  = 32,
00029
00030   FRAME_ASIC_HEADER_SHIFT = 0,
00031   FRAME_BCID_SHIFT        = 1,
00032   FRAME_DATA_SHIFT        = 4,
00033   FRAME_SIZE              = 20,
00034
00035   NUMBER_PAD = 64
00036 };
```

5.30 Words.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include <cstdint>
00008
00009 enum DU :  std::uint8_t
00010 {
00011   START_OF_DIF      = 0xB0,
00012   START_OF_DIF_TEMP = 0xBB,
00013   END_OF_DIF        = 0xA0,
00014   START_OF_LINES    = 0xC4,
00015   END_OF_LINES      = 0xD4,
00016
00017   START_OF_FRAME = 0xB4,
00018   END_OF_FRAME   = 0xA3,
00019
00020   ID_SHIFT    = 1,
00021   DTC_SHIFT   = 2,
00022   GTC_SHIFT   = 10,
00023   ABCID_SHIFT = 14,
00024   BCID_SHIFT  = 20,
00025   LINES_SHIFT = 23,
00026   TASU1_SHIFT = 24,
00027   TASU2_SHIFT = 28,
00028   TDIF_SHIFT  = 32,
00029
00030   FRAME_ASIC_HEADER_SHIFT = 0,
00031   FRAME_BCID_SHIFT        = 1,
00032   FRAME_DATA_SHIFT        = 4,
00033   FRAME_SIZE              = 20,
00034
00035   NUMBER_PAD = 64
00036 };
```

5.31 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.

5.31.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Bits.cc.

5.31.2 Function Documentation

5.31.2.1 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; }
```

5.32 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; }
```

5.33 libs/core/src/BufferLooperCounter.cc File Reference

```
#include "BufferLooperCounter.h"
#include <fmt/core.h>
```

5.34 BufferLooperCounter.cc

Go to the documentation of this file.

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

5.35 libs/core/src/DIFSlowControl.cc File Reference

```
#include "DIFSlowControl.h"
```

Functions

- std::string to_string (const DIFSlowControl &c)

5.35.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file DIFSlowControl.cc.

5.35.2 Function Documentation

5.35.2.1 to_string() `std::string to_string ( const DIFSlowControl & c )`

Definition at line 256 of file DIFSlowControl.cc.

```
00257 {
00258   std::string ret;
00259   for(std::map<int, std::map<std::string, int»::const_iterator it = c.cbegin(); it != c.cend(); it++)
00260   {
00261     ret += "ASIC " + std::to_string(it->first) + " :\n";
00262     for(std::map<std::string, int>::const_iterator jt = (it->second).begin(); jt !=
      (it->second).end(); jt++) ret += jt->first + " :  " + std::to_string(jt->second) + "\n";
00263   }
00264   return ret;
00265 }
```

5.36 DIFSlowControl.cc

Go to the documentation of this file.

```
00001
00005 #include "DIFSlowControl.h"
00006
00007 DIFSlowControl::DIFSlowControl(const std::uint8_t& version, const std::uint8_t& DIfId, unsigned char*
      cbuf) :  m_Version(version), m_DIFId(DIfId), m_AsicType(2)
00008 {
00009   if(cbuf[0] != 0xb1) return;
00010   int header_shift{6};
00011   if(m_Version < 8) m_NbrAsic = cbuf[5];
00012   else
00013   {
00014     m_DIFId      = cbuf[1];
00015     m_NbrAsic    = cbuf[2];
00016     header_shift = 3;
00017   }
00018   int size_hardroc1 = m_NbrAsic * 72 + header_shift + 1;
00019   if(cbuf[size_hardroc1 - 1] != 0xa1) size_hardroc1 = 0;
00020
00021   int size_hardroc2 = m_NbrAsic * 109 + header_shift + 1;
00022   if(cbuf[size_hardroc2 - 1] != 0xa1) size_hardroc2 = 0;
00023   if(size_hardroc1 != 0)
```

```
00024   {
00025     FillHR1(header_shift, cbuf);
00026     m_AsicType = 1;
00027   }
00028   else if(size_hardroc2 != 0)
00029     FillHR2(header_shift, cbuf);
00030   else
00031     return;
00032 }
00033
00034 inline std::uint8_t DIFSlowControl::getDIFId() { return m_DIFId; }
00035
00036 inline std::map<int, std::map<std::string, int» DIFSlowControl::getChipsMap() { return m_MapSC; }
00037
00038 inline std::map<std::string, int> DIFSlowControl::getChipSlowControl(const int& asicid) { return
      m_MapSC[asicid]; }
00039
00040 inline int DIFSlowControl::getChipSlowControl(const std::int8_t& asicid, const std::string& param) {
      return getChipSlowControl(asicid)[param]; }
00041
00042 void DIFSlowControl::FillHR1(const int& header_shift, unsigned char* cbuf)
00043 {
00044   int nasic{cbuf[header_shift - 1]};
00045   int idx{header_shift};
00046   for(int k = 0; k < nasic; k++)
00047   {
00048     std::bitset<72 * 8> bs;
00049     // printf("%x %x \n",cbuf[idx+k*72+69],cbuf[idx+k*72+70]);
00050     for(int l = 71; l >= 0; l--)
00051     {
00052       //  printf("%d %x :  %d -->",l,cbuf[idx+k*72+l],(71-l)*8);
00053       for(int m = 0; m < 8; m++)
00054       {
00055         if(((1 « m) & cbuf[idx + k * 72 + l]) != 0) bs.set((71 - l) * 8 + m, 1);
00056         else
00057           bs.set((71 - l) * 8 + m, 0);
00058         // printf("%d",(int) bs[(71-l)*8+m]);
00059       }
00060       // printf("\n");
00061     }
00062     FillAsicHR1(bs);
00063   }
00064 }
00065
00066 void DIFSlowControl::FillHR2(const int& header_shift, unsigned char* cbuf)
00067 {
00068   // int scsize1=cbuf[header_shift-1]*109+(header_shift-1)+2;
00069   int nasic{cbuf[header_shift - 1]};
00070   int idx{header_shift};
00071   // std::cout«" DIFSlowControl::FillHR nasic "«nasic«std::endl;
00072   for(int k = 0; k < nasic; k++)
00073   {
00074     std::bitset<109 * 8> bs;
00075     // printf("%x %x \n",cbuf[idx+k*109+69],cbuf[idx+k*109+70]);
00076     for(int l = 108; l >= 0; l--)
00077     {
00078       //  printf("%d %x :  %d -->",l,cbuf[idx+k*109+l],(71-l)*8);
00079       for(int m = 0; m < 8; m++)
00080       {
00081         if(((1 « m) & cbuf[idx + k * 109 + l]) != 0) bs.set((108 - l) * 8 + m, 1);
00082         else
00083           bs.set((108 - l) * 8 + m, 0);
00084         // printf("%d",(int) bs[(71-l)*8+m]);
00085       }
00086       // printf("\n");
00087     }
00088     FillAsicHR2(bs);
00089   }
00090 }
00091
00092 void DIFSlowControl::FillAsicHR1(const std::bitset<72 * 8>& bs)
00093 {
00094   // Asic Id
00095   int asicid{0};
00096   for(int j = 0; j < 8; j++)
00097     if(bs[j + 9] != 0) asicid += (1 « (7 - j));
00098   std::map<std::string, int> mAsic;
00099   // Slow Control
00100   mAsic["SSC0"]          = static_cast<int>(bs[575]);
00101   mAsic["SSC1"]          = static_cast<int>(bs[574]);
00102   mAsic["SSC2"]          = static_cast<int>(bs[573]);
00103   mAsic["Choix_caisson"] = static_cast<int>(bs[572]);
00104   mAsic["SW_50k"]        = static_cast<int>(bs[571]);
00105   mAsic["SW_100k"]       = static_cast<int>(bs[570]);
00106   mAsic["SW_100f"]       = static_cast<int>(bs[569]);
00107   mAsic["SW_50f"]        = static_cast<int>(bs[568]);
00108
```

```
00109   mAsic["Valid_DC"]  = static_cast<int>(bs[567]);
00110   mAsic["ON_Discri"] = static_cast<int>(bs[566]);
00111   mAsic["ON_Fsb"]    = static_cast<int>(bs[565]);
00112   mAsic["ON_Otaq"]   = static_cast<int>(bs[564]);
00113   mAsic["ON_W"]      = static_cast<int>(bs[563]);
00114   mAsic["ON_Ss"]     = static_cast<int>(bs[562]);
00115   mAsic["ON_Buf"]    = static_cast<int>(bs[561]);
00116   mAsic["ON_Paf"]    = static_cast<int>(bs[560]);
00117   // Gain
00118   for(int i = 0; i < 64; i++)
00119   {
00120     int gain{0};
00121     for(int j = 0; j < 6; j++)
00122       if(bs[176 + i * 6 + j] != 0) gain += (1 << j);
00123     mAsic["Channel_" + std::to_string(i) + "_" + "Gain"]       = gain;
00124     mAsic["Channel_" + std::to_string(i) + "_" + "cTest"]      = bs[112 + i];
00125     mAsic["Channel_" + std::to_string(i) + "_" + "Valid_trig"] = static_cast<int>(bs[25 + i]);
00126   }
00127
00128   mAsic["ON_Otabg"]  = static_cast<int>(bs[111]);
00129   mAsic["ON_Dac"]    = static_cast<int>(bs[110]);
00130   mAsic["ON_Otadac"] = static_cast<int>(bs[109]);
00131   // DAC
00132   int dac1{0};
00133   for(int j = 0; j < 10; j++)
00134     if(bs[j + 99] != 0) dac1 += (1 << j);
00135   mAsic["DAC1"] = dac1;
00136   int dac0{0};
00137   for(int j = 0; j < 10; j++)
00138     if(bs[j + 89] != 0) dac0 += (1 << j);
00139   mAsic["DAC0"]             = dac0;
00140   mAsic["EN_Raz_Ext"]       = static_cast<int>(bs[23]);
00141   mAsic["EN_Raz_Int"]       = static_cast<int>(bs[22]);
00142   mAsic["EN_Out_Raz_Int"]   = static_cast<int>(bs[21]);
00143   mAsic["EN_Trig_Ext"]      = static_cast<int>(bs[20]);
00144   mAsic["EN_Trig_Int"]      = static_cast<int>(bs[19]);
00145   mAsic["EN_Out_Trig_Int"]  = static_cast<int>(bs[18]);
00146   mAsic["Bypass_Chip"]      = static_cast<int>(bs[17]);
00147   mAsic["HardrocHeader"]    = static_cast<int>(asicid);
00148   mAsic["EN_Out_Discri"]    = static_cast<int>(bs[8]);
00149   mAsic["EN_Transmit_On"]   = static_cast<int>(bs[7]);
00150   mAsic["EN_Dout"]          = static_cast<int>(bs[6]);
00151   mAsic["EN_RamFull"]       = static_cast<int>(bs[5]);
00152   m_MapSC[asicid]           = mAsic;
00153 }
00154
00155 void DIFSlowControl::FillAsicHR2(const std::bitset<109 * 8>& bs)
00156 {
00157   int asicid{0};
00158   for(int j = 0; j < 8; j++)
00159     if(bs[j + (108 - 7) * 8 + 2] != 0) asicid += (1 << (7 - j));
00160   std::map<std::string, int> mAsic;
00161   for(int i = 0; i < 64; i++)
00162   {
00163     int gain{0};
00164     int mask{0};
00165     mAsic["Channel_" + std::to_string(i) + "_" + "cTest"] = bs[i];
00166     for(int j = 0; j < 8; j++)
00167       if(bs[64 + i * 8 + j] != 0) gain += (1 << j);
00168     mAsic["Channel_" + std::to_string(i) + "_" + "Gain"] = gain;
00169     for(int j = 0; j < 3; j++)
00170       if(bs[8 * 77 + 2 + i * 3 + j] != 0) mask += (1 << j);
00171     mAsic["Channel_" + std::to_string(i) + "_" + "Mask"] = mask;
00172   }
00173   mAsic["PwrOnPA"] = static_cast<int>(bs[8 * 72]);
00174   mAsic["Cmdb3SS"] = static_cast<int>(bs[8 * 72 + 1]);
00175   mAsic["Cmdb2SS"] = static_cast<int>(bs[8 * 72 + 2]);
00176   mAsic["Cmdb1SS"] = static_cast<int>(bs[8 * 72 + 3]);
00177   mAsic["Cmdb0SS"] = static_cast<int>(bs[8 * 72 + 4]);
00178   mAsic["SwSsc0"]  = static_cast<int>(bs[8 * 72 + 5]);
00179   mAsic["SwSsc1"]  = static_cast<int>(bs[8 * 72 + 6]);
00180   mAsic["SwSsc2"]  = static_cast<int>(bs[8 * 72 + 7]);
00181
00182   mAsic["PwrOnBuff"] = static_cast<int>(bs[8 * 73]);
00183   mAsic["PwrOnSS"]   = static_cast<int>(bs[8 * 73 + 1]);
00184   mAsic["PwrOnW"]    = static_cast<int>(bs[8 * 73 + 2]);
00185   mAsic["Cmdb3Fsb2"] = static_cast<int>(bs[8 * 73 + 3]);
00186   mAsic["Cmdb2Fsb2"] = static_cast<int>(bs[8 * 73 + 4]);
00187   mAsic["Cmdb1Fsb2"] = static_cast<int>(bs[8 * 73 + 5]);
00188   mAsic["Cmdb0Fsb2"] = static_cast<int>(bs[8 * 73 + 6]);
00189   mAsic["Sw50k2"]    = static_cast<int>(bs[8 * 73 + 7]);
00190
00191   mAsic["Sw100k2"]   = static_cast<int>(bs[8 * 74]);
00192   mAsic["Sw100f2"]   = static_cast<int>(bs[8 * 74 + 1]);
00193   mAsic["Sw50f2"]    = static_cast<int>(bs[8 * 74 + 2]);
00194   mAsic["Cmdb3Fsb1"] = static_cast<int>(bs[8 * 74 + 3]);
00195   mAsic["Cmdb2Fsb1"] = static_cast<int>(bs[8 * 74 + 4]);
```

```
00196   mAsic["Cmdb1Fsb1"] = static_cast<int>(bs[8 * 74 + 5]);
00197   mAsic["Cmdb0Fsb1"] = static_cast<int>(bs[8 * 74 + 6]);
00198   mAsic["Sw50k1"]    = static_cast<int>(bs[8 * 74 + 7]);
00199
00200   mAsic["Sw100k1"]   = static_cast<int>(bs[8 * 75]);
00201   mAsic["Sw100f1"]   = static_cast<int>(bs[8 * 75 + 1]);
00202   mAsic["Sw50f1"]    = static_cast<int>(bs[8 * 75 + 2]);
00203   mAsic["Sel0"]      = static_cast<int>(bs[8 * 75 + 3]);
00204   mAsic["Sel11"]     = static_cast<int>(bs[8 * 75 + 4]);
00205   mAsic["PwrOnFsb"]  = static_cast<int>(bs[8 * 75 + 5]);
00206   mAsic["PwrOnFsb1"] = static_cast<int>(bs[8 * 75 + 6]);
00207   mAsic["PwrOnFsb2"] = static_cast<int>(bs[8 * 75 + 7]);
00208
00209   mAsic["Sw50k0"]      = static_cast<int>(bs[8 * 76]);
00210   mAsic["Sw100k0"]     = static_cast<int>(bs[8 * 76 + 1]);
00211   mAsic["Sw100f0"]     = static_cast<int>(bs[8 * 76 + 2]);
00212   mAsic["Sw50f0"]      = static_cast<int>(bs[8 * 76 + 3]);
00213   mAsic["EnOtaQ"]      = static_cast<int>(bs[8 * 76 + 4]);
00214   mAsic["OtaQ_PwrADC"] = static_cast<int>(bs[8 * 76 + 5]);
00215   mAsic["Discri_PwrA"] = static_cast<int>(bs[8 * 76 + 6]);
00216   mAsic["Discri2"]     = static_cast<int>(bs[8 * 76 + 7]);
00217
00218   mAsic["Discri1"]      = static_cast<int>(bs[8 * 77]);
00219   mAsic["RS_or_Discri"] = static_cast<int>(bs[8 * 77 + 1]);
00220
00221   mAsic["Header"] = asicid;
00222   for(int i = 0; i < 3; i++)
00223   {
00224     int B = 0;
00225     for(int j = 0; j < 10; j++)
00226       if(bs[8 * 102 + 2 + i * 10 + j] != 0) B += (1 << j);
00227     mAsic["B" + std::to_string(i)] = B;
00228   }
00229
00230   mAsic["Smalldac"]  = static_cast<int>(bs[8 * 106]);
00231   mAsic["DacSw"]     = static_cast<int>(bs[8 * 106 + 1]);
00232   mAsic["OtagBgSw"]  = static_cast<int>(bs[8 * 106 + 2]);
00233   mAsic["Trig2b"]    = static_cast<int>(bs[8 * 106 + 3]);
00234   mAsic["Trig1b"]    = static_cast<int>(bs[8 * 106 + 4]);
00235   mAsic["Trig0b"]    = static_cast<int>(bs[8 * 106 + 5]);
00236   mAsic["EnTrigOut"] = static_cast<int>(bs[8 * 106 + 6]);
00237   mAsic["DiscrOrOr"] = static_cast<int>(bs[8 * 106 + 7]);
00238
00239   mAsic["TrigExtVal"]   = static_cast<int>(bs[8 * 107]);
00240   mAsic["RazChnIntVal"] = static_cast<int>(bs[8 * 107 + 1]);
00241   mAsic["RazChnExtVal"] = static_cast<int>(bs[8 * 107 + 2]);
00242   mAsic["ScOn"]         = static_cast<int>(bs[8 * 107 + 3]);
00243   mAsic["CLKMux"]       = static_cast<int>(bs[8 * 107 + 4]);
00244
00245   // EnOCDout1b   EnOCDout2b   EnOCTransmitOn1b   EnOCTransmitOn2b   EnOCChipsatb   SelStartReadout
    SelEndReadout
00246   mAsic["SelEndReadout"]    = static_cast<int>(bs[8 * 108 + 1]);
00247   mAsic["SelStartReadout"]  = static_cast<int>(bs[8 * 108 + 2]);
00248   mAsic["EnOCChipsatb"]     = static_cast<int>(bs[8 * 108 + 3]);
00249   mAsic["EnOCTransmitOn2b"] = static_cast<int>(bs[8 * 108 + 4]);
00250   mAsic["EnOCTransmitOn1b"] = static_cast<int>(bs[8 * 108 + 5]);
00251   mAsic["EnOCDout2b"]       = static_cast<int>(bs[8 * 108 + 6]);
00252   mAsic["EnOCDout1b"]       = static_cast<int>(bs[8 * 108 + 7]);
00253   m_MapSC[asicid]           = mAsic;
00254 }
00255
00256 std::string to_string(const DIFSlowControl& c)
00257 {
00258   std::string ret;
00259   for(std::map<int, std::map<std::string, int>>::const_iterator it = c.cbegin(); it != c.cend(); it++)
00260   {
00261     ret += "ASIC " + std::to_string(it->first) + " :\n";
00262     for(std::map<std::string, int>::const_iterator jt = (it->second).begin(); jt !=
    (it->second).end(); jt++) ret += jt->first + " :  " + std::to_string(jt->second) + "\n";
00263   }
00264   return ret;
00265 }
```

5.37 libs/core/src/Filesystem.cc File Reference

```
#include "Filesystem.h"
```

Functions

- std::string path (const std::string &file)

- std::string extension (const std::string &file)
- std::string filename (const std::string &file)

5.37.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Filesystem.cc.

5.37.2 Function Documentation

5.37.2.1 extension() `std::string extension (`

`const std::string & file )`

Definition at line 13 of file Filesystem.cc.

```
00014 {
00015   std::size_t position = file.find_last_of(".");
00016   return (std::string::npos == position || position == 0) ?  "" :  file.substr(position + 1);
00017 }
```

5.37.2.2 filename() `std::string filename (`

`const std::string & file )`

Definition at line 19 of file Filesystem.cc.

```
00020 {
00021   std::size_t position = file.find_last_of(".");
00022   std::size_t pos      = file.find_last_of("\\/");
00023   return (std::string::npos == pos) ?  file.substr(0, position) :  file.substr(pos + 1, position - pos
      - 1);
00024 }
```

5.37.2.3 path() `std::string path (`

`const std::string & file )`

Definition at line 7 of file Filesystem.cc.

```
00008 {
00009   std::size_t pos = file.find_last_of("\\/");
00010   return (std::string::npos == pos) ?  "" :  file.substr(0, pos);
00011 }
```

5.38 Filesystem.cc

Go to the documentation of this file.

```
00001
00005 #include "Filesystem.h"
00006
00007 std::string path(const std::string& file)
00008 {
00009   std::size_t pos = file.find_last_of("\\/");
00010   return (std::string::npos == pos) ?  "" :  file.substr(0, pos);
00011 }
00012
00013 std::string extension(const std::string& file)
00014 {
00015   std::size_t position = file.find_last_of(".");
00016   return (std::string::npos == position || position == 0) ?  "" :  file.substr(position + 1);
00017 }
00018
00019 std::string filename(const std::string& file)
00020 {
00021   std::size_t position = file.find_last_of(".");
00022   std::size_t pos      = file.find_last_of("\\/");
00023   return (std::string::npos == pos) ?  file.substr(0, position) :  file.substr(pos + 1, position - pos
      - 1);
00024 }
```

5.39 libs/core/src/Formatters.cc File Reference

```
#include "Formatters.h"
#include "Bits.h"
#include "Buffer.h"
#include "Words.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)

5.39.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Formatters.cc.

5.39.2 Function Documentation

5.39.2.1 to_bin() [1/5] std::string to_bin (const bit16_t & *b*)

Definition at line 71 of file Formatters.cc.

```
00071 { return fmt::format("{:#016b}", b); }
```

5.39.2.2 to_bin() [2/5] std::string to_bin (const bit32_t & *b*)

Definition at line 73 of file Formatters.cc.

```
00073 { return fmt::format("{:#032b}", b); }
```

5.39.2.3 to_bin() [3/5] std::string to_bin (const bit64_t & *b*)

Definition at line 75 of file Formatters.cc.

```
00075 { return fmt::format("{:#064b}", b); }
```

5.39.2.4 to_bin() [4/5] std::string to_bin (const bit8_t & *b*)

Definition at line 69 of file Formatters.cc.

```
00069 { return fmt::format("{:#08b}", b); }
```

5.39.2.5 to_bin() [5/5]

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

Definition at line 56 of file Formatters.cc.

```
00057 {
00058   std::size_t iend = end;
00059   if(iend == -1) iend = b.size();
00060   std::string ret;
00061   for(std::size_t k = begin; k < iend; k++)
00062   {
00063     ret += to_bin(b[k]);
00064     ret += " - ";
00065   }
00066   return ret;
00067 }
```

5.39.2.6 to_dec() [1/5]

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

Definition at line 29 of file Formatters.cc.

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

5.39.2.7 to_dec() [2/5]

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

Definition at line 31 of file Formatters.cc.

```
00031 { return fmt::format("{:#d}", b); }
```

5.39.2.8 to_dec() [3/5]

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

Definition at line 33 of file Formatters.cc.

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

5.39.2.9 to_dec() [4/5]

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

Definition at line 27 of file Formatters.cc.

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

5.39.2.10 to_dec() [5/5] std::string to_dec (

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

Definition at line 14 of file Formatters.cc.

```
00015 {
00016   std::size_t iend = end;
00017   if(iend == -1) iend = b.size();
00018   std::string ret;
00019   for(std::size_t k = begin; k < iend; k++)
00020   {
00021     ret += to_dec(b[k]);
00022     ret += " - ";
00023   }
00024   return ret;
00025 }
```

5.39.2.11 to_hex() [1/5] std::string to_hex (

```
const bit16_t & b )
```

Definition at line 50 of file Formatters.cc.

```
00050 { return fmt::format("{:#04x}", b); }
```

5.39.2.12 to_hex() [2/5] std::string to_hex (

```
const bit32_t & b )
```

Definition at line 52 of file Formatters.cc.

```
00052 { return fmt::format("{:#08x}", b); }
```

5.39.2.13 to_hex() [3/5] std::string to_hex (

```
const bit64_t & b )
```

Definition at line 54 of file Formatters.cc.

```
00054 { return fmt::format("{:#016x}", b); }
```

5.39.2.14 to_hex() [4/5] std::string to_hex (

```
const bit8_t & b )
```

Definition at line 48 of file Formatters.cc.

```
00048 { return fmt::format("{:#02x}", b); }
```

5.39.2.15 to_hex() [5/5] std::string to_hex (

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

Definition at line 35 of file Formatters.cc.

```
00036 {
00037   std::size_t iend = end;
00038   if(iend == -1) iend = b.size();
00039   std::string ret;
00040   for(std::size_t k = begin; k < iend; k++)
00041   {
00042     ret += to_hex(b[k]);
00043     ret += " - ";
00044   }
00045   return ret;
00046 }
```

5.39.2.16 to_oct() [1/5] std::string to_oct (

```
const bit16_t & b )
```

Definition at line 92 of file Formatters.cc.

```
00092 { return fmt::format("{:#08o}", b); }
```

5.39.2.17 to_oct() [2/5] std::string to_oct (

```
const bit32_t & b )
```

Definition at line 94 of file Formatters.cc.

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

5.39.2.18 to_oct() [3/5] std::string to_oct (

```
const bit64_t & b )
```

Definition at line 96 of file Formatters.cc.

```
00096 { return fmt::format("{:#032o}", b); }
```

5.39.2.19 to_oct() [4/5] std::string to_oct (

```
const bit8_t & b )
```

Definition at line 90 of file Formatters.cc.

```
00090 { return fmt::format("{:#04o}", b); }
```

5.39.2.20 to_oct() [5/5] std::string to_oct (

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

Definition at line 77 of file Formatters.cc.

```
00078 {
00079   std::size_t iend = end;
00080   if(iend == -1) iend = b.size();
00081   std::string ret;
00082   for(std::size_t k = begin; k < iend; k++)
00083   {
00084     ret += to_oct(b[k]);
00085     ret += " - ";
00086   }
00087   return ret;
00088 }
```

5.40 Formatters.cc

Go to the documentation of this file.

```
00001
00006 #include "Formatters.h"
00007
00008 #include "Bits.h"
00009 #include "Buffer.h"
00010 #include "Words.h"
00011
00012 #include <fmt/format.h>
00013
00014 std::string to_dec(const Buffer& b, const std::size_t& begin, const std::size_t& end)
00015 {
00016   std::size_t iend = end;
00017   if(iend == -1) iend = b.size();
00018   std::string ret;
00019   for(std::size_t k = begin; k < iend; k++)
00020   {
00021     ret += to_dec(b[k]);
00022     ret += " - ";
00023   }
00024   return ret;
00025 }
00026
00027 std::string to_dec(const bit8_t& b) { return fmt::format("{:#d}", b); }
00028
00029 std::string to_dec(const bit16_t& b) { return fmt::format("{:#d}", b); }
00030
00031 std::string to_dec(const bit32_t& b) { return fmt::format("{:#d}", b); }
00032
00033 std::string to_dec(const bit64_t& b) { return fmt::format("{:#d}", b); }
00034
00035 std::string to_hex(const Buffer& b, const std::size_t& begin, const std::size_t& end)
00036 {
00037   std::size_t iend = end;
00038   if(iend == -1) iend = b.size();
00039   std::string ret;
00040   for(std::size_t k = begin; k < iend; k++)
00041   {
00042     ret += to_hex(b[k]);
00043     ret += " - ";
00044   }
00045   return ret;
00046 }
00047
00048 std::string to_hex(const bit8_t& b) { return fmt::format("{:#02x}", b); }
00049
00050 std::string to_hex(const bit16_t& b) { return fmt::format("{:#04x}", b); }
00051
00052 std::string to_hex(const bit32_t& b) { return fmt::format("{:#08x}", b); }
00053
00054 std::string to_hex(const bit64_t& b) { return fmt::format("{:#016x}", b); }
00055
00056 std::string to_bin(const Buffer& b, const std::size_t& begin, const std::size_t& end)
00057 {
00058   std::size_t iend = end;
00059   if(iend == -1) iend = b.size();
00060   std::string ret;
00061   for(std::size_t k = begin; k < iend; k++)
00062   {
00063     ret += to_bin(b[k]);
```

```
00064     ret += " - ";
00065   }
00066   return ret;
00067 }
00068
00069 std::string to_bin(const bit8_t& b) { return fmt::format("{:#08b}", b); }
00070
00071 std::string to_bin(const bit16_t& b) { return fmt::format("{:#016b}", b); }
00072
00073 std::string to_bin(const bit32_t& b) { return fmt::format("{:#032b}", b); }
00074
00075 std::string to_bin(const bit64_t& b) { return fmt::format("{:#064b}", b); }
00076
00077 std::string to_oct(const Buffer& b, const std::size_t& begin, const std::size_t& end)
00078 {
00079   std::size_t iend = end;
00080   if(iend == -1) iend = b.size();
00081   std::string ret;
00082   for(std::size_t k = begin; k < iend; k++)
00083   {
00084     ret += to_oct(b[k]);
00085     ret += " - ";
00086   }
00087   return ret;
00088 }
00089
00090 std::string to_oct(const bit8_t& b) { return fmt::format("{:#04o}", b); }
00091
00092 std::string to_oct(const bit16_t& b) { return fmt::format("{:#08o}", b); }
00093
00094 std::string to_oct(const bit32_t& b) { return fmt::format("{:#016o}", b); }
00095
00096 std::string to_oct(const bit64_t& b) { return fmt::format("{:#032o}", b); }
```

5.41 libs/core/src/RawBufferNavigator.cc File Reference

```
#include "RawBufferNavigator.h"
#include "Words.h"
#include "spdlog/spdlog.h"
```

5.41.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file RawBufferNavigator.cc.

5.42 RawBufferNavigator.cc

Go to the documentation of this file.

```
00001
00005 #include "RawBufferNavigator.h"
00006
00007 #include "Words.h"
00008 #include "spdlog/spdlog.h"
00009
00010 RawBufferNavigator::RawBufferNavigator(const std::shared_ptr<spdlog::logger>& logger) { m_Logger =
      logger; }
00011
00012 std::int32_t RawBufferNavigator::getStartOfPayload()
00013 {
00014   for(std::size_t i = m_Start; i < m_Buffer.size(); i++)
00015   {
00016     if(m_Buffer[i] == DU::START_OF_DIF || m_Buffer[i] == DU::START_OF_DIF_TEMP)
00017     {
00018       m_StartPayload = i;
00019       return m_StartPayload;
```

```
00020     }
00021   }
00022   m_StartPayload = -1;
00023   return m_StartPayload;
00024 }
00025
00026 int RawBufferNavigator::m_Start = 92;
00027
00028 void RawBufferNavigator::StartAt(const int& start)
00029 {
00030   if(start >= 0) m_Start = start;
00031 }
00032
00033 void RawBufferNavigator::setBuffer(const Buffer& b)
00034 {
00035   m_BadSCdata = false;
00036   m_Buffer    = b;
00037   // m_DIFstartIndex = getStartOfPayload();
00038 }
00039
00040 RawBufferNavigator::RawBufferNavigator(const Buffer& b) :  m_Buffer(b) { setBuffer(b); }
00041
00042 std::uint8_t RawBufferNavigator::getDetectorID() { return m_Buffer[0]; }
00043
00044 bool RawBufferNavigator::validBuffer() { return m_StartPayload != -1; }
00045
00046 bit8_t* RawBufferNavigator::getDIFBufferStart() { return &(m_Buffer.begin()[m_StartPayload]); }
00047
00048 std::uint32_t RawBufferNavigator::getDIFBufferSize() { return m_Buffer.size() - m_StartPayload; }
00049
00050 Buffer RawBufferNavigator::getDIFBuffer() { return Buffer(getDIFBufferStart(), getDIFBufferSize()); }
00051
00052 DIFPtr& RawBufferNavigator::getDIFPtr()
00053 {
00054   m_TheDIFPtr.setBuffer(getDIFBufferStart(), getDIFBufferSize());
00055   return m_TheDIFPtr;
00056 }
00057
00058 std::uint32_t RawBufferNavigator::getEndOfDIFData() { return getDIFPtr().getGetFramePtrReturn() + 3; }
00059
00060 std::uint32_t RawBufferNavigator::getSizeAfterDIFPtr() { return getDIFBufferSize() -
      getDIFPtr().getGetFramePtrReturn(); }
00061
00062 std::uint32_t RawBufferNavigator::getDIF_CRC()
00063 {
00064   uint32_t i{getEndOfDIFData()};
00065   uint32_t ret{0};
00066   ret |= ((m_Buffer.begin()[i - 2]) << 8);
00067   ret |= m_Buffer.begin()[i - 1];
00068   return ret;
00069 }
00070
00071 bool RawBufferNavigator::hasSlowControlData() { return getDIFBufferStart()[getEndOfDIFData()] == 0xb1;
      }
00072
00073 Buffer RawBufferNavigator::getSCBuffer()
00074 {
00075   setSCBuffer();
00076   return m_SCbuffer;
00077 }
00078
00079 bool RawBufferNavigator::badSCData()
00080 {
00081   setSCBuffer();
00082   return m_BadSCdata;
00083 }
00084
00085 void RawBufferNavigator::setSCBuffer()
00086 {
00087   if(!hasSlowControlData()) return;
00088   if(m_SCbuffer.size() != 0) return;  // deja fait
00089   if(m_BadSCdata) return;
00090   m_SCbuffer.set(&(getDIFBufferStart()[getEndOfDIFData()]));
00091   // compute Slow Control size
00092   std::size_t maxsize{m_Buffer.size() - m_StartPayload - getEndOfDIFData() + 1};  // should I +1 here
      ?
00093   uint32_t    k{1};                                                              // SC Header
00094   uint32_t    dif_ID{m_SCbuffer[1]};
00095   uint32_t    chipSize{m_SCbuffer[3]};
00096   while((dif_ID != 0xa1 && m_SCbuffer[k] != 0xa1 && k < maxsize) || (dif_ID == 0xa1 && m_SCbuffer[k +
      2] == chipSize && k < maxsize))
00097   {
00098     k += 2;  // DIF ID + ASIC Header
00099     uint32_t scsize = m_SCbuffer[k];
00100     if(scsize != 74 && scsize != 109)
00101     {
00102       m_Logger->error("PROBLEM WITH SC SIZE {}", scsize);
```

```
00103       k           = 0;
00104       m_BadSCdata = true;
00105       break;
00106     }
00107     k++;          // skip size bit
00108     k += scsize;  // skip the data
00109   }
00110   if(m_SCbuffer[k] == 0xa1 && !m_BadSCdata) m_SCbuffer.setSize(k + 1);  // add the trailer
00111   else
00112   {
00113     m_BadSCdata = true;
00114     m_Logger->error("PROBLEM SC TRAILER NOT FOUND ");
00115   }
00116 }
00117
00118 Buffer RawBufferNavigator::getEndOfAllData()
00119 {
00120   setSCBuffer();
00121   if(hasSlowControlData() && !m_BadSCdata) { return Buffer(&(m_SCbuffer.begin()[m_SCbuffer.size()]),
      getSizeAfterDIFPtr() - 3 - m_SCbuffer.size()); }
00122   else
00123     return Buffer(&(getDIFBufferStart()[getEndOfDIFData()]), getSizeAfterDIFPtr() - 3);  // remove the
      2 bytes for CRC and the DIF trailer
00124 }
```

5.43 libs/core/src/Version.cc File Reference

```
#include "Version.h"
```

5.43.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Version.cc.

5.44 Version.cc

Go to the documentation of this file.

```
00001
00005 #include "Version.h"
00006
00007 const static Version streamout_version;
00008
00009 std::uint8_t Version::getMajor() { return major; }
00010
00011 std::uint8_t Version::getMinor() { return minor; }
00012
00013 std::uint8_t Version::getPatch() { return patch; }
00014
00015 std::string Version::getPreRelease()
00016 {
00017   switch(prerelease_type)
00018   {
00019     case semver::prerelease::alpha:  return "alpha";
00020     case semver::prerelease::beta:  return "beta";
00021     case semver::prerelease::rc:  return "rc";
00022     case semver::prerelease::none:  return "";
00023     default:  return "";
00024   }
00025 }
00026
00027 std::uint8_t Version::getPreReleaseNumber() { return prerelease_number; }
```

5.45 libs/interface/Dump/include/textDump.h File Reference

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

Classes

- class textDump

5.45.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file textDump.h.

5.46 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 <spdlog/logger.h>
00013
00014 class textDump :  public InterfaceWriter
00015 {
00016 public:
00017   textDump();
00018   void                              start();
00019   void                              processDIF(const DIFPtr&);
00020   void                              processFrame(const DIFPtr&, uint32_t frameIndex);
00021   void                              processPadInFrame(const DIFPtr&, uint32_t frameIndex, uint32_t
      channelIndex);
00022   void                              processSlowControl(Buffer);
00023   void                              end();
00024   std::shared_ptr<spdlog::logger>& print() { return m_InternalLogger; }
00025   void                              setLevel(const spdlog::level::level_enum& level) {
      m_InternalLogger->set_level(level); }
00026
00027 private:
00028   // This class is a dumb class to print on terminal so we need the logger + the standard one given by
      the interface.
00029   std::shared_ptr<spdlog::logger> m_InternalLogger{nullptr};
00030 };
```

5.47 libs/interface/Dump/src/textDump.cc File Reference

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

5.47.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file textDump.cc.

5.48 textDump.cc

Go to the documentation of this file.

```
00001
00005 #include "textDump.h"
00006
00007 #include "DIFPtr.h"
00008
00009 textDump::textDump() :  InterfaceWriter("textDump", "1.0.0")
00010 {
00011   m_InternalLogger = std::make_shared<spdlog::logger>("textDump",
      std::make_shared<spdlog::sinks::stdout_color_sink_mt>());
00012   m_InternalLogger->set_level(spdlog::level::trace);
00013   addCompatibility("RawdataReader", ">=1.0.0");
00014   addCompatibility("DIFdataExample", ">=1.0.0");
00015 }
00016
00017 void textDump::start() { print()->info("Will dump bunch of DIF data"); }
00018
00019 void textDump::processDIF(const DIFPtr& d) { print()->info("DIF_ID : {}, DTC : {}, GTC : {}, DIF BCID
      {}, Absolute BCID : {}, Nbr frames {}", d.getDIFid(), d.getDTC(), d.getGTC(), d.getBCID(),
      d.getAbsoluteBCID(), d.getNumberOfFrames()); }
00020
00021 void textDump::processFrame(const DIFPtr& d, uint32_t frameIndex)
00022 {
00023   print()->info("\tDisplaying frame number {} :  ASIC ID {}, Frame BCID {}, Frame Time To Trigger
      (a.k.a timestamp) is {}", frameIndex, d.getASICid(frameIndex), d.getFrameBCID(frameIndex),
      d.getFrameTimeToTrigger(frameIndex));
00024 }
00025
00026 void textDump::processPadInFrame(const DIFPtr& d, uint32_t frameIndex, uint32_t channelIndex)
00027 {
00028   if(d.getThresholdStatus(frameIndex, channelIndex) > 0) { print()->info("\t\tChannel {}, Threshold
      {}", channelIndex, d.getThresholdStatus(frameIndex, channelIndex)); }
00029 }
00030
00031 void textDump::processSlowControl(Buffer) { print()->error("textDump::processSlowControl not
      implemented yet."); }
00032
00033 void textDump::end() { print()->info("textDump end of report"); }
```

5.49 libs/interface/LCIO/include/LCIOWriter.h File Reference

5.49.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file LCIOWriter.h.

5.50 LCIOWriter.h

Go to the documentation of this file.

```
00001
00005 #pragma once
```

5.51 libs/interface/LCIO/src/LCIOWriter.cc File Reference

5.51.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file LCIOWriter.cc.

5.52 LCIOWriter.cc

Go to the documentation of this file.
00001

5.53 libs/interface/RawDataReader/include/RawdataReader.h File Reference

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

Classes

- class RawdataReader

5.53.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file RawdataReader.h.

5.54 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 InterfaceReader
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   const Buffer& getBuffer();
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   std::string        m_Filename;
00042 };
```

5.55 libs/interface/RawDataReader/src/RawdataReader.cc File Reference

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

5.55.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file RawdataReader.cc.

5.56 RawdataReader.cc

Go to the documentation of this file.

```
00001
00004 #include "RawdataReader.h"
00005
00006 #include <cstdint>
00007 #include <cstring>
00008 #include <stdexcept>
00009 #include <zlib.h>
00010
00012 std::size_t RawdataReader::m_BufferSize = 0x100000;
00013
00014 void RawdataReader::setDefaultBufferSize(const std::size_t& size) { m_BufferSize = size; }
00015
00016 RawdataReader::RawdataReader(const char* fileName) :  InterfaceReader("RawdataReader", "1.0.0")
00017 {
00018   m_buf.reserve(m_BufferSize);
00019   m_Filename = fileName;
00020 }
00021
00022 void RawdataReader::start() { openFile(m_Filename); }
00023
00024 void RawdataReader::end() { closeFile(); }
00025
00026 void RawdataReader::uncompress()
00027 {
00028   static const std::size_t size_buffer{0x20000};
00029   std::size_t              shift{3 * sizeof(std::uint32_t) + sizeof(std::uint64_t)};
00030   static bit8_t            obuf[size_buffer];
00031   unsigned long            size_buffer_end{0x20000};  // NOLINT(runtime/int)
00032   std::int8_t              rc = ::uncompress(obuf, &size_buffer_end, &m_Buffer[shift], m_Buffer.size()
      - shift);
00033   switch(rc)
00034   {
00035     case Z_OK: break;
00036     default:  throw "decompress error"; break;
00037   }
00038   memcpy(&m_Buffer[shift], obuf, size_buffer_end);
00039   m_Buffer.setSize(size_buffer_end + shift);
00040 }
00041
00042 void RawdataReader::closeFile()
00043 {
00044   try
00045   {
00046     if(m_FileStream.is_open()) m_FileStream.close();
00047   }
00048   catch(const std::ios_base::failure& e)
00049   {
00050     log()->error("Caught an ios_base::failure in closeFile :  {} {}", e.what(), e.code().value());
00051     throw;
00052   }
00053 }
00054
00055 void RawdataReader::openFile(const std::string& fileName)
00056 {
00057   try
00058   {
00059     m_FileStream.rdbuf()->pubsetbuf(0, 0);
00060     m_FileStream.exceptions(std::ifstream::failbit | std::ifstream::badbit);
00061     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
00062     m_FileStream.rdbuf()->pubsetbuf(0, 0);
00063     if(m_FileStream.is_open())
00064     {
00065       setFileSize(m_FileStream.tellg());
00066       m_FileStream.seekg(0, std::ios::beg);
00067     }
00068   }
00069   catch(const std::ios_base::failure& e)
00070   {
00071     log()->error("Caught an ios_base::failure in openFile :  {}", e.what());
00072     throw;
00073   }
00074 }
00075
00076 bool RawdataReader::nextEvent()
00077 {
00078   try
00079   {
00080     m_FileStream.read(reinterpret_cast<char*>(&m_EventNumber), sizeof(std::uint32_t));
00081     m_FileStream.read(reinterpret_cast<char*>(&m_NumberOfDIF), sizeof(std::uint32_t));
00082   }
00083   catch(const std::ios_base::failure& e)
00084   {
```

```
00085     return false;
00086   }
00087   return true;
00088 }
00089
00090 bool RawdataReader::nextDIFbuffer()
00091 {
00092   try
00093   {
00094     static int DIF_processed{0};
00095     if(DIF_processed >= m_NumberOfDIF)
00096     {
00097       DIF_processed = 0;
00098       return false;
00099     }
00100     else
00101     {
00102       DIF_processed++;
00103       std::uint32_t bsize{0};
00104       m_FileStream.read(reinterpret_cast<char*>(&bsize), sizeof(std::uint32_t));
00105       m_FileStream.read(reinterpret_cast<char*>(&m_buf[0]), bsize);
00106       m_Buffer = Buffer(m_buf);
00107     }
00108   }
00109   catch(const std::ios_base::failure& e)
00110   {
00111     log()->error("Caught an ios_base::failure in openFile :  {}", e.what());
00112     return false;
00113   }
00114   return true;
00115 }
00116
00117 const Buffer& RawdataReader::getBuffer()
00118 {
00119   uncompress();
00120   return m_Buffer;
00121 }
00122
00123 void RawdataReader::setFileSize(const std::size_t& size) { m_FileSize = size; }
00124
00125 float RawdataReader::getFileSize() { return m_FileSize; }
```

5.57 libs/interface/ROOT/include/DIF.h File Reference

```
#include "Hit.h"
#include <TObject.h>
#include <cstdint>
#include <map>
#include <vector>
```

Classes

- class DIF

Typedefs

- using Hits_const_iterator = std::vector< Hit >::const_iterator

5.57.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file DIF.h.

5.57.2 Typedef Documentation

5.57.2.1 Hits_const_iterator `using Hits_const_iterator = std::vector<Hit>::const_iterator`

Definition at line 14 of file DIF.h.

5.58 DIF.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include "Hit.h"
00008
00009 #include <TObject.h>
00010 #include <cstdint>
00011 #include <map>
00012 #include <vector>
00013
00014 using Hits_const_iterator = std::vector<Hit>::const_iterator;
00015
00016 class DIF :  public TObject
00017 {
00018 public:
00019   void                              clear();
00020   void                              addHit(const Hit&);
00021   void                              setID(const std::uint8_t&);
00022   std::uint8_t                      getID() const;
00023   void                              setDTC(const std::uint32_t&);
00024   std::uint32_t                     getDTC() const;
00025   void                              setGTC(const std::uint32_t&);
00026   std::uint32_t                     getGTC() const;
00027   void                              setDIFBCID(const std::uint32_t&);
00028   std::uint32_t                     getDIFBCID() const;
00029   void                              setAbsoluteBCID(const std::uint64_t&);
00030   std::uint64_t                     getAbsoluteBCID() const;
00031   std::vector<Hit>::const_iterator cbegin() const;
00032   std::vector<Hit>::const_iterator cend() const;
00033
00034 private:
00035   std::uint8_t     m_ID{0};
00036   std::uint32_t    m_DTC{0};
00037   std::uint32_t    m_GTC{0};
00038   std::uint32_t    m_DIFBCID{0};
00039   std::uint64_t    m_AbsoluteBCID{0};
00040   std::vector<Hit> m_Hits;
00041   ClassDef(DIF, 1);
00042 };
```

5.59 libs/interface/ROOT/include/DIFLinkDef.h File Reference

`#include <vector>`

5.59.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file DIFLinkDef.h.

5.60 DIFLinkDef.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006 #include <vector>
00007
00008 #ifdef __CLING__
00009 #pragma link C++ class DIF;
00010 #pragma link C++ class Hit;
00011 #pragma link C++ class std::vector < Hit>;
00012 #endif
```

5.61 libs/interface/ROOT/include/Event.h File Reference

```
#include "DIF.h"
#include <TObject.h>
#include <cstdint>
#include <map>
```

Classes

- class Event

Typedefs

- using DIFs_const_iterator = std::map< std::uint8_t, DIF >::const_iterator

5.61.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Event.h.

5.61.2 Typedef Documentation

5.61.2.1 DIFs_const_iterator `using DIFs_const_iterator = std::map<std::uint8_t, DIF>::const_iterator`

Definition at line 13 of file Event.h.

5.62 Event.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include "DIF.h"
00008
00009 #include <TObject.h>
00010 #include <cstdint>
00011 #include <map>
00012
00013 using DIFs_const_iterator = std::map<std::uint8_t, DIF>::const_iterator;
00014
00015 class Event :  public TObject
00016 {
00017 public:
00018   void                                        clear();
00019   void                                        addDIF(const DIF& dif);
00020   std::map<std::uint8_t, DIF>::const_iterator cbegin() const;
00021   std::map<std::uint8_t, DIF>::const_iterator cend() const;
00022
00023 private:
00024   std::map<std::uint8_t, DIF> DIFs;
00025   ClassDef(Event, 1);
00026 };
```

5.63 libs/interface/ROOT/include/EventLinkDef.h File Reference

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

5.63.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file EventLinkDef.h.

5.64 EventLinkDef.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006 #include <cstdint>
00007 #include <map>
00008 #include <vector>
00009 #ifdef __CLING__
00010 #pragma link C++ class DIF;
00011 #pragma link C++ class std::vector < DIF>;
00012 #pragma link C++ class Hit;
00013 #pragma link C++ class std::vector < Hit>;
00014 #pragma link C++ class Event;
00015 #pragma link C++ class std::vector < Event>;
00016 #pragma link C++ class std::map < std::uint8_t, DIF>;
00017 #endif
```

5.65 libs/interface/ROOT/include/Hit.h File Reference

```
#include <TObject.h>
#include <cstdint>
```

Classes

- class Hit

5.65.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Hit.h.

5.66 Hit.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include <TObject.h>
00008 #include <cstdint>
00009
00010 class Hit :  public TObject
00011 {
00012 public:
00013   void          clear();
00014   void          setDIF(const std::uint8_t&);
00015   void          setASIC(const std::uint8_t&);
00016   void          setChannel(const std::uint8_t&);
00017   void          setThreshold(const std::uint8_t&);
00018   void          setDTC(const std::uint32_t&);
00019   void          setGTC(const std::uint32_t&);
00020   void          setDIFBCID(const std::uint32_t&);
00021   void          setFrameBCID(const std::uint32_t&);
00022   void          setTimestamp(const std::uint32_t&);
00023   void          setAbsoluteBCID(const std::uint64_t&);
00024   std::uint8_t  getDIFid() const;
00025   std::uint8_t  getASICid() const;
00026   std::uint8_t  getChannel() const;
00027   std::uint8_t  getThreshold() const;
00028   std::uint32_t getDTC() const;
00029   std::uint32_t getGTC() const;
00030   std::uint32_t getDIFBCID() const;
00031   std::uint32_t getFrameBCID() const;
00032   std::uint32_t getTimestamp() const;
00033   std::uint64_t getAbsoluteBCID() const;
00034
00035 private:
00036   std::uint8_t  m_DIF{0};
00037   std::uint8_t  m_ASIC{0};
00038   std::uint8_t  m_Channel{0};
00039   std::uint8_t  m_Threshold{0};
00040   std::uint32_t m_DTC{0};
00041   std::uint32_t m_GTC{0};
00042   std::uint32_t m_DIFBCID{0};
00043   std::uint32_t m_FrameBCID{0};
00044   std::uint32_t m_Timestamp{0};
00045   std::uint64_t m_AbsoluteBCID{0};
00046   ClassDef(Hit, 1);
00047 };
```

5.67 libs/interface/ROOT/include/HitLinkDef.h File Reference

5.67.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file HitLinkDef.h.

5.68 HitLinkDef.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006 #ifdef __CLING__
00007 #pragma link C++ class Hit;
00008 #endif
```

5.69 libs/interface/ROOT/include/ROOTWriter.h File Reference

```
#include "Buffer.h"
#include "DIFPtr.h"
#include "Event.h"
#include "Interface.h"
#include <TFile.h>
#include <TTree.h>
#include <string>
#include <vector>
```

Classes

- class ROOTWriter

5.70 ROOTWriter.h

Go to the documentation of this file.

```
00001
00006 #pragma once
00007
00008 #include "Buffer.h"
00009 #include "DIFPtr.h"
00010 #include "Event.h"
00011 #include "Interface.h"
00012
00013 #include <TFile.h>
00014 #include <TTree.h>
00015 #include <string>
00016 #include <vector>
00017
00018 class ROOTWriter :  public InterfaceWriter
00019 {
00020 public:
00021   ROOTWriter();
00022
00023   void setFilename(const std::string&);
00024
00025   void start();
00026   void processDIF(const DIFPtr&);
00027   void processFrame(const DIFPtr&, const std::uint32_t& frameIndex);
00028   void processPadInFrame(const DIFPtr&, const std::uint32_t& frameIndex, const std::uint32_t&
      channelIndex);
00029   void processSlowControl(const Buffer&) { ; }
00030   void end();
00031
00032   virtual void startEvent();
00033   virtual void endEvent();
00034   virtual void startDIF();
00035   virtual void endDIF();
00036   virtual void startFrame();
00037   virtual void endFrame();
00038   virtual void startPad();
00039   virtual void endPad();
00040
00041 private:
00042   TFile*      m_File{nullptr};
00043   TTree*      m_Tree{nullptr};
00044   Event*      m_Event{nullptr};
00045   DIF*        m_DIF{nullptr};
00046   Hit*        m_Hit{nullptr};
00047   std::string m_Filename;
00048 };
```

5.71 libs/interface/ROOT/src/DIF.cc File Reference

```
#include "DIF.h"
#include <cstdint>
```

5.71.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file DIF.cc.

5.72 DIF.cc

Go to the documentation of this file.

```
00001
00006 #include "DIF.h"
00007
00008 #include <cstdint>
00009
00010 void DIF::addHit(const Hit& hit) { m_Hits.push_back(hit); }
00011
00012 void DIF::setID(const std::uint8_t& id) { m_ID = id; }
00013
00014 std::uint8_t DIF::getID()const { return m_ID; }
00015
00016 void DIF::setDTC(const std::uint32_t& dtc) { m_DTC = dtc; }
00017
00018 std::uint32_t DIF::getDTC()const { return m_DTC; }
00019
00020 void DIF::setGTC(const std::uint32_t& gtc) { m_GTC = gtc; }
00021
00022 std::uint32_t DIF::getGTC()const { return m_GTC; }
00023
00024 void DIF::setDIFBCID(const std::uint32_t& difbcid) { m_DIFBCID = difbcid; }
00025
00026 std::uint32_t DIF::getDIFBCID()const { return m_DIFBCID; }
00027
00028 void DIF::setAbsoluteBCID(const std::uint64_t& absolutebcid) { m_AbsoluteBCID = absolutebcid; }
00029
00030 std::uint64_t DIF::getAbsoluteBCID()const { return m_AbsoluteBCID; }
00031
00032 std::vector<Hit>::const_iterator DIF::cbegin()const { return m_Hits.cbegin(); }
00033
00034 std::vector<Hit>::const_iterator DIF::cend()const { return m_Hits.cend(); }
00035
00036 void DIF::clear() { m_Hits.clear(); }
```

5.73 libs/interface/ROOT/src/Event.cc File Reference

```
#include "Event.h"
```

5.73.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Event.cc.

5.74 Event.cc

Go to the documentation of this file.

```
00001
00006 #include "Event.h"
00007
00008 void Event::clear() { DIFs.clear(); }
00009
00010 void Event::addDIF(const DIF& dif) { DIFs[dif.getID()] = dif; }
00011
00012 std::map<std::uint8_t, DIF>::const_iterator Event::cbegin()const { return DIFs.cbegin(); }
00013
00014 std::map<std::uint8_t, DIF>::const_iterator Event::cend()const { return DIFs.cend(); }
```

5.75 libs/interface/ROOT/src/Hit.cc File Reference

```
#include "Hit.h"
```

5.75.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Hit.cc.

5.76 Hit.cc

Go to the documentation of this file.

```
00001
00006 #include "Hit.h"
00007 void Hit::clear()
00008 {
00009   m_DIF          = 0;
00010   m_ASIC         = 0;
00011   m_Channel      = 0;
00012   m_Threshold    = 0;
00013   m_DTC          = 0;
00014   m_GTC          = 0;
00015   m_DIFBCID      = 0;
00016   m_FrameBCID    = 0;
00017   m_Timestamp    = 0;
00018   m_AbsoluteBCID = 0;
00019 }
00020
00021 void Hit::setDIF(const std::uint8_t& dif) { m_DIF = dif; }
00022
00023 void Hit::setASIC(const std::uint8_t& asic) { m_ASIC = asic; }
00024
00025 void Hit::setChannel(const std::uint8_t& channel) { m_Channel = channel; }
00026
00027 void Hit::setThreshold(const std::uint8_t& threshold) { m_Threshold = threshold; }
00028
00029 void Hit::setDTC(const std::uint32_t& dtc) { m_DTC = dtc; }
00030
00031 void Hit::setGTC(const std::uint32_t& gtc) { m_GTC = gtc; }
00032
00033 void Hit::setDIFBCID(const std::uint32_t& difbcid) { m_DIFBCID = difbcid; }
00034
00035 void Hit::setFrameBCID(const std::uint32_t& framebcid) { m_FrameBCID = framebcid; }
00036
00037 void Hit::setTimestamp(const std::uint32_t& timestamp) { m_Timestamp = timestamp; }
00038
00039 void Hit::setAbsoluteBCID(const std::uint64_t& absolutebcid) { m_AbsoluteBCID = absolutebcid; }
00040
00041 std::uint8_t Hit::getDIFid()const { return m_DIF; }
00042
```

```
00043 std::uint8_t Hit::getASICid()const { return m_ASIC; }
00044
00045 std::uint8_t Hit::getChannel()const { return m_Channel; }
00046
00047 std::uint8_t Hit::getThreshold()const { return m_Threshold; }
00048
00049 std::uint32_t Hit::getDTC()const { return m_DTC; }
00050
00051 std::uint32_t Hit::getGTC()const { return m_GTC; }
00052
00053 std::uint32_t Hit::getDIFBCID()const { return m_DIFBCID; }
00054
00055 std::uint32_t Hit::getFrameBCID()const { return m_FrameBCID; }
00056
00057 std::uint32_t Hit::getTimestamp()const { return m_Timestamp; }
00058
00059 std::uint64_t Hit::getAbsoluteBCID()const { return m_AbsoluteBCID; }
```

5.77 libs/interface/ROOT/src/ROOTWriter.cc File Reference

```
#include "ROOTWriter.h"
```

5.77.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file ROOTWriter.cc.

5.78 ROOTWriter.cc

Go to the documentation of this file.

```
00001
00006 #include "ROOTWriter.h"
00007
00008 void ROOTWriter::setFilename(const std::string& filename) { m_Filename = filename; }
00009
00010 ROOTWriter::ROOTWriter() :  InterfaceWriter("ROOTWriter", "1.0.0") { addCompatibility("RawdataReader",
      ">=1.0.0"); }
00011
00012 void ROOTWriter::start()
00013 {
00014   m_File = TFile::Open(m_Filename.c_str(), "RECREATE", m_Filename.c_str(),
      ROOT::CompressionSettings(ROOT::kZLIB, 5));
00015   m_Tree = new TTree("RawData", "Raw SDHCAL data tree");
00016   m_Tree->Branch("Events", &m_Event, 512000, 99);
00017 }
00018
00019 void ROOTWriter::end()
00020 {
00021   if(m_Tree) m_Tree->Write();
00022   if(m_File)
00023   {
00024     m_File->Write();
00025     m_File->Close();
00026   }
00027   if(m_File) delete m_File;
00028 }
00029
00030 void ROOTWriter::processDIF(const DIFPtr& d)
00031 {
00032   m_DIF->setID(d.getDIFid());
00033   m_DIF->setDTC(d.getDTC());
00034   m_DIF->setGTC(d.getGTC());
00035   m_DIF->setDIFBCID(d.getBCID());
00036   m_DIF->setAbsoluteBCID(d.getAbsoluteBCID());
00037 }
00038
```

```
00039 void ROOTWriter::processFrame(const DIFPtr& d, const std::uint32_t& frameIndex)
00040 {
00041   m_Hit->setDIF(d.getDIFid());
00042   m_Hit->setASIC(d.getASICid(frameIndex));
00043   m_Hit->setDTC(d.getDTC());
00044   m_Hit->setGTC(d.getGTC());
00045   m_Hit->setDIFBCID(d.getBCID());
00046   m_Hit->setAbsoluteBCID(d.getAbsoluteBCID());
00047   m_Hit->setFrameBCID(d.getFrameBCID(frameIndex));
00048   m_Hit->setTimestamp(d.getFrameTimeToTrigger(frameIndex));
00049 }
00050
00051 void ROOTWriter::processPadInFrame(const DIFPtr& d, const std::uint32_t& frameIndex, const
      std::uint32_t& channelIndex)
00052 {
00053   m_Hit->setChannel(channelIndex);
00054   m_Hit->setThreshold(static_cast<std::uint8_t>(d.getThresholdStatus(frameIndex, channelIndex)));
00055 }
00056
00057 void ROOTWriter::startEvent()
00058 {
00059   m_Event = new Event();
00060   // m_Event->clear();
00061 }
00062
00063 void ROOTWriter::endEvent()
00064 {
00065   m_Tree->Fill();
00066   if(m_Event) delete m_Event;
00067 }
00068
00069 void ROOTWriter::startDIF()
00070 {
00071   m_DIF = new DIF();
00072   // m_DIF->clear();
00073 }
00074
00075 void ROOTWriter::endDIF()
00076 {
00077   m_Event->addDIF(*m_DIF);
00078   delete m_DIF;
00079 }
00080
00081 void ROOTWriter::startFrame()
00082 {
00083   m_Hit = new Hit();
00084   // m_Hit->clear();
00085 }
00086
00087 void ROOTWriter::endFrame()
00088 {
00089   m_DIF->addHit(*m_Hit);
00090   delete m_Hit;
00091 }
00092
00093 void ROOTWriter::startPad() {}
00094
00095 void ROOTWriter::endPad() {}
```