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

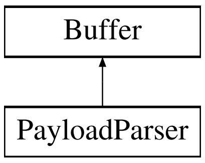
Here is a list of all files with brief descriptions:

4 Class Documentation

4.1 Buffer Class Reference

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

Inheritance diagram for Buffer:



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
- bool empty ()
- void set (unsigned char ∗b)
- void set (const Buffer &buffer)
- 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 35 of file Buffer.h.

```
00035 { 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 empty() `bool Buffer::empty ( ) [inline]`

Definition at line 27 of file Buffer.h.

```
00027 { return m_Size == 0; }
```

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

Definition at line 36 of file Buffer.h.

```
00036 { return m_Buffer + m_Size; }
```

4.1.3.5 operator[]() [1/2] `bit8_t & Buffer::operator[] ( const std::size_t & pos ) [inline]`

Definition at line 37 of file Buffer.h.

```
00037 { return m_Buffer[pos]; }
```

4.1.3.6 operator[]() [2/2] `bit8_t & Buffer::operator[] ( const std::size_t & pos ) const [inline]`

Definition at line 38 of file Buffer.h.

```
00038 { return m_Buffer[pos]; }
```

4.1.3.7 set() [1/2] `void Buffer::set ( const Buffer & buffer ) [inline]`

Definition at line 29 of file Buffer.h.

```
00030   {
00031     m_Buffer   = buffer.begin();
00032     m_Size     = buffer.size();
00033     m_Capacity = buffer.capacity();
00034   }
```

4.1.3.8 set() [2/2]

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

Definition at line 28 of file Buffer.h.

```
00028 { m_Buffer = b; }
```

4.1.3.9 setSize()

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

Definition at line 40 of file Buffer.h.

```
00040 { m_Size = size; }
```

4.1.3.10 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 28 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 31 of file BufferLooper.h.

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

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 39 of file BufferLooper.h.

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

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

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

Definition at line 223 of file BufferLooper.h.

```
{ 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 48 of file BufferLooper.h.

```
  {
    // clang-format off
    fmt::print(fg(fmt::color::medium_orchid) | fmt::emphasis::bold,
            "\n"
" SSSSSSSSSSSSSSS      tttt
tttt\n"
"SS:::::::::::::::S  ttt:::t
ttt:::t\n"
"S:::::SSSSSS::::::S  t:::::t
t:::::t\n"
"S:::::S     SSSSSSS  t:::::t
t:::::t\n"
"S:::::S        ttttttt:::::ttttttt   rrrrr   rrrrrrrrr       eeeeeeeeeeee    aaaaaaaaaaaaa
mmmmmmm    mmmmmmm      ooooooooooo   uuuuuu    uuuuuuttttttt:::::ttttttt\n"
"S:::::S        t:::::::::::::::::t   r::::rrr:::::::::r    ee::::::::::::ee  a::::::::::::a
mm:::::::m  m:::::::mm  oo:::::::::::oo u::::u    u::::ut:::::::::::::::::t\n"
" S::::SSSS     t:::::::::::::::::t   r:::::::::::::::::r  e::::::eeeee:::::eeaaaaaaaaa:::::a
m::::::::::mm::::::::::mo:::::::::::::::ou::::u    u::::ut:::::::::::::::::t\n"
"  SS::::::SSSSStttttt:::::::tttttt   rr::::::rrrrr::::::re::::::e     e:::::e         a::::a
m::::::::::::::::::::::mo::::ooooo::::ou::::u    u::::utttttt:::::::tttttt\n"
"    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"
"       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"
"             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"
"             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    tttttt\n"
"SSSSSSS     S:::::S  t::::::tttt:::::tr:::::r           e::::::::e        a::::a    a:::::a m::::m
m::::m   m::::mo:::::ooooo:::::ou:::::::::::::::::uu    t::::::tttt:::::t\n"
"S::::::SSSSSS:::::S  tt::::::::::::::tr:::::r            e::::::::eeeeeeeea:::::aaaa::::::a m::::m
m::::m   m::::mo:::::::::::::::o u::::::::::::::::u    tt::::::::::::::t\n"
"S:::::::::::::::SS     tt:::::::::::ttr:::::r             ee:::::::::::::e a::::::::::aa:::am::::m
m::::m   m::::m oo:::::::::::oo   uu::::::::uu:::u      tt:::::::::::tt\n"
" SSSSSSSSSSSSSSS          ttttttttttt  rrrrrrr               eeeeeeeeeeeeee  aaaaaaaaaa  aaaammmmmm
mmmmmm   mmmmmm   ooooooooooo        uuuuuuuu  uuuu        ttttttttttt {}\n"
"\n",
fmt::format(fg(fmt::color::red) | fmt::emphasis::bold, "v{}", streamout_version.to_string()));
    // clang-format on
    log()->info("*********************************************************************");
    log()->info("Streamout Version :  {}", streamout_version.to_string());
    log()->info("Using InterfaceReader {} version {}", m_Source.getName(),
m_Source.getVersion().to_string());
    log()->info("Using InterfaceWriter {} version {}", m_Destination.getName(),
m_Destination.getVersion().to_string());

    if(!m_Destination.checkCompatibility(m_Source.getName(), m_Source.getVersion().to_string()))
    {
      log()->critical("{} version {} is not compatible with {} version {} !  ", m_Source.getName(),
m_Source.getVersion().to_string(), m_Destination.getName(), m_Destination.getVersion().to_string());
      log()->info("Compatible Interfaces for {} are", m_Destination.getName());
      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); }
      std::exit(-1);
    }
    if(!m_DetectorIDs.empty())
    {
      std::string ids;
      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)) + ";";
      log()->info("Detector ID(s) other than {} will be ignored", ids);
    }
    log()->info("*********************************************************************");
    RawBufferNavigator bufferNavigator;
    Timer              timer;
    timer.start();
    m_Source.start();
    m_Destination.start();
    while(m_Source.nextEvent() && m_NbrEventsToProcess >= m_NbrEvents)
    {
      m_Source.startEvent();
      m_Destination.startEvent();

      m_Logger->warn("===*** Event {} ***===", m_NbrEvents);
      while(m_Source.nextDIFbuffer())
      {
        const Buffer& buffer = m_Source.getBuffer();

        bufferNavigator.setBuffer(buffer);
        if(std::find(m_DetectorIDs.begin(), m_DetectorIDs.end(),
static_cast<DetectorID>(bufferNavigator.getDetectorID())) == m_DetectorIDs.end())
        {
          m_Logger->debug("Ignoring detector ID : {}", bufferNavigator.getDetectorID());
          continue;
        }
        std::int32_t idstart = bufferNavigator.getStartOfPayload();
        if(m_Debug && idstart == -1) m_Logger->info(to_hex(buffer));
        c.DIFStarter[idstart]++;
        if(!bufferNavigator.validPayload())
        {
          m_Logger->error("!bufferNavigator.validBuffer()");
          continue;
        }

        m_Source.startDIF();
        m_Destination.startDIF();
        PayloadParser d;
        // This is really a big error so skip DIF entirely if exception occurs
        try
        {
          d.setBuffer(bufferNavigator.getPayload());
        }
        catch(const Exception& e)
        {
          m_Logger->error("{}", e.what());
          continue;
        }
        bit8_t* debug_variable_1 = buffer.end();
        bit8_t* debug_variable_2 = d.end();
        if(debug_variable_1 != debug_variable_2) m_Logger->error("DIF BUFFER END {} {}",
fmt::ptr(debug_variable_1), fmt::ptr(debug_variable_2));
        if(m_Debug) assert(debug_variable_1 == debug_variable_2);

        c.DIFPtrValueAtReturnedPos[d.begin()[d.getGetFramePtrReturn()]]++;
        if(m_Debug) assert(d.begin()[d.getGetFramePtrReturn()] == 0xa0);
        c.SizeAfterDIFPtr[d.getSizeAfterDIFPtr()]++;
        m_Destination.processDIF(d);
        for(std::size_t i = 0; i < d.getNumberOfFrames(); ++i)
        {
          m_Source.startFrame();
          m_Destination.startFrame();
          m_Destination.processFrame(d, i);
          for(std::size_t j = 0; j < static_cast<std::size_t>(Hardware::NUMBER_PAD); ++j)
          {
            if(d.getThresholdStatus(i, j) != 0)
            {
              m_Source.startPad();
              m_Destination.startPad();
              m_Destination.processPadInFrame(d, i, j);
              m_Source.endPad();
              m_Destination.endPad();
            }
          }
          m_Source.endFrame();
          m_Destination.endFrame();
        }
        // If I want SlowControl I need to check for it first, If there is an error then it’s not a
big deal just continue and say is bad SlowControl
        try
        {
          d.setSCBuffer();
        }
        catch(const Exception& e)
        {
          m_Logger->error("{}", e.what());
        }

        bool processSC = false;
        if(d.hasSlowControl())
        {
          c.hasSlowControl++;
          processSC = true;
        }
        if(d.badSCData())
        {
          c.hasBadSlowControl++;
          processSC = false;
        }
        if(processSC) { m_Destination.processSlowControl(d.getSCBuffer()); }

        Buffer eod = d.getEndOfAllData();
        c.SizeAfterAllData[eod.size()]++;
        bit8_t* debug_variable_3 = eod.end();
        if(debug_variable_1 != debug_variable_3) m_Logger->info("END DATA BUFFER END {} {}",
fmt::ptr(debug_variable_1), fmt::ptr(debug_variable_3));
        if(m_Debug) assert(debug_variable_1 == debug_variable_3);
        if(eod.size() != 0) m_Logger->info("End of Data remaining stuff :  {}", to_hex(eod));

        int nonzeroCount = 0;
        for(bit8_t* it = eod.begin(); it != eod.end(); it++)
          if(static_cast<int>(*it) != 0) nonzeroCount++;
        c.NonZeroValusAtEndOfData[nonzeroCount]++;
        m_Source.endDIF();
        m_Destination.endDIF();
      }  // end of DIF while loop
      m_Logger->warn("===*** Event {} ***===", m_NbrEvents);
      m_NbrEvents++;
      m_Source.endEvent();
      m_Destination.endEvent();
    }  // end of event while loop
    m_Destination.end();
    m_Source.end();
    timer.stop();
    fmt::print(fg(fmt::color::green) | fmt::emphasis::bold, "=== elapsed time {}ms ({}ms/event)
===\n", timer.getElapsedTime() / 1000, timer.getElapsedTime() / (1000 * m_NbrEvents));
  }
```

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

Definition at line 222 of file BufferLooper.h.

```
{ 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 225 of file BufferLooper.h.

```
{ 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 10 of file BufferLooperCounter.cc.

```
{
  fmt::print(fg(fmt::color::crimson) | fmt::emphasis::bold, "BUFFER LOOP FINAL STATISTICS : \n");
  printCounter("Start of DIF header", DIFStarter);
  printCounter("Value after DIF data are processed", DIFPtrValueAtReturnedPos);
  printCounter("Size remaining in buffer after end of DIF data", SizeAfterDIFPtr);
  fmt::print(fg(fmt::color::crimson) | fmt::emphasis::bold, "Number of Slow Control found {}  out of
     which {} are bad\n", hasSlowControl, hasBadSlowControl);
  printCounter("Size remaining after all of data have been processed", SizeAfterAllData);
  printCounter("Number on non zero values in end of data buffer", NonZeroValusAtEndOfData);
}
```

4.3.2.2 printCounter() `void BufferLooperCounter::printCounter ( const std::string & description, const std::map< int, int > & m )`

Definition at line 21 of file BufferLooperCounter.cc.

```
{
  std::string out{"statistics for " + description + " :  \n"};
  for(std::map<int, int>::const_iterator it = m.begin(); it != m.end(); it++)
  {
    if(it != m.begin()) out += ",";
    out += " [" + std::to_string(it->first) + "]=" + std::to_string(it->second);
  }
  out += "\n";
  fmt::print(fg(fmt::color::crimson) | fmt::emphasis::bold, out);
}
```

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.

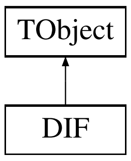
The documentation for this struct was generated from the following files:

- 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

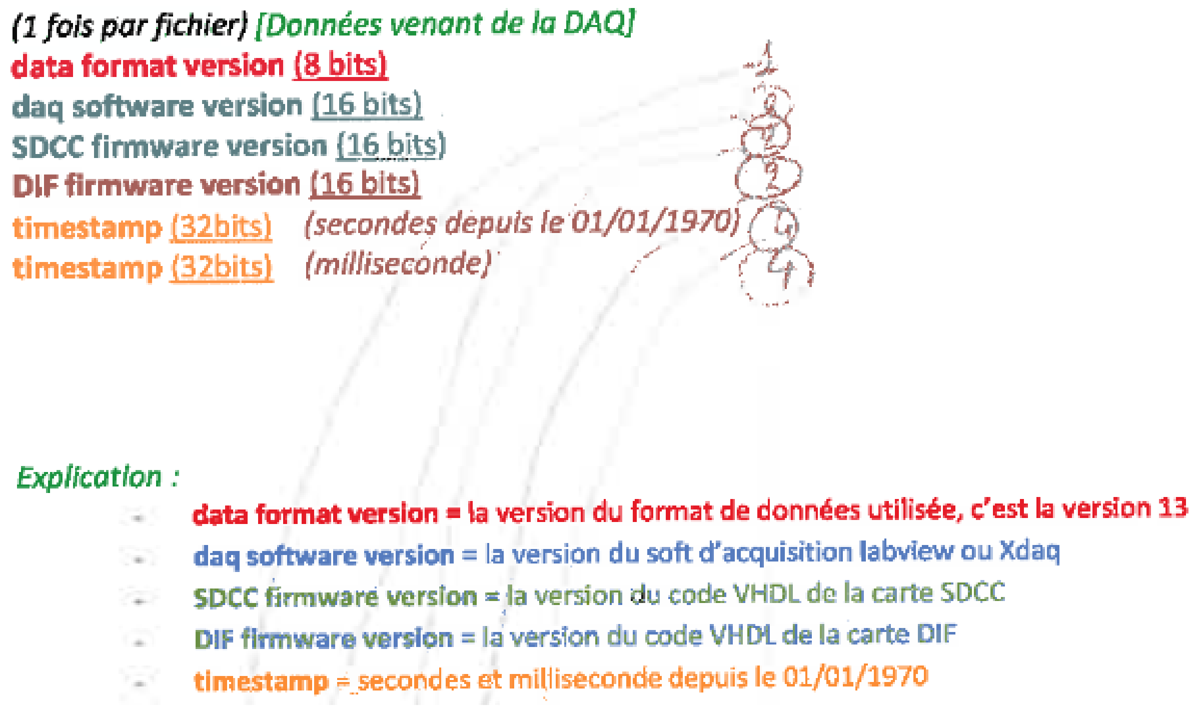
M3 MICROROC and HARDROC2 dataformat.

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

4.5.1 Detailed Description

M3 MICROROC and HARDROC2 dataformat.

Data from the DAQ (once at the beginning of the file) :


(1 fois par fichier) *[Données venant de la DAQ]*

data format version (8 bits) 1

daq software version (16 bits) 2

SDCC firmware version (16 bits) 2

DIF firmware version (16 bits) 2

timestamp (32bits) *(secondes depuis le 01/01/1970)* 4

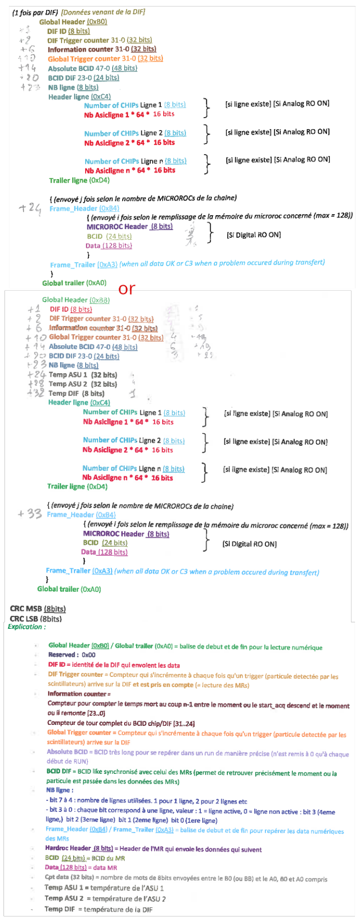
timestamp (32bits) *(milliseconde)* 4

Explication :

- **data format version** = la version du format de données utilisée, c'est la version 13
- **daq software version** = la version du soft d'acquisition labview ou Xdaq
- **SDCC firmware version** = la version du code VHDL de la carte SDCC
- **DIF firmware version** = la version du code VHDL de la carte DIF
- **timestamp** = secondes et milliseconde depuis le 01/01/1970

Figure 1 Data from the DAQ (once at the beginning of the file)

Data from the DIF analog or/and digital (loop) :



```
(1 fois par DIF) [Données venant de la DIF]
      Global Header (0xB0)
+1       DIF ID (8 bits)
+2       DIF Trigger counter 31-0 (32 bits)
+6       Information counter 31-0 (32 bits)
+10      Global Trigger counter 31-0 (32 bits)
+14      Absolute BCID 47-0 (48 bits)
+20      BCID DIF 23-0 (24 bits)
+23      NB ligne (8 bits)
         Header ligne (0xC4)
                  Number of CHIPs Ligne 1 (8 bits)    }   [si ligne existe] [Si Analog RO ON]
                  Nb Asicligne 1 * 64 *  16 bits      }

                  Number of CHIPs Ligne 2 (8 bits)    }   [si ligne existe] [Si Analog RO ON]
                  Nb Asicligne 2 * 64 *  16 bits      }

                  Number of CHIPs Ligne n (8 bits)    }   [si ligne existe] [Si Analog RO ON]
                  Nb Asicligne n * 64 *  16 bits      }
         Trailer ligne (0xD4)

         { (envoyé j fois selon le nombre de MICROROCs de la chaine)
+24      Frame_Header (0xB4)
                  { (envoyé i fois selon le remplissage de la mémoire du microroc concerné (max = 128))
                  MICROROC Header  (8 bits)   }
                  BCID  (24 bits)             }   [Si Digital RO ON]
                  Data (128 bits)             }
                  }
         Frame_Trailer (0xA3) (when all data OK or C3 when a problem occured during transfert)
         }
      Global trailer (0xA0)
```

or

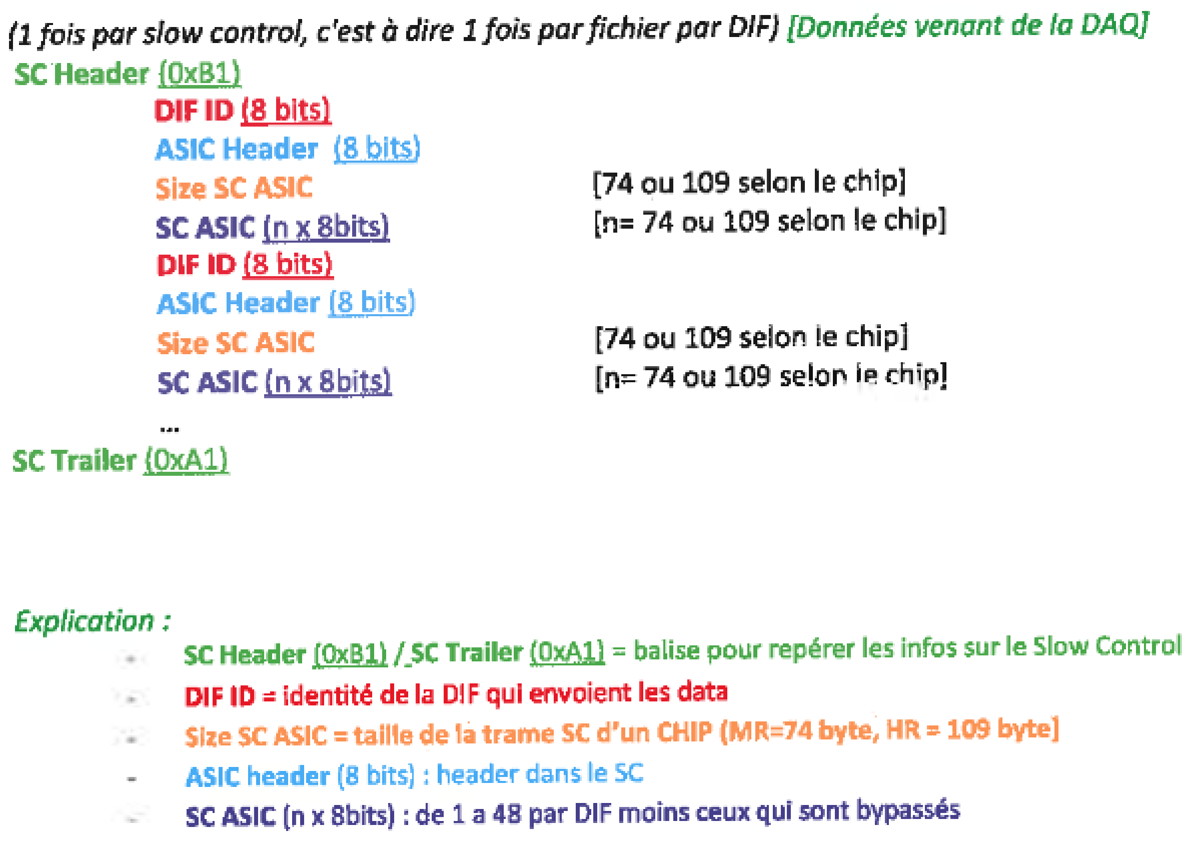
```
      Global Header (0xBB)
+1       DIF ID (8 bits)
+2       DIF Trigger counter 31-0 (32 bits)
+6       Information counter 31-0 (32 bits)
+10      Global Trigger counter 31-0 (32 bits)
+14      Absolute BCID 47-0 (48 bits)
+20      BCID DIF 23-0 (24 bits)
+23      NB ligne (8 bits)
+24      Temp ASU 1  (32 bits)
+28      Temp ASU 2  (32 bits)
+32      Temp DIF  (8 bits)
         Header ligne (0xC4)
                  Number of CHIPs Ligne 1 (8 bits)    }   [si ligne existe] [Si Analog RO ON]
                  Nb Asicligne 1 * 64 *  16 bits      }

                  Number of CHIPs Ligne 2 (8 bits)    }   [si ligne existe] [Si Analog RO ON]
                  Nb Asicligne 2 * 64 *  16 bits      }

                  Number of CHIPs Ligne n (8 bits)    }   [si ligne existe] [Si Analog RO ON]
                  Nb Asicligne n * 64 *  16 bits      }
         Trailer ligne (0xD4)

         { (envoyé j fois selon le nombre de MICROROCs de la chaine)
+33      Frame_Header (0xB4)
                  { (envoyé i fois selon le remplissage de la mémoire du microroc concerné (max = 128))
                  MICROROC Header  (8 bits)   }
                  BCID  (24 bits)             }   [Si Digital RO ON]
                  Data (128 bits)             }
                  }
         Frame_Trailer (0xA3) (when all data OK or C3 when a problem occured during transfert)
         }
      Global trailer (0xA0)

CRC MSB (8bits)
CRC LSB (8bits)
```

Explication :

- **Global Header** (0xB0) / **Global trailer** (0xA0) = balise de debut et de fin pour la lecture numérique
- **Reserved :** 0x00
- **DIF ID** = identité de la DIF qui envoient les data
- **DIF Trigger counter** = Compteur qui s'incrémente à chaque fois qu'un trigger (particule detectée par les scintillateurs) arrive sur la DIF **et est pris en compte** (= lecture des MRs)
- **Information counter =**
  Compteur pour compter le temps mort au coup n-1 entre le moment ou le start_acq descend et le moment ou il remonte [23..0]
  Compteur de tour complet du BCID chip/DIF [31..24]
- **Global Trigger counter** = Compteur qui s'incrémente à chaque fois qu'un trigger (particule detectée par les scintillateurs) arrive sur la DIF
- **Absolute BCID** = BCID très long pour se repérer dans un run de manière précise (n'est remis à 0 qu'à chaque début de RUN)
- **BCID DIF** = BCID like synchronisé avec celui des MRs (permet de retrouver précisément le moment ou la particule est passée dans les données des MRs)
- **NB ligne :**
  - bit 7 à 4 : nombre de lignes utilisées. 1 pour 1 ligne, 2 pour 2 lignes etc
  - bit 3 à 0 : chaque bit correspond à une ligne, valeur : 1 = ligne active, 0 = ligne non active : bit 3 (4eme ligne,) bit 2 (3eme ligne) bit 1 (2eme ligne) bit 0 (1ere ligne)
- **Frame_Header** (0xB4) / **Frame_Trailer** (0xA3) = balise de debut et de fin pour repérer les data numériques des MRs
- **Hardroc Header** (8 bits) = Header de l'MR qui envoie les données qui suivent
- **BCID** (24 bits) = BCID du MR
- **Data** (128 bits) = data MR
- **Cpt data** (32 bits) = nombre de mots de 8bits envoyées entre le B0 (ou BB) et le A0, B0 et A0 compris
- **Temp ASU 1** = température de l'ASU 1
- **Temp ASU 2** = température de l'ASU 2
- **Temp DIF** = température de la DIF

Figure 2 Data from the DIF analog or/and digital

Data from the DAQ (slowcontrol) :


(1 fois par slow control, c'est à dire 1 fois par fichier par DIF) *[Données venant de la DAQ]*

SC Header (0xB1)

DIF ID (8 bits)

ASIC Header (8 bits)

Size SC ASIC [74 ou 109 selon le chip]

SC ASIC (n x 8bits) [n= 74 ou 109 selon le chip]

DIF ID (8 bits)

ASIC Header (8 bits)

Size SC ASIC [74 ou 109 selon le chip]

SC ASIC (n x 8bits) [n= 74 ou 109 selon le chip]

...

SC Trailer (0xA1)

Explication :

- SC Header (0xB1) / SC Trailer (0xA1) = balise pour repérer les infos sur le Slow Control
- DIF ID = identité de la DIF qui envoient les data
- Size SC ASIC = taille de la trame SC d'un CHIP (MR=74 byte, HR = 109 byte]
- ASIC header (8 bits) : header dans le SC
- SC ASIC (n x 8bits) : de 1 a 48 par DIF moins ceux qui sont bypassés

Figure 3 Data from the DAQ (slowcontrol)

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

- libs/core/include/PayloadParser.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 DIFSlow←
Control::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; }
```

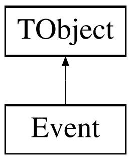
The documentation for this class was generated from the following files:

- 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 Exception Class Reference

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

Public Member Functions

- virtual const char ∗ what () const noexcept
- Exception (const std::string &message)
- Exception (const std::int32_t &error, const std::string &message)
- std::int32_t error ()
- std::string message ()

4.8.1 Detailed Description

Definition at line 11 of file Exception.h.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Exception() [1/2] `Exception::Exception ( const std::string & message ) [inline], [explicit]`

Definition at line 15 of file Exception.h.

```
00015 : m_Message(message) { constructWhat(); }
```

4.8.2.2 Exception() [2/2]

```
Exception::Exception (
        const std::int32_t & error,
        const std::string & message )  [inline]
```

Definition at line 16 of file Exception.h.

```
00016 :  m_Error(error), m_Message(message) { constructWhat(); }
```

4.8.3 Member Function Documentation

4.8.3.1 error()

```
std::int32_t Exception::error ( )  [inline]
```

Definition at line 17 of file Exception.h.

```
00017 { return m_Error; }
```

4.8.3.2 message()

```
std::string Exception::message ( )  [inline]
```

Definition at line 18 of file Exception.h.

```
00018 { return m_Message; }
```

4.8.3.3 what()

```
virtual const char * Exception::what ( ) const  [inline], [virtual], [noexcept]
```

Definition at line 14 of file Exception.h.

```
00014 { return m_What.c_str(); }
```

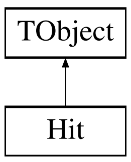
The documentation for this class was generated from the following file:

- libs/core/include/Exception.h

4.9 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

4.9.1 Detailed Description

Definition at line 10 of file Hit.h.

4.9.2 Member Function Documentation

4.9.2.1 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 }
```

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

Definition at line 59 of file Hit.cc.

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

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

Definition at line 43 of file Hit.cc.

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

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

Definition at line 45 of file Hit.cc.

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

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

Definition at line 53 of file Hit.cc.

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

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

Definition at line 41 of file Hit.cc.

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

4.9.2.7 getDTC() `std::uint32_t Hit::getDTC ( ) const`

Definition at line 49 of file Hit.cc.

```
00049 { return m_DTC; }
```

4.9.2.8 getFrameBCID() `std::uint32_t Hit::getFrameBCID ( ) const`

Definition at line 55 of file Hit.cc.

```
00055 { return m_FrameBCID; }
```

4.9.2.9 getGTC() `std::uint32_t Hit::getGTC ( ) const`

Definition at line 51 of file Hit.cc.

```
00051 { return m_GTC; }
```

4.9.2.10 getThreshold() `std::uint8_t Hit::getThreshold ( ) const`

Definition at line 47 of file Hit.cc.

```
00047 { return m_Threshold; }
```

4.9.2.11 getTimestamp() `std::uint32_t Hit::getTimestamp ( ) const`

Definition at line 57 of file Hit.cc.

```
00057 { return m_Timestamp; }
```

4.9.2.12 setAbsoluteBCID() `void Hit::setAbsoluteBCID ( const std::uint64_t & absolutebcid )`

Definition at line 39 of file Hit.cc.

```
00039 { m_AbsoluteBCID = absolutebcid; }
```

4.9.2.13 setASIC() `void Hit::setASIC ( const std::uint8_t & asic )`

Definition at line 23 of file Hit.cc.

```
00023 { m_ASIC = asic; }
```

4.9.2.14 setChannel() `void Hit::setChannel ( const std::uint8_t & channel )`

Definition at line 25 of file Hit.cc.

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

4.9.2.15 setDIF() `void Hit::setDIF ( const std::uint8_t & dif )`

Definition at line 21 of file Hit.cc.

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

4.9.2.16 setDIFBCID() `void Hit::setDIFBCID ( const std::uint32_t & difbcid )`

Definition at line 33 of file Hit.cc.

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

4.9.2.17 setDTC() `void Hit::setDTC ( const std::uint32_t & dtc )`

Definition at line 29 of file Hit.cc.

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

4.9.2.18 setFrameBCID() `void Hit::setFrameBCID ( const std::uint32_t & framebcid )`

Definition at line 35 of file Hit.cc.

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

4.9.2.19 setGTC() `void Hit::setGTC ( const std::uint32_t & gtc )`

Definition at line 31 of file Hit.cc.

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

4.9.2.20 setThreshold() `void Hit::setThreshold ( const std::uint8_t & threshold )`

Definition at line 27 of file Hit.cc.

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

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

Definition at line 37 of file Hit.cc.

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

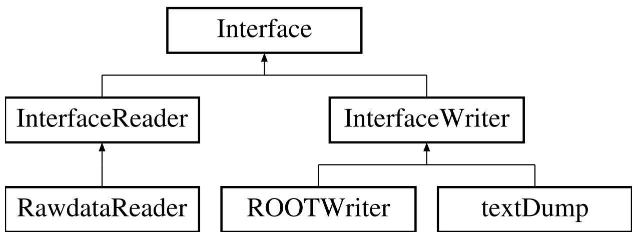
The documentation for this class was generated from the following files:

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

4.10 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 ()

4.10.1 Detailed Description

Definition at line 38 of file Interface.h.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Interface() `Interface::Interface (`

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

Definition at line 41 of file Interface.h.

```
00041 :  m_Name(name), m_Version(version) {}
```

4.10.2.2 ∼Interface() `virtual Interface::∼Interface ( ) [virtual], [default]`

4.10.3 Member Function Documentation

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

Reimplemented in ROOTWriter.

Definition at line 46 of file Interface.h.

```
00046 {}
```

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

Reimplemented in ROOTWriter.

Definition at line 44 of file Interface.h.
```
00044 {}
```

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

Reimplemented in ROOTWriter.

Definition at line 48 of file Interface.h.
```
00048 {}
```

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

Reimplemented in ROOTWriter.

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

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

Definition at line 53 of file Interface.h.
```
00053 { return m_Name; }
```

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

Definition at line 54 of file Interface.h.
```
00054 { return m_Version; }
```

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

Definition at line 51 of file Interface.h.
```
00051 { return m_Logger; }
```

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

Definition at line 52 of file Interface.h.
```
00052 { m_Logger = logger; }
```

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

Reimplemented in ROOTWriter.

Definition at line 45 of file Interface.h.
```
00045 {}
```

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

Reimplemented in ROOTWriter.

Definition at line 43 of file Interface.h.
```
00043 {}
```

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

Reimplemented in ROOTWriter.

Definition at line 47 of file Interface.h.
```
00047 {}
```

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

Reimplemented in ROOTWriter.

Definition at line 49 of file Interface.h.
```
00049 {}
```

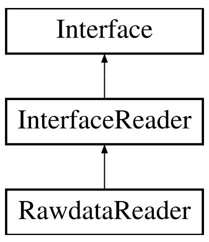
The documentation for this class was generated from the following file:

- libs/core/include/Interface.h

4.11 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

4.11.1 Detailed Description

Definition at line 63 of file Interface.h.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 InterfaceReader()

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

Definition at line 66 of file Interface.h.

```
00066 :  Interface(name, version, InterfaceType::Reader) {}
```

4.11.2.2 ∼InterfaceReader()

```
virtual InterfaceReader::∼InterfaceReader ( )  [virtual], [default]
```

4.11.3 Member Data Documentation

4.11.3.1 m_Buffer

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

Definition at line 70 of file Interface.h.

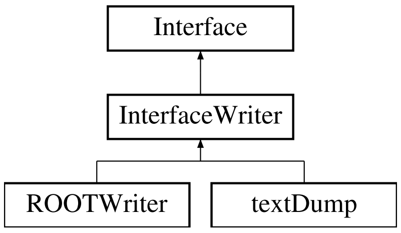
The documentation for this class was generated from the following file:

- libs/core/include/Interface.h

4.12 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

4.12.1 Detailed Description

Definition at line 73 of file Interface.h.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 InterfaceWriter()

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

Definition at line 76 of file Interface.h.

```
00076 :  Interface(name, version, InterfaceType::Writer) {}
```

4.12.2.2 ∼InterfaceWriter()

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

4.12.3 Member Function Documentation

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

Definition at line 78 of file Interface.h.

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

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

Definition at line 82 of file Interface.h.

```
{
  if(m_Compatible.find(name) != m_Compatible.end())
  {
    auto            ran = semver::range::detail::range(m_Compatible[name]);
    semver::version ver = semver::version(version);
    if(ran.satisfies(ver, false)) return true;
    else
      return false;
  }
  else
    return false;
}
```

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

Definition at line 80 of file Interface.h.

```
{ return m_Compatible; }
```

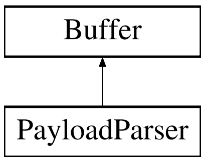
The documentation for this class was generated from the following file:

- libs/core/include/Interface.h

4.13 PayloadParser Class Reference

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

Inheritance diagram for PayloadParser:



Public Member Functions

- PayloadParser ()=default
- void setBuffer (const Buffer &buffer)
- bool hasTemperature () const
- bool hasAnalogReadout () const
- bool hasSlowControl () const
- float getTemperatureDIF () const
- float getTemperatureASU1 () const
- float getTemperatureASU2 () const
- Buffer getSlowControl () const
- std::vector$<$ bit8_t $*$ $>$ & getFramesVector ()
- std::vector$<$ bit8_t $*$ $>$ & getLinesVector ()
- std::uint32_t getSizeAfterDIFPtr ()
- std::uint32_t getEndOfDIFData () const
- bool badSCData ()
- std::uint32_t getGetFramePtrReturn () const
- std::uint32_t getDTC () const
- std::uint32_t getGTC () const
- std::uint64_t getAbsoluteBCID () const
- std::uint32_t getBCID () const
- bool hasLine (const std::uint32_t &) const
- std::uint32_t getNumberOfFrames () const
- bit8_t $*$ getFramePtr (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
- Buffer getSCBuffer ()
- Buffer getEndOfAllData ()
- std::uint32_t getDIF_CRC ()
- void setSCBuffer ()

4.13.1 Detailed Description

Definition at line 36 of file PayloadParser.h.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 PayloadParser() `PayloadParser::PayloadParser ( ) [default]`

4.13.3 Member Function Documentation

4.13.3.1 badSCData() `bool PayloadParser::badSCData ( ) [inline]`

Definition at line 63 of file PayloadParser.h.

```
  {
    setSCBuffer();
    return m_BadSlowControl;
  }
```

4.13.3.2 getAbsoluteBCID() `std::uint64_t PayloadParser::getAbsoluteBCID ( ) const [inline]`

Definition at line 298 of file PayloadParser.h.

```
{
  std::uint32_t shift{Size::GLOBAL_HEADER + Size::DIF_IF + Size::DIF_TRIGGER_COUNTER +
  Size::INFORMATION_COUNTER + Size::GLOBAL_TRIGGER_COUNTER};
  std::uint64_t LBC = ((begin()[shift] « 16) | (begin()[shift + 1] « 8) | (begin()[shift + 2])) *
  16777216ULL /* to shift the value from the 24 first bits*/
                      + ((begin()[shift + 3] « 16) | (begin()[shift + 4] « 8) | (begin()[shift + 5]));
  return LBC;
}
```

4.13.3.3 getASICid() `uint32_t PayloadParser::getASICid ( const std::uint32_t & i ) const [inline]`

Definition at line 342 of file PayloadParser.h.

```
{ return m_Frames[i][0] & 0xFF; }
```

4.13.3.4 getBCID() `std::uint32_t PayloadParser::getBCID ( ) const [inline]`

Definition at line 306 of file PayloadParser.h.

```
{
  std::uint32_t shift{Size::GLOBAL_HEADER + Size::DIF_IF + Size::DIF_TRIGGER_COUNTER +
  Size::INFORMATION_COUNTER + Size::GLOBAL_TRIGGER_COUNTER + Size::ABSOLUTE_BCID};
  return (begin()[shift] « 16) + (begin()[shift + 1] « 8) + begin()[shift + 2];
}
```

4.13.3.5 getDIF_CRC() `std::uint32_t PayloadParser::getDIF_CRC ( ) [inline]`

Definition at line 95 of file PayloadParser.h.

```
  {
    uint32_t i{getEndOfDIFData()};
    uint32_t ret{0};
    ret |= ((begin()[i - 2]) « 8);
    ret |= begin()[i - 1];
    return ret;
  }
```

4.13.3.6 getDIFid() `uint32_t PayloadParser::getDIFid ( ) const [inline]`

Definition at line 336 of file PayloadParser.h.

```
{
  std::uint32_t shift{+Size::GLOBAL_HEADER};
  return begin()[shift] & 0xFF;
}
```

4.13.3.7 getDTC() `std::uint32_t PayloadParser::getDTC ( ) const [inline]`

Definition at line 286 of file PayloadParser.h.

```
00287 {
00288   std::uint32_t shift{Size::GLOBAL_HEADER + Size::DIF_IF};
00289   return (begin()[shift] « 24) + (begin()[shift + 1] « 16) + (begin()[shift + 2] « 8) + begin()[shift
      + 3];
00290 }
```

4.13.3.8 getEndOfAllData() `Buffer PayloadParser::getEndOfAllData ( ) [inline]`

Definition at line 88 of file PayloadParser.h.

```
00089   {
00090     setSCBuffer();
00091     if(hasSlowControl() && !m_BadSlowControl) { return
      Buffer(&(m_SCbuffer.begin()[m_SCbuffer.size()]), getSizeAfterDIFPtr() - 3 - m_SCbuffer.size()); }
00092     else
00093       return Buffer(&(begin()[getEndOfDIFData()]), getSizeAfterDIFPtr() - 3);  // remove the 2 bytes
      for CRC and the DIF trailer
00094   }
```

4.13.3.9 getEndOfDIFData() `std::uint32_t PayloadParser::getEndOfDIFData ( ) const [inline]`

Definition at line 62 of file PayloadParser.h.

```
00062 { return getGetFramePtrReturn() + 3; }
```

4.13.3.10 getFrameBCID() `std::uint32_t PayloadParser::getFrameBCID ( const std::uint32_t & i ) const [inline]`

Definition at line 322 of file PayloadParser.h.

```
00323 {
00324   std::uint32_t shift{+Size::MICROROC_HEADER};
00325   return GrayToBin((m_Frames[i][shift] « 16) + (m_Frames[i][shift + 1] « 8) + m_Frames[i][shift + 2]);
00326 }
```

4.13.3.11 getFrameLevel() `bool PayloadParser::getFrameLevel ( const std::uint32_t & i, const std::uint32_t & ipad, const std::uint32_t & ilevel ) const [inline]`

Definition at line 330 of file PayloadParser.h.

```
00331 {
00332   std::uint32_t shift{Size::MICROROC_HEADER + Size::BCID};
00333   return ((m_Frames[i][shift + ((3 - ipad / 16) * 4 + (ipad % 16) / 4)] » (7 - (((ipad % 16) % 4) * 2
      + ilevel))) & 0x1);
00334 }
```

4.13.3.12 getFramePtr()

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

Definition at line 320 of file PayloadParser.h.

```
00320 { return m_Frames[i]; }
```

4.13.3.13 getFramesVector()

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

Definition at line 280 of file PayloadParser.h.

```
00280 { return m_Frames; }
```

4.13.3.14 getFrameTimeToTrigger()

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

Definition at line 328 of file PayloadParser.h.

```
00328 { return getBCID() - getFrameBCID(i); }
```

4.13.3.15 getGetFramePtrReturn()

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

Definition at line 284 of file PayloadParser.h.

```
00284 { return theGetFramePtrReturn_ - 3; }
```

4.13.3.16 getGTC()

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

Definition at line 292 of file PayloadParser.h.

```
00293 {
00294   std::uint32_t shift{Size::GLOBAL_HEADER + Size::DIF_IF + Size::DIF_TRIGGER_COUNTER +
      Size::INFORMATION_COUNTER};
00295   return (begin()[shift] « 24) + (begin()[shift + 1] « 16) + (begin()[shift + 2] « 8) + begin()[shift
      + 3];
00296 }
```

4.13.3.17 getLinesVector()

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

Definition at line 282 of file PayloadParser.h.

```
00282 { return m_Lines; }
```

4.13.3.18 getNumberOfFrames()

`std::uint32_t PayloadParser::getNumberOfFrames ( ) const [inline]`

Definition at line 318 of file PayloadParser.h.

```
00318 { return m_Frames.size(); }
```

4.13.3.19 getSCBuffer() `Buffer PayloadParser::getSCBuffer ( ) [inline]`

Definition at line 83 of file PayloadParser.h.

```
00084   {
00085     setSCBuffer();
00086     return m_SCbuffer;
00087   }
```

4.13.3.20 getSizeAfterDIFPtr() `std::uint32_t PayloadParser::getSizeAfterDIFPtr ( ) [inline]`

Definition at line 61 of file PayloadParser.h.

```
00061 { return size() - getGetFramePtrReturn(); }
```

4.13.3.21 getSlowControl() `Buffer PayloadParser::getSlowControl ( ) const [inline]`

Definition at line 273 of file PayloadParser.h.

```
00274 {
00275   if(hasSlowControl()) return Buffer(&begin()[getEndOfDIFData()], size() - getEndOfDIFData());
00276   else
00277     return Buffer();
00278 }
```

4.13.3.22 getTemperatureASU1() `float PayloadParser::getTemperatureASU1 ( ) const [inline]`

Definition at line 261 of file PayloadParser.h.

```
00262 {
00263   if(!hasTemperature()) throw Exception("Don’t have TemperatureASU1 information");
00264   return (getTASU1() » 3) * 0.0625;
00265 }
```

4.13.3.23 getTemperatureASU2() `float PayloadParser::getTemperatureASU2 ( ) const [inline]`

Definition at line 267 of file PayloadParser.h.

```
00268 {
00269   if(!hasTemperature()) throw Exception("Don’t have TemperatureASU2 information");
00270   return (getTASU2() » 3) * 0.0625;
00271 }
```

4.13.3.24 getTemperatureDIF() `float PayloadParser::getTemperatureDIF ( ) const [inline]`

Definition at line 255 of file PayloadParser.h.

```
00256 {
00257   if(!hasTemperature()) throw Exception("Don’t have TemperatureDIF information");
00258   return 0.508 * getTDIF() - 9.659;
00259 }
```

4.13.3.25 getThresholdStatus()

```
uint32_t PayloadParser::getThresholdStatus (
            const std::uint32_t & i,
            const std::uint32_t & ipad ) const [inline]
```

Definition at line 344 of file PayloadParser.h.

```
00344 { return (((std::uint32_t)getFrameLevel(i, ipad, 1)) « 1) | ((std::uint32_t)getFrameLevel(i, ipad,
      0)); }
```

4.13.3.26 hasAnalogReadout()

```
bool PayloadParser::hasAnalogReadout ( ) const [inline]
```

Definition at line 207 of file PayloadParser.h.

```
00207 { return getNumberLines() != 0; }
```

4.13.3.27 hasLine()

```
bool PayloadParser::hasLine (
            const std::uint32_t & line ) const [inline]
```

Definition at line 312 of file PayloadParser.h.

```
00313 {
00314   std::uint32_t shift{Size::GLOBAL_HEADER + Size::DIF_IF + Size::DIF_TRIGGER_COUNTER +
      Size::INFORMATION_COUNTER + Size::GLOBAL_TRIGGER_COUNTER + Size::ABSOLUTE_BCID + Size::BCID_DIF};
00315   return ((begin()[shift] » line) & 0x1);
00316 }
```

4.13.3.28 hasSlowControl()

```
bool PayloadParser::hasSlowControl ( ) const [inline]
```

Definition at line 235 of file PayloadParser.h.

```
00235 { return theGetFramePtrReturn_ != size(); }
```

4.13.3.29 hasTemperature()

```
bool PayloadParser::hasTemperature ( ) const [inline]
```

Definition at line 205 of file PayloadParser.h.

```
00205 { return (static_cast<std::uint8_t>(begin()[0]) ==
      static_cast<std::uint8_t>(Value::GLOBAL_HEADER_TEMP)); }
```

4.13.3.30 setBuffer()

```
void PayloadParser::setBuffer (
            const Buffer & buffer ) [inline]
```

Definition at line 153 of file PayloadParser.h.

```
00154 {
00155   set(buffer);
00156   m_Frames.clear();
00157   m_Lines.clear();
00158   theGetFramePtrReturn_ = parsePayload();
00159   if(theGetFramePtrReturn_ != size()) { fmt::print("****************************\n"); }
00160   m_BadSlowControl = false;
00161 }
```

4.13.3.31 setSCBuffer() `void PayloadParser::setSCBuffer ( ) [inline]`

Definition at line 103 of file PayloadParser.h.

```
{
  if(!hasSlowControl()) return;
  if(m_SCbuffer.size() != 0) return;  // deja fait
  if(m_BadSlowControl) return;
  m_SCbuffer.set(&(begin()[getEndOfDIFData()]));
  // compute Slow Control size
  std::size_t maxsize{size() - getEndOfDIFData() + 1};  // should I +1 here ?
  uint32_t    k{1};                                     // SC Header
  uint32_t    dif_ID{m_SCbuffer[1]};
  uint32_t    chipSize{m_SCbuffer[3]};
  while((dif_ID != 0xa1 && m_SCbuffer[k] != 0xa1 && k < maxsize) || (dif_ID == 0xa1 && m_SCbuffer[k
  + 2] == chipSize && k < maxsize))
  {
    k += 2;  // DIF ID + ASIC Header
    uint32_t scsize = m_SCbuffer[k];
    if(scsize != 74 && scsize != 109)
    {
      k                = 0;
      m_BadSlowControl = true;
      throw Exception(fmt::format("PROBLEM WITH SC SIZE {}", scsize));
    }
    k++;          // skip size bit
    k += scsize;  // skip the data
  }
  if(m_SCbuffer[k] == 0xa1 && !m_BadSlowControl) m_SCbuffer.setSize(k + 1);  // add the trailer
  else
  {
    m_BadSlowControl = true;
    throw Exception(fmt::format("PROBLEM SC TRAILER NOT FOUND "));
  }
}
```

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

- libs/core/include/PayloadParser.h

4.14 RawBufferNavigator Class Reference

class to navigate in the raw data buffer parse the header and send the payload as Buffer

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

Public Member Functions

- RawBufferNavigator ()
- ∼RawBufferNavigator ()=default
- void setBuffer (const Buffer &)
- std::uint8_t getDetectorID ()
- bool findStartOfPayload ()
- std::int32_t getStartOfPayload ()
- bool validPayload ()
- Buffer getPayload ()

Static Public Member Functions

- static void StartAt (const int &start)

4.14.1 Detailed Description

class to navigate in the raw data buffer parse the header and send the payload as Buffer

Definition at line 13 of file RawBufferNavigator.h.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 RawBufferNavigator() `RawBufferNavigator::RawBufferNavigator ( )`

Definition at line 16 of file RawBufferNavigator.cc.

```
00016 {}
```

4.14.2.2 ∼RawBufferNavigator() `RawBufferNavigator::∼RawBufferNavigator ( ) [default]`

4.14.3 Member Function Documentation

4.14.3.1 findStartOfPayload() `bool RawBufferNavigator::findStartOfPayload ( )`

Definition at line 27 of file RawBufferNavigator.cc.

```
00028 {
00029   if(m_StartPayloadDone == true)
00030   {
00031     if(m_StartPayload == -1) return false;
00032     else
00033       return true;
00034   }
00035   else
00036   {
00037     m_StartPayloadDone = true;
00038     for(std::size_t i = m_Start; i < m_Buffer.size(); i++)
00039     {
00040       if(static_cast<std::uint8_t>(m_Buffer[i]) == static_cast<std::uint8_t>(Value::GLOBAL_HEADER) ||
      static_cast<std::uint8_t>(m_Buffer[i]) == static_cast<std::uint8_t>(Value::GLOBAL_HEADER_TEMP))
00041       {
00042         m_StartPayload = i;
00043         return true;
00044       }
00045     }
00046     m_StartPayload = -1;
00047     return false;
00048   }
00049 }
```

4.14.3.2 getDetectorID() `std::uint8_t RawBufferNavigator::getDetectorID ( )`

Definition at line 25 of file RawBufferNavigator.cc.

```
00025 { return m_Buffer[0]; }
```

4.14.3.3 getPayload() `Buffer RawBufferNavigator::getPayload ( )`

Definition at line 59 of file RawBufferNavigator.cc.

```
00059 { return Buffer(&(m_Buffer.begin()[m_StartPayload]), m_Buffer.size() - m_StartPayload); }
```

4.14.3.4 getStartOfPayload() `std::int32_t RawBufferNavigator::getStartOfPayload ( )`

Definition at line 51 of file RawBufferNavigator.cc.

```
00052 {
00053   findStartOfPayload();
00054   return m_StartPayload;
00055 }
```

4.14.3.5 setBuffer()

```
void RawBufferNavigator::setBuffer (
            const Buffer & b )
```

Definition at line 18 of file RawBufferNavigator.cc.

```
00019 {
00020   m_Buffer            = b;
00021   m_StartPayload      = -1;
00022   m_StartPayloadDone = false;
00023 }
```

4.14.3.6 StartAt()

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

Definition at line 11 of file RawBufferNavigator.cc.

```
00012 {
00013   if(start >= 0) m_Start = start;
00014 }
```

4.14.3.7 validPayload() `bool RawBufferNavigator::validPayload ( )`

Definition at line 57 of file RawBufferNavigator.cc.

```
00057 { return m_StartPayload != -1; }
```

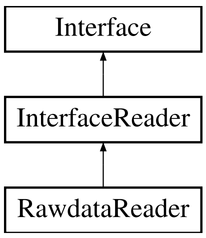
The documentation for this class was generated from the following files:

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

4.15 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

4.15.1 Detailed Description

Definition at line 17 of file RawdataReader.h.

4.15.2 Constructor & Destructor Documentation

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

Definition at line 18 of file RawdataReader.cc.

```
00018                                            : InterfaceReader("RawdataReader", "1.0.0")
00019 {
00020   m_buf.reserve(m_BufferSize);
00021   m_Filename = fileName;
00022 }
```

4.15.2.2 ∼RawdataReader() `virtual RawdataReader::∼RawdataReader ( ) [inline], [virtual]`

Definition at line 29 of file RawdataReader.h.

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

4.15.3 Member Function Documentation

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

Definition at line 47 of file RawdataReader.cc.
```
{
  try
  {
    if(m_FileStream.is_open()) m_FileStream.close();
  }
  catch(const std::ios_base::failure& e)
  {
    log()->error("Caught an ios_base::failure in closeFile :  {} {}", e.what(), e.code().value());
    throw;
  }
}
```

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

Definition at line 26 of file RawdataReader.cc.
```
{ closeFile(); }
```

4.15.3.3 getBuffer() `const Buffer & RawdataReader::getBuffer ( )`

Definition at line 122 of file RawdataReader.cc.
```
{
  uncompress();
  return m_Buffer;
}
```

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

Definition at line 130 of file RawdataReader.cc.
```
{ return m_FileSize; }
```

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

Definition at line 95 of file RawdataReader.cc.
```
{
  try
  {
    static int DIF_processed{0};
    if(DIF_processed >= m_NumberOfDIF)
    {
      DIF_processed = 0;
      return false;
    }
    else
    {
      DIF_processed++;
      std::uint32_t bsize{0};
      m_FileStream.read(reinterpret_cast<char*>(&bsize), sizeof(std::uint32_t));
      m_FileStream.read(reinterpret_cast<char*>(&m_buf[0]), bsize);
      m_Buffer = Buffer(m_buf);
    }
  }
  catch(const std::ios_base::failure& e)
  {
    log()->error("Caught an ios_base::failure in openFile :  {}", e.what());
    return false;
  }
  return true;
}
```

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

Definition at line 81 of file RawdataReader.cc.

```
{
  try
  {
    m_FileStream.read(reinterpret_cast<char*>(&m_EventNumber), sizeof(std::uint32_t));
    m_FileStream.read(reinterpret_cast<char*>(&m_NumberOfDIF), sizeof(std::uint32_t));
  }
  catch(const std::ios_base::failure& e)
  {
    return false;
  }
  return true;
}
```

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

Definition at line 60 of file RawdataReader.cc.

```
{
  try
  {
    m_FileStream.rdbuf()->pubsetbuf(0, 0);
    m_FileStream.exceptions(std::ifstream::failbit | std::ifstream::badbit);
    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
    m_FileStream.rdbuf()->pubsetbuf(0, 0);
    if(m_FileStream.is_open())
    {
      setFileSize(m_FileStream.tellg());
      m_FileStream.seekg(0, std::ios::beg);
    }
  }
  catch(const std::ios_base::failure& e)
  {
    log()->error("Caught an ios_base::failure in openFile :  {}", e.what());
    throw;
  }
}
```

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

Definition at line 16 of file RawdataReader.cc.

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

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

Definition at line 24 of file RawdataReader.cc.

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

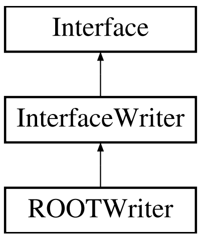
The documentation for this class was generated from the following files:

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

4.16 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 PayloadParser &)
- void processFrame (const PayloadParser &, const std::uint32_t &frameIndex)
- void processPadInFrame (const PayloadParser &, const std::uint32_t &frameIndex, const std::uint32_↩ t &channelIndex)
- 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 ()

4.16.1 Detailed Description

Definition at line 18 of file ROOTWriter.h.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 ROOTWriter() `ROOTWriter::ROOTWriter ( )`

Definition at line 10 of file ROOTWriter.cc.

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

4.16.3 Member Function Documentation

4.16.3.1 end() `void ROOTWriter::end ( )`

Definition at line 19 of file ROOTWriter.cc.

```
{
  if(m_Tree) m_Tree->Write();
  if(m_File)
  {
    m_File->Write();
    m_File->Close();
  }
  if(m_File) delete m_File;
}
```

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

Reimplemented from Interface.

Definition at line 75 of file ROOTWriter.cc.

```
{
  m_Event->addDIF(*m_DIF);
  delete m_DIF;
}
```

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

Reimplemented from Interface.

Definition at line 63 of file ROOTWriter.cc.

```
{
  m_Tree->Fill();
  if(m_Event) delete m_Event;
}
```

4.16.3.4 endFrame() `void ROOTWriter::endFrame ( ) [virtual]`

Reimplemented from Interface.

Definition at line 87 of file ROOTWriter.cc.

```
{
  m_DIF->addHit(*m_Hit);
  delete m_Hit;
}
```

4.16.3.5 endPad() `void ROOTWriter::endPad ( ) [virtual]`

Reimplemented from Interface.

Definition at line 95 of file ROOTWriter.cc.

```
{}
```

4.16.3.6 processDIF()

```
void ROOTWriter::processDIF (
        const PayloadParser & 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 }
```

4.16.3.7 processFrame()

```
void ROOTWriter::processFrame (
        const PayloadParser & 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 }
```

4.16.3.8 processPadInFrame()

```
void ROOTWriter::processPadInFrame (
        const PayloadParser & 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 }
```

4.16.3.9 processSlowControl()

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

Definition at line 29 of file ROOTWriter.h.

```
00029 { ; }
```

4.16.3.10 setFilename()

```
void ROOTWriter::setFilename (
        const std::string & filename )
```

Definition at line 8 of file ROOTWriter.cc.

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

4.16.3.11 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 }
```

4.16.3.12 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 }
```

4.16.3.13 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 }
```

4.16.3.14 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 }
```

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

Reimplemented from Interface.

Definition at line 93 of file ROOTWriter.cc.

```
00093 {}
```

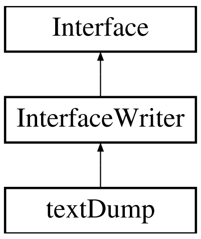
The documentation for this class was generated from the following files:

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

4.17 textDump Class Reference

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

Inheritance diagram for textDump:



Public Member Functions

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

4.17.1 Detailed Description

Definition at line 14 of file textDump.h.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 textDump() `textDump::textDump ( )`

Definition at line 9 of file textDump.cc.

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

4.17.3 Member Function Documentation

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

Definition at line 33 of file textDump.cc.

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

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

Definition at line 24 of file textDump.h.

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

4.17.3.3 processDIF()

```
void textDump::processDIF (
            const PayloadParser & 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()); }
```

4.17.3.4 processFrame()

```
void textDump::processFrame (
            const PayloadParser & 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 }
```

4.17.3.5 processPadInFrame()

```
void textDump::processPadInFrame (
            const PayloadParser & 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 }
```

4.17.3.6 processSlowControl()

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

Definition at line 31 of file textDump.cc.

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

4.17.3.7 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); }
```

4.17.3.8 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

4.18 Timer Class Reference

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

Public Member Functions

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

4.18.1 Detailed Description

Definition at line 9 of file Timer.h.

4.18.2 Member Function Documentation

4.18.2.1 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(); }
```

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

Definition at line 12 of file Timer.h.

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

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

Definition at line 13 of file Timer.h.

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

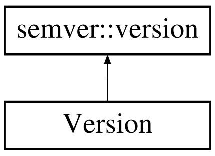
The documentation for this class was generated from the following file:

- libs/core/include/Timer.h

4.19 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 ()

4.19.1 Detailed Description

Definition at line 11 of file Version.h.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 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) {}
```

4.19.2.2 Version() [2/3]

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

Definition at line 15 of file Version.h.

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

4.19.2.3 Version() [3/3]

```
Version::Version ( ) [default]
```

4.19.3 Member Function Documentation

4.19.3.1 getMajor()

```
std::uint8_t Version::getMajor ( )
```

Definition at line 9 of file Version.cc.

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

4.19.3.2 getMinor()

```
std::uint8_t Version::getMinor ( )
```

Definition at line 11 of file Version.cc.

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

4.19.3.3 getPatch()

```
std::uint8_t Version::getPatch ( )
```

Definition at line 13 of file Version.cc.

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

4.19.3.4 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 }
```

4.19.3.5 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.

```
#pragma once

#include "Bits.h"

#include <array>
#include <string>
#include <vector>

class Buffer
{
public:
  Buffer() :  m_Buffer(nullptr), m_Size(0), m_Capacity(0) {}
  virtual ~Buffer() {}
  Buffer(const bit8_t b[], const std::size_t& i) :  m_Buffer(const_cast<bit8_t*>(&b[0])), m_Size(i),
m_Capacity(i) {}
  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)) {}
  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)) {}
  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)) {}

  std::size_t size()const { return m_Size; }
  std::size_t capacity()const { return m_Capacity; }

  bool empty() { return m_Size == 0; }
  void set(unsigned char* b) { m_Buffer = b; }
  void set(const Buffer& buffer)
  {
    m_Buffer   = buffer.begin();
    m_Size     = buffer.size();
    m_Capacity = buffer.capacity();
  }
  bit8_t* begin()const { return m_Buffer; }
  bit8_t* end()const { return m_Buffer + m_Size; }
  bit8_t& operator[](const std::size_t& pos) { return m_Buffer[pos]; }
  bit8_t& operator[](const std::size_t& pos)const { return m_Buffer[pos]; }

  void setSize(const std::size_t& size) { m_Size = size; }

private:
  bit8_t*     m_Buffer{nullptr};
  std::size_t m_Size{0};
  std::size_t m_Capacity{0};
};
```

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 "PayloadParser.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.

```

#pragma once

#include "AppVersion.h"
#include "Buffer.h"
#include "BufferLooperCounter.h"
#include "DetectorId.h"
#include "Formatters.h"
#include "PayloadParser.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>
// function to loop on buffers

```

```
template<typename SOURCE, typename DESTINATION> class BufferLooper
{
public:
  BufferLooper(SOURCE& source, DESTINATION& dest, bool debug = false) :  m_Source(source),
m_Destination(dest), m_Debug(debug)
  {
    m_Logger = spdlog::create<spdlog::sinks::null_sink_mt>("streamout");
    if(!spdlog::get("streamout")) { spdlog::register_logger(m_Logger); }
    m_Source.setLogger(m_Logger);
    m_Destination.setLogger(m_Logger);
  }

  void addSink(const spdlog::sink_ptr& sink, const spdlog::level::level_enum& level =
spdlog::get_level())
  {
    sink->set_level(level);
    m_Sinks.push_back(sink);
    m_Logger = std::make_shared<spdlog::logger>("streamout", begin(m_Sinks), end(m_Sinks));
    m_Source.setLogger(m_Logger);
    m_Destination.setLogger(m_Logger);
  }

  void loop(const std::uint32_t& m_NbrEventsToProcess = 0)
  {
    // clang-format off
    fmt::print(fg(fmt::color::medium_orchid) | fmt::emphasis::bold,
            "\n"
" SSSSSSSSSSSSSSS      tttt
tttt\n"
"SS:::::::::::::::S  ttt:::t
ttt:::t\n"
"S:::::SSSSSS::::::S  t:::::t
t:::::t\n"
"S:::::S     SSSSSSS  t:::::t
t:::::t\n"
"S:::::S        ttttttt:::::ttttttt   rrrrr   rrrrrrrrr       eeeeeeeeeeee    aaaaaaaaaaaaa
mmmmmmm    mmmmmmm      ooooooooooo   uuuuuu    uuuuuuttttttt:::::ttttttt\n"
"S:::::S        t:::::::::::::::::t   r::::rrr:::::::::r    ee::::::::::::ee  a::::::::::::a
mm:::::::m  m:::::::mm  oo:::::::::::oo u::::u    u::::ut:::::::::::::::::t\n"
" S::::SSSS     t:::::::::::::::::t   r:::::::::::::::::r  e::::::eeeee:::::eeaaaaaaaaa:::::a
m::::::::::mm::::::::::mo:::::::::::::::ou::::u    u::::ut:::::::::::::::::t\n"
"  SS::::::SSSSStttttt:::::::tttttt   rr::::::rrrrr::::::re::::::e     e:::::e         a::::a
m::::::::::::::::::::::mo:::::ooooo:::::ou::::u    u::::utttttt:::::::tttttt\n"
"    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"
"       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"
"            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"
"            S:::::S  t:::::t    tttttttr:::::r           e:::::::e         a::::a    a:::::a m::::m
m::::m   m::::mo::::o     o::::ou:::::uuuu:::::u      t:::::t    tttttt\n"
"SSSSSSS     S:::::S  t::::::tttt:::::tr:::::r           e::::::::e        a::::a    a:::::a m::::m
m::::m   m::::mo:::::ooooo:::::ou:::::::::::::::uu    t::::::tttt:::::t\n"
"S::::::SSSSSS:::::S  tt::::::::::::::tr:::::r            e::::::::eeeeeeeea:::::aaaa::::::a m::::m
m::::m   m::::mo:::::::::::::::::o u:::::::::::::::u    tt::::::::::::::t\n"
"S:::::::::::::::SS     tt:::::::::::ttr:::::r             ee:::::::::::::e a::::::::::aa:::am::::m
m::::m   m::::m oo:::::::::::oo   uu::::::::uu:::u      tt:::::::::::tt\n"
" SSSSSSSSSSSSSSS          ttttttttttt  rrrrrrr               eeeeeeeeeeeeee  aaaaaaaaaa  aaaammmmmm
mmmmmm   mmmmmm   ooooooooooo       uuuuuuuu  uuuu        ttttttttttt {}\n"
"\n",
fmt::format(fg(fmt::color::red) | fmt::emphasis::bold, "v{}", streamout_version.to_string()));
    // clang-format on
    log()->info("*******************************************************************");
    log()->info("Streamout Version :  {}", streamout_version.to_string());
    log()->info("Using InterfaceReader {} version {}", m_Source.getName(),
m_Source.getVersion().to_string());
    log()->info("Using InterfaceWriter {} version {}", m_Destination.getName(),
m_Destination.getVersion().to_string());

    if(!m_Destination.checkCompatibility(m_Source.getName(), m_Source.getVersion().to_string()))
    {
      log()->critical("{} version {} is not compatible with {} version {} !  ", m_Source.getName(),
m_Source.getVersion().to_string(), m_Destination.getName(), m_Destination.getVersion().to_string());
      log()->info("Compatible Interfaces for {} are", m_Destination.getName());
      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); }
      std::exit(-1);
    }
    if(!m_DetectorIDs.empty())
    {
      std::string ids;
      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)) + ";";
      log()->info("Detector ID(s) other than {} will be ignored", ids);
    }
    log()->info("*******************************************************************");
    RawBufferNavigator bufferNavigator;
    Timer               timer;
    timer.start();
    m_Source.start();
    m_Destination.start();
    while(m_Source.nextEvent() && m_NbrEventsToProcess >= m_NbrEvents)
    {
      m_Source.startEvent();
      m_Destination.startEvent();

      m_Logger->warn("===*** Event {} ***===", m_NbrEvents);
      while(m_Source.nextDIFbuffer())
      {
        const Buffer& buffer = m_Source.getBuffer();

        bufferNavigator.setBuffer(buffer);
        if(std::find(m_DetectorIDs.begin(), m_DetectorIDs.end(),
static_cast<DetectorID>(bufferNavigator.getDetectorID())) == m_DetectorIDs.end())
        {
          m_Logger->debug("Ignoring detector ID : {}", bufferNavigator.getDetectorID());
          continue;
        }

        std::int32_t idstart = bufferNavigator.getStartOfPayload();
        if(m_Debug && idstart == -1) m_Logger->info(to_hex(buffer));
        c.DIFStarter[idstart]++;
        if(!bufferNavigator.validPayload())
        {
          m_Logger->error("!bufferNavigator.validBuffer()");
          continue;
        }

        m_Source.startDIF();
        m_Destination.startDIF();
        PayloadParser d;
        // This is really a big error so skip DIF entirely if exception occurs
        try
        {
          d.setBuffer(bufferNavigator.getPayload());
        }
        catch(const Exception& e)
        {
          m_Logger->error("{}", e.what());
          continue;
        }
        bit8_t* debug_variable_1 = buffer.end();
        bit8_t* debug_variable_2 = d.end();
        if(debug_variable_1 != debug_variable_2) m_Logger->error("DIF BUFFER END {} {}",
fmt::ptr(debug_variable_1), fmt::ptr(debug_variable_2));
        if(m_Debug) assert(debug_variable_1 == debug_variable_2);

        c.DIFPtrValueAtReturnedPos[d.begin()[d.getGetFramePtrReturn()]]++;
        if(m_Debug) assert(d.begin()[d.getGetFramePtrReturn()] == 0xa0);
        c.SizeAfterDIFPtr[d.getSizeAfterDIFPtr()]++;
        m_Destination.processDIF(d);
        for(std::size_t i = 0; i < d.getNumberOfFrames(); ++i)
        {
          m_Source.startFrame();
          m_Destination.startFrame();
          m_Destination.processFrame(d, i);
          for(std::size_t j = 0; j < static_cast<std::size_t>(Hardware::NUMBER_PAD); ++j)
          {
            if(d.getThresholdStatus(i, j) != 0)
            {
              m_Source.startPad();
              m_Destination.startPad();
              m_Destination.processPadInFrame(d, i, j);
              m_Source.endPad();
              m_Destination.endPad();
            }
          }
          m_Source.endFrame();
          m_Destination.endFrame();
        }
        // If I want SlowControl I need to check for it first, If there is an error then it's not a
big deal just continue and say is bad SlowControl
        try
        {
          d.setSCBuffer();
        }
        catch(const Exception& e)
        {
          m_Logger->error("{}", e.what());
        }

        bool processSC = false;
        if(d.hasSlowControl())
        {
          c.hasSlowControl++;
          processSC = true;
        }
        if(d.badSCData())
        {
          c.hasBadSlowControl++;
          processSC = false;
        }
        if(processSC) { m_Destination.processSlowControl(d.getSCBuffer()); }

        Buffer eod = d.getEndOfAllData();
        c.SizeAfterAllData[eod.size()]++;
        bit8_t* debug_variable_3 = eod.end();
        if(debug_variable_1 != debug_variable_3) m_Logger->info("END DATA BUFFER END {} {}",
fmt::ptr(debug_variable_1), fmt::ptr(debug_variable_3));
        if(m_Debug) assert(debug_variable_1 == debug_variable_3);
        if(eod.size() != 0) m_Logger->info("End of Data remaining stuff :  {}", to_hex(eod));

        int nonzeroCount = 0;
        for(bit8_t* it = eod.begin(); it != eod.end(); it++)
          if(static_cast<int>(*it) != 0) nonzeroCount++;
        c.NonZeroValusAtEndOfData[nonzeroCount]++;
        m_Source.endDIF();
        m_Destination.endDIF();
      }  // end of DIF while loop
      m_Logger->warn("===*** Event {} ****===", m_NbrEvents);
      m_NbrEvents++;
      m_Source.endEvent();
      m_Destination.endEvent();
    }  // end of event while loop
    m_Destination.end();
    m_Source.end();
    timer.stop();
    fmt::print(fg(fmt::color::green) | fmt::emphasis::bold, "=== elapsed time {}ms ({}ms/event)
===\n", timer.getElapsedTime() / 1000, timer.getElapsedTime() / (1000 * m_NbrEvents));
  }
  void                            printAllCounters() { c.printAllCounters(); }
  std::shared_ptr<spdlog::logger> log() { return m_Logger; }

  void setDetectorIDs(const std::vector<DetectorID>& detectorIDs) { m_DetectorIDs = detectorIDs; }

private:
  std::vector<DetectorID>         m_DetectorIDs;
  std::shared_ptr<spdlog::logger> m_Logger{nullptr};
  std::vector<spdlog::sink_ptr>   m_Sinks;
  BufferLooperCounter             c;
  SOURCE&                         m_Source{nullptr};
  DESTINATION&                    m_Destination{nullptr};
  bool                            m_Debug{false};
  std::uint32_t                   m_NbrEvents{1};
};
```

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.

```
#pragma once

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

struct BufferLooperCounter
{
public:
  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;

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

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.

```
{
  HARDROC     = 100,
  HARDROC_NEW = 150,
  RUNHEADER   = 255
};
```

5.10 DetectorId.h

Go to the documentation of this file.

```

#pragma once

#include <cstdint>

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

5.11 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)

5.11.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file DIFSlowControl.h.

5.11.2 Function Documentation

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

Definition at line 256 of file DIFSlowControl.cc.

```
{
  std::string ret;
  for(std::map<int, std::map<std::string, int»::const_iterator it = c.cbegin(); it != c.cend(); it++)
  {
    ret += "ASIC " + std::to_string(it->first) + " :\n";
    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";
  }
  return ret;
}
```

5.12 DIFSlowControl.h

Go to the documentation of this file.

```

#pragma once

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

class DIFSlowControl
{
public:

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

  std::uint8_t getDIFId();


  std::map<int, std::map<std::string, int» getChipsMap();


  std::map<std::string, int> getChipSlowControl(const int& asicid);


  int getChipSlowControl(const std::int8_t& asicid, const std::string& param);

  std::map<int, std::map<std::string, int»::const_iterator cbegin()const { return m_MapSC.cbegin(); }

  std::map<int, std::map<std::string, int»::const_iterator cend()const { return m_MapSC.cend(); }

private:
  DIFSlowControl() = delete;
  void FillHR1(const int& header_shift, unsigned char* cbuf);
  void FillHR2(const int& header_shift, unsigned char* cbuf);
  void FillAsicHR1(const std::bitset<72 * 8>& bs);
  void FillAsicHR2(const std::bitset<109 * 8>& bs);

  unsigned int                                m_DIFId{0};
  unsigned int                                m_Version{0};
  unsigned int                                m_AsicType{0};  // asicType_
  unsigned int                                m_NbrAsic{0};
  std::map<int, std::map<std::string, int» m_MapSC;
};

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

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

```
#include <cstdint>
#include <exception>
#include <string>
```

Classes

- class Exception

5.13.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Exception.h.

5.14 Exception.h

Go to the documentation of this file.

```cpp

#pragma once

#include <cstdint>
#include <exception>
#include <string>

class Exception
{
public:
  virtual const char* what() const noexcept { return m_What.c_str(); }
  explicit Exception(const std::string& message) :  m_Message(message) { constructWhat(); }
  Exception(const std::int32_t& error, const std::string& message) :  m_Error(error), m_Message(message) { constructWhat(); }
  std::int32_t error() { return m_Error; }
  std::string  message() { return m_Message; }

private:
  void constructWhat()
  {
    if(m_Error == 0) m_What = m_Message;
    else
      m_What = std::string("Error ") + std::to_string(m_Error) + std::string(" :  ") + m_Message;
  }
  std::string  m_What;
  std::string  m_Message;
  std::int32_t m_Error{0};
};
```

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.

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

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

Definition at line 19 of file Filesystem.cc.

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

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

Definition at line 7 of file Filesystem.cc.

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

5.16 Filesystem.h

Go to the documentation of this file.

```

#pragma once

#include <string>

std::string path(const std::string&);
std::string extension(const std::string&);
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.

```
{ 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.

```
{ 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.

```
{
  std::size_t iend = end;
  if(iend == -1) iend = b.size();
  std::string ret;
  for(std::size_t k = begin; k < iend; k++)
  {
    ret += to_bin(b[k]);
    ret += " - ";
  }
  return ret;
}
```

5.17.2.6 to_dec() [1/5]

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

Definition at line 29 of file Formatters.cc.

```
{ 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.

```
{ 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.

```
{ 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.

```
{ 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.

```
{
  std::size_t iend = end;
  if(iend == -1) iend = b.size();
  std::string ret;
  for(std::size_t k = begin; k < iend; k++)
  {
    ret += to_dec(b[k]);
    ret += " - ";
  }
  return ret;
}
```

5.17.2.11 to_hex() [1/5]

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

Definition at line 50 of file Formatters.cc.

```
{ 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.

```
{ 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.

```
{ 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.

```
{ 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.

```
{
  std::size_t iend = end;
  if(iend == -1) iend = b.size();
  std::string ret;
  for(std::size_t k = begin; k < iend; k++)
  {
    ret += to_hex(b[k]);
    ret += " - ";
  }
  return ret;
}
```

5.17.2.16 to_oct() [1/5]

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

Definition at line 92 of file Formatters.cc.

```
{ 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.

```
{ 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.

```
{ 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.

```
{ 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.

```
{
  std::size_t iend = end;
  if(iend == -1) iend = b.size();
  std::string ret;
  for(std::size_t k = begin; k < iend; k++)
  {
    ret += to_oct(b[k]);
    ret += " - ";
  }
  return ret;
}
```

5.18 Formatters.h

Go to the documentation of this file.

```

#pragma once

#include "Bits.h"

#include <iosfwd>
#include <string>

class Buffer;

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&);
```

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

```
#include "AppVersion.h"
#include "Buffer.h"
#include "Version.h"
#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 31 of file Interface.h.

```
{
  Unknown = 0,
  Reader  = 1,
  Writer  = 2
};
```

5.20 Interface.h

Go to the documentation of this file.

```

#pragma once

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

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

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

class Interface
{
public:
  Interface(const std::string& name, const std::string& version, const InterfaceType& type) : m_Name(name), m_Version(version) {}
  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() { return m_Logger; }
  void                             setLogger(const std::shared_ptr<spdlog::logger>& logger) { m_Logger = logger; }
  std::string                      getName() { return m_Name; }
  Version                          getVersion() { return m_Version; }

private:
  std::shared_ptr<spdlog::logger> m_Logger{nullptr};
  std::string                     m_Name;
  Version                         m_Version;
  InterfaceType                   m_Type{InterfaceType::Unknown};
};

class InterfaceReader :  public Interface
{
public:
  InterfaceReader(const std::string& name, const std::string& version) :  Interface(name, version, InterfaceType::Reader) {}
  virtual ~InterfaceReader() = default;

protected:
  Buffer m_Buffer;
};

class InterfaceWriter :  public Interface
{
public:
  InterfaceWriter(const std::string& name, const std::string& version) :  Interface(name, version, InterfaceType::Writer) {}

  void addCompatibility(const std::string& name, const std::string& version) { m_Compatible[name] = version; }

  std::map<std::string, std::string> getCompatibility() { return m_Compatible; }

  bool checkCompatibility(const std::string& name, const std::string& version)
  {
    if(m_Compatible.find(name) != m_Compatible.end())
    {
      auto            ran = semver::range::detail::range(m_Compatible[name]);
      semver::version ver = semver::version(version);
      if(ran.satisfies(ver, false)) return true;
      else
        return false;
    }
    else
      return false;
  }

  virtual ~InterfaceWriter() = default;

private:
  std::map<std::string, std::string> m_Compatible;
};
```

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

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

Classes

- class PayloadParser

5.21.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file PayloadParser.h.

5.22 PayloadParser.h

Go to the documentation of this file.

```

#pragma once

#include "Bits.h"
#include "Buffer.h"
#include "Exception.h"
#include "Formatters.h"
#include "Utilities.h"
#include "Words.h"

#include <cstdint>
#include <spdlog/spdlog.h>
#include <string>
#include <vector>

class PayloadParser :  public Buffer
{
public:
  PayloadParser() = default;

  void setBuffer(const Buffer& buffer);

  bool hasTemperature() const;

  bool hasAnalogReadout() const;

  bool hasSlowControl() const;

  float getTemperatureDIF() const;

  float getTemperatureASU1() const;

  float getTemperatureASU2() const;

  Buffer getSlowControl() const;

  std::vector<bit8_t*>& getFramesVector();

  std::vector<bit8_t*>& getLinesVector();

  std::uint32_t getSizeAfterDIFPtr() { return size() - getGetFramePtrReturn(); }
  std::uint32_t getEndOfDIFData()const { return getGetFramePtrReturn() + 3; }
  bool          badSCData()
  {
    setSCBuffer();
    return m_BadSlowControl;
  }
  std::uint32_t getGetFramePtrReturn() const;
  std::uint32_t getDTC() const;
  std::uint32_t getGTC() const;
  std::uint64_t getAbsoluteBCID() const;
  std::uint32_t getBCID() const;
  bool          hasLine(const std::uint32_t&) const;
  std::uint32_t getNumberOfFrames() const;
  bit8_t*       getFramePtr(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;
  // Addition by GG
  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;
  Buffer        getSCBuffer()
  {
    setSCBuffer();
    return m_SCbuffer;
  }
  Buffer getEndOfAllData()
  {
    setSCBuffer();
    if(hasSlowControl() && !m_BadSlowControl) { return
Buffer(&(m_SCbuffer.begin()[m_SCbuffer.size()]), getSizeAfterDIFPtr() - 3 - m_SCbuffer.size()); }
    else
      return Buffer(&(begin()[getEndOfDIFData()]), getSizeAfterDIFPtr() - 3);  // remove the 2 bytes
for CRC and the DIF trailer
  }
  std::uint32_t getDIF_CRC()
  {
    uint32_t i{getEndOfDIFData()};
    uint32_t ret{0};
    ret |= ((begin()[i - 2]) « 8);
    ret |= begin()[i - 1];
    return ret;
  }
  void setSCBuffer()
  {
    if(!hasSlowControl()) return;
    if(m_SCbuffer.size() != 0) return;  // deja fait
    if(m_BadSlowControl) return;
    m_SCbuffer.set(&(begin()[getEndOfDIFData()]));
    // compute Slow Control size
    std::size_t maxsize{size() - getEndOfDIFData() + 1};  // should I +1 here ?
    uint32_t    k{1};                                     // SC Header
    uint32_t    dif_ID{m_SCbuffer[1]};
    uint32_t    chipSize{m_SCbuffer[3]};
    while((dif_ID != 0xa1 && m_SCbuffer[k] != 0xa1 && k < maxsize) || (dif_ID == 0xa1 && m_SCbuffer[k
+ 2] == chipSize && k < maxsize))
    {
      k += 2;  // DIF ID + ASIC Header
      uint32_t scsize = m_SCbuffer[k];
      if(scsize != 74 && scsize != 109)
      {
        k                = 0;
        m_BadSlowControl = true;
        throw Exception(fmt::format("PROBLEM WITH SC SIZE {}", scsize));
      }
      k++;         // skip size bit
      k += scsize;  // skip the data
    }
    if(m_SCbuffer[k] == 0xa1 && !m_BadSlowControl) m_SCbuffer.setSize(k + 1);  // add the trailer
    else
    {
      m_BadSlowControl = true;
      throw Exception(fmt::format("PROBLEM SC TRAILER NOT FOUND "));
    }
  }

private:
  std::uint16_t m_Version{13};
  std::uint32_t parsePayload();
  std::uint32_t getNumberLines() const;
  std::uint32_t parseAnalogLine(const std::uint32_t& idx);
  std::uint32_t getTASU1() const;
  std::uint32_t getTASU2() const;
  std::uint32_t getTDIF() const;

  std::vector<bit8_t*> m_Lines;
  std::vector<bit8_t*> m_Frames;
  bool                 m_BadSlowControl{false};

  std::uint32_t theGetFramePtrReturn_{0};

  Buffer m_SCbuffer;
};

inline void PayloadParser::setBuffer(const Buffer& buffer)
{
  set(buffer);
  m_Frames.clear();
  m_Lines.clear();
  theGetFramePtrReturn_ = parsePayload();
  if(theGetFramePtrReturn_ != size()) { fmt::print("****************************\n"); }
  m_BadSlowControl = false;
}

inline std::uint32_t PayloadParser::parsePayload()
{
  std::uint32_t fshift{static_cast<std::uint32_t>(Size::GLOBAL_HEADER)};  // Pass Global Header
  if(m_Version >= 13)
  {
    // Pass DIF_ID, DIF Trigger counter, Information counter, Global Trigger counter, Absolute BCID, BCID DIF, NB line
    fshift += Size::DIF_IF + Size::DIF_TRIGGER_COUNTER + Size::INFORMATION_COUNTER + Size::GLOBAL_TRIGGER_COUNTER + Size::ABSOLUTE_BCID + Size::BCID_DIF + Size::NUMBER_LINE;
    // If has temperature infos then pass Temp ASU 1, Temp ASU 2, Temp DIF
    if(hasTemperature()) fshift += Size::TEMP_ASU1 + Size::TEMP_ASU2 + Size::TEMP_DIF;
    // If has AnalogReadout pass them
    if(hasAnalogReadout()) fshift = parseAnalogLine(fshift);  // to be implemented
  }
  else
    throw Exception(fmt::format("Version {} is not implemented", m_Version));

  while(static_cast<std::uint8_t>(begin()[fshift]) != static_cast<std::uint8_t>(Value::GLOBAL_TRAILER))
  {
    // If I found a FRAME_HEADER there is 2 cases :
    // 1) Nothing inside so FRAME_TRAILER comes just after
    // 2) Come MICROROC Header, BCID, DATA max 128 times
    if(static_cast<std::uint8_t>(begin()[fshift]) == static_cast<std::uint8_t>(Value::FRAME_HEADER))
    {
      fshift += +Size::FRAME_HEADER;
      if(static_cast<std::uint8_t>(begin()[fshift]) == static_cast<std::uint8_t>(Value::FRAME_TRAILER) || static_cast<std::uint8_t>(begin()[fshift]) == static_cast<std::uint8_t>(Value::FRAME_TRAILER_ERROR)) { fshift += +Size::FRAME_TRAILER; }
      else
      {
        while(static_cast<std::uint8_t>(begin()[fshift]) != static_cast<std::uint8_t>(Value::FRAME_TRAILER) && static_cast<std::uint8_t>(begin()[fshift]) != static_cast<std::uint8_t>(Value::FRAME_TRAILER_ERROR))
        {
          m_Frames.push_back(&begin()[fshift]);
          fshift += Size::MICROROC_HEADER + Size::BCID + Size::DATA;
        }
        fshift += +Size::FRAME_TRAILER;
      }
    }
  }
  // Pass Global trailer
  fshift += +Size::GLOBAL_TRAILER;
  // Pass CRC MSB, CRC LSB
  fshift += Size::CRC_MSB + Size::CRC_LSB;
  return fshift;
}

inline bool PayloadParser::hasTemperature()const { return (static_cast<std::uint8_t>(begin()[0]) == static_cast<std::uint8_t>(Value::GLOBAL_HEADER_TEMP)); }

inline bool PayloadParser::hasAnalogReadout()const { return getNumberLines() != 0; }

inline std::uint32_t PayloadParser::getNumberLines()const
{
  std::uint32_t shift{Size::GLOBAL_HEADER + Size::DIF_IF + Size::DIF_TRIGGER_COUNTER + Size::INFORMATION_COUNTER + Size::GLOBAL_TRIGGER_COUNTER + Size::ABSOLUTE_BCID + Size::BCID_DIF};
  return ((begin()[shift] » 4) & 0x5);
}
```

```

inline std::uint32_t PayloadParser::parseAnalogLine(const std::uint32_t& idx)
{
  std::uint32_t fshift{idx};
  // Pass Header line
  if(static_cast<std::uint8_t>(begin()[fshift]) != static_cast<std::uint8_t>(Value::HEADER_LINE)) return fshift;
  else
    fshift += +Size::HEADER_LINE;
  while(static_cast<std::uint8_t>(begin()[fshift]) != static_cast<std::uint8_t>(Value::TRAILER_LINE))
  {
    m_Lines.push_back(&begin()[fshift]);
    // Get Number of CHIPS
    std::uint32_t nchip{begin()[fshift]};
    // Pass Number of CHIPS, NB Asicline*64*16bits
    fshift += +Size::NUMBER_CHIPS + static_cast<std::uint32_t>(Size::LINE_SIZE) * nchip;
  }
  // Pass Trailer line
  fshift += +Size::TRAILER_LINE;
  return fshift;
}

inline bool PayloadParser::hasSlowControl()const { return theGetFramePtrReturn_ != size(); }

inline std::uint32_t PayloadParser::getTASU1()const
{
  std::uint32_t shift{Size::GLOBAL_HEADER + Size::DIF_IF + Size::DIF_TRIGGER_COUNTER + Size::INFORMATION_COUNTER + Size::GLOBAL_TRIGGER_COUNTER + Size::ABSOLUTE_BCID + Size::BCID_DIF + Size::NUMBER_LINE};
  return (begin()[shift] « 24) + (begin()[shift + 1] « 16) + (begin()[shift + 2] « 8) + begin()[shift + 3];
}

inline std::uint32_t PayloadParser::getTASU2()const
{
  std::uint32_t shift{Size::GLOBAL_HEADER + Size::DIF_IF + Size::DIF_TRIGGER_COUNTER + Size::INFORMATION_COUNTER + Size::GLOBAL_TRIGGER_COUNTER + Size::ABSOLUTE_BCID + Size::BCID_DIF + Size::NUMBER_LINE + Size::TEMP_ASU1};
  return (begin()[shift] « 24) + (begin()[shift + 1] « 16) + (begin()[shift + 2] « 8) + begin()[shift + 3];
}

inline std::uint32_t PayloadParser::getTDIF()const
{
  std::uint32_t shift{Size::GLOBAL_HEADER + Size::DIF_IF + Size::DIF_TRIGGER_COUNTER + Size::INFORMATION_COUNTER + Size::GLOBAL_TRIGGER_COUNTER + Size::ABSOLUTE_BCID + Size::BCID_DIF + Size::NUMBER_LINE + Size::TEMP_ASU1 + Size::TEMP_ASU2};
  return begin()[shift];
}

inline float PayloadParser::getTemperatureDIF()const
{
  if(!hasTemperature()) throw Exception("Don't have TemperatureDIF information");
  return 0.508 * getTDIF() - 9.659;
}

inline float PayloadParser::getTemperatureASU1()const
{
  if(!hasTemperature()) throw Exception("Don't have TemperatureASU1 information");
  return (getTASU1() » 3) * 0.0625;
}

inline float PayloadParser::getTemperatureASU2()const
{
  if(!hasTemperature()) throw Exception("Don't have TemperatureASU2 information");
  return (getTASU2() » 3) * 0.0625;
}

inline Buffer PayloadParser::getSlowControl()const
{
  if(hasSlowControl()) return Buffer(&begin()[getEndOfDIFData()], size() - getEndOfDIFData());
  else
    return Buffer();
}

inline std::vector<bit8_t*>& PayloadParser::getFramesVector() { return m_Frames; }

inline std::vector<bit8_t*>& PayloadParser::getLinesVector() { return m_Lines; }

inline std::uint32_t PayloadParser::getGetFramePtrReturn()const { return theGetFramePtrReturn_ - 3; }

inline std::uint32_t PayloadParser::getDTC()const
{
  std::uint32_t shift{Size::GLOBAL_HEADER + Size::DIF_IF};
  return (begin()[shift] « 24) + (begin()[shift + 1] « 16) + (begin()[shift + 2] « 8) + begin()[shift + 3];
}
```

```

inline std::uint32_t PayloadParser::getGTC()const
{
  std::uint32_t shift{Size::GLOBAL_HEADER + Size::DIF_IF + Size::DIF_TRIGGER_COUNTER + Size::INFORMATION_COUNTER};
  return (begin()[shift] « 24) + (begin()[shift + 1] « 16) + (begin()[shift + 2] « 8) + begin()[shift + 3];
}

inline std::uint64_t PayloadParser::getAbsoluteBCID()const
{
  std::uint32_t shift{Size::GLOBAL_HEADER + Size::DIF_IF + Size::DIF_TRIGGER_COUNTER + Size::INFORMATION_COUNTER + Size::GLOBAL_TRIGGER_COUNTER};
  std::uint64_t LBC = ((begin()[shift] « 16) | (begin()[shift + 1] « 8) | (begin()[shift + 2])) * 16777216ULL /* to shift the value from the 24 first bits*/
                    + ((begin()[shift + 3] « 16) | (begin()[shift + 4] « 8) | (begin()[shift + 5]));
  return LBC;
}

inline std::uint32_t PayloadParser::getBCID()const
{
  std::uint32_t shift{Size::GLOBAL_HEADER + Size::DIF_IF + Size::DIF_TRIGGER_COUNTER + Size::INFORMATION_COUNTER + Size::GLOBAL_TRIGGER_COUNTER + Size::ABSOLUTE_BCID};
  return (begin()[shift] « 16) + (begin()[shift + 1] « 8) + begin()[shift + 2];
}

inline bool PayloadParser::hasLine(const std::uint32_t& line)const
{
  std::uint32_t shift{Size::GLOBAL_HEADER + Size::DIF_IF + Size::DIF_TRIGGER_COUNTER + Size::INFORMATION_COUNTER + Size::GLOBAL_TRIGGER_COUNTER + Size::ABSOLUTE_BCID + Size::BCID_DIF};
  return ((begin()[shift] » line) & 0x1);
}

inline std::uint32_t PayloadParser::getNumberOfFrames()const { return m_Frames.size(); }

inline bit8_t* PayloadParser::getFramePtr(const std::uint32_t& i)const { return m_Frames[i]; }

inline std::uint32_t PayloadParser::getFrameBCID(const std::uint32_t& i)const
{
  std::uint32_t shift{+Size::MICROROC_HEADER};
  return GrayToBin((m_Frames[i][shift] « 16) + (m_Frames[i][shift + 1] « 8) + m_Frames[i][shift + 2]);
}

inline std::uint32_t PayloadParser::getFrameTimeToTrigger(const std::uint32_t& i)const { return getBCID() - getFrameBCID(i); }

inline bool PayloadParser::getFrameLevel(const std::uint32_t& i, const std::uint32_t& ipad, const std::uint32_t& ilevel)const
{
  std::uint32_t shift{Size::MICROROC_HEADER + Size::BCID};
  return ((m_Frames[i][shift + ((3 - ipad / 16) * 4 + (ipad % 16) / 4)] » (7 - (((ipad % 16) % 4) * 2 + ilevel))) & 0x1);
}

inline uint32_t PayloadParser::getDIFid()const
{
  std::uint32_t shift{+Size::GLOBAL_HEADER};
  return begin()[shift] & 0xFF;
}

inline uint32_t PayloadParser::getASICid(const std::uint32_t& i)const { return m_Frames[i][0] & 0xFF; }

inline uint32_t PayloadParser::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)); }
```

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

`#include "Buffer.h"`

Classes

- class RawBufferNavigator

  class to navigate in the raw data buffer parse the header and send the payload as Buffer

5.23.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file RawBufferNavigator.h.

5.24 RawBufferNavigator.h

Go to the documentation of this file.

```

#pragma once

#include "Buffer.h"

class RawBufferNavigator
{
public:
  static void StartAt(const int& start);
  RawBufferNavigator();
  ~RawBufferNavigator() = default;
  void          setBuffer(const Buffer&);
  std::uint8_t getDetectorID();
  bool          findStartOfPayload();
  std::int32_t getStartOfPayload();
  bool          validPayload();
  Buffer        getPayload();

private:
  static int    m_Start;
  Buffer        m_Buffer;
  bool          m_StartPayloadDone{false};
  std::int32_t m_StartPayload{-1};  // -1 Means not found !
};
```

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

`#include <chrono>`

Classes

- class Timer

5.25.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Timer.h.

5.26 Timer.h

Go to the documentation of this file.

```
#pragma once

#include <chrono>

class Timer
{
public:
  void  start() { m_StartTime = std::chrono::high_resolution_clock::now(); }
  void  stop() { m_StopTime = std::chrono::high_resolution_clock::now(); }
  float getElapsedTime() { return std::chrono::duration_cast<std::chrono::microseconds>(m_StopTime -
m_StartTime).count(); }

private:
  std::chrono::time_point<std::chrono::high_resolution_clock> m_StartTime;
  std::chrono::time_point<std::chrono::high_resolution_clock> m_StopTime;
};
```

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

`#include <cstdint>`

Functions

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

5.27.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Utilities.h.

5.27.2 Function Documentation

5.27.2.1 GrayToBin()

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

Definition at line 9 of file Utilities.h.

```
{
  std::uint64_t ish{1};
  std::uint64_t anss{n};
  std::uint64_t idiv{0};
  std::uint64_t ishmax{sizeof(std::uint64_t) * 8};
  while(true)
  {
    idiv = anss » ish;
    anss ^= idiv;
    if(idiv <= 1 || ish == ishmax) return anss;
    ish «= 1;
  }
}
```

5.28 Utilities.h

Go to the documentation of this file.

```
#pragma once

#include <cstdint>

inline std::uint64_t GrayToBin(const std::uint64_t& n)
{
  std::uint64_t ish{1};
  std::uint64_t anss{n};
  std::uint64_t idiv{0};
  std::uint64_t ishmax{sizeof(std::uint64_t) * 8};
  while(true)
  {
    idiv = anss » ish;
    anss ^= idiv;
    if(idiv <= 1 || ish == ishmax) return anss;
    ish «= 1;
  }
}
```

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

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

Classes

- class Version

5.29.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Version.h.

5.30 Version.h

Go to the documentation of this file.

```
#pragma once

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

class Version :  public semver::version
{
public:
  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) {}
  explicit Version(const std::string_view& str) :  semver::version(str) {}
  Version() = default;
  std::uint8_t getMajor();
  std::uint8_t getMinor();
  std::uint8_t getPatch();
  std::string  getPreRelease();
  std::uint8_t getPreReleaseNumber();
};
```

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

```
#include <cstdint>
```

Enumerations

- enum class Hardware : std::uint8_t { NUMBER_PAD = 64 }
- enum class Size : std::uint8_t {
  DATA_FORMAT_VERSION = 1 , DAQ_SOFTWARE_VERSION = 2 , SDCC_FIRMWARE_VERSION = 2 , DIF_FIRMWARE_VERSION = 2 ,
  TIMESTAMP_SECONDES = 4 , TIMESTAMP_MILLISECONDS = 4 , GLOBAL_HEADER = 1 , DIF_IF = 1 , DIF_TRIGGER_COUNTER = 4 , INFORMATION_COUNTER = 4 , GLOBAL_TRIGGER_COUNTER = 4 , ABSOLUTE_BCID = 6 ,
  BCID_DIF = 3 , NUMBER_LINE = 1 , TEMP_ASU1 = 4 , TEMP_ASU2 = 4 ,
  TEMP_DIF = 1 , HEADER_LINE = 1 , NUMBER_CHIPS = 1 , LINE_SIZE = 64 $*$ 2 ,
  TRAILER_LINE = 1 , FRAME_HEADER = 1 , MICROROC_HEADER = 1 , BCID = 3 ,
  DATA = 16 , FRAME_TRAILER = 1 , GLOBAL_TRAILER = 1 , CRC_MSB = 1 ,
  CRC_LSB = 1 , SC_HEADER = 1 , DIF_ID = 1 , ASIC_HEADER = 1 ,
  SC_ASIC_SIZE = 1 , SC_TRAILER = 1 }
- enum class Value : std::uint8_t {
  GLOBAL_HEADER = 0xb0 , GLOBAL_HEADER_TEMP = 0xbb , HEADER_LINE = 0xc4 , TRAILER_LINE = 0xd4 ,
  FRAME_HEADER = 0xb4 , FRAME_TRAILER = 0xa3 , FRAME_TRAILER_ERROR = 0xc3 , GLOBAL_TRAILER = 0xa0 ,
  SC_HEADER = 0xb1 , SC_TRAILER = 0xa1 }

5.31.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Words.h.

5.31.2 Enumeration Type Documentation

5.31.2.1 Hardware `enum class Hardware : std::uint8_t [strong]`

Enumerator

NUMBER_PAD	

Definition at line 9 of file Words.h.

```
00010 {
00011   NUMBER_PAD = 64,
00012 };
```

5.31.2.2 Size `enum class Size : std::uint8_t [strong]`

Enumerator

DATA_FORMAT_VERSION	
DAQ_SOFTWARE_VERSION	
SDCC_FIRMWARE_VERSION	
DIF_FIRMWARE_VERSION	
TIMESTAMP_SECONDES	
TIMESTAMP_MILLISECONDS	
GLOBAL_HEADER	
DIF_IF	
DIF_TRIGGER_COUNTER	
INFORMATION_COUNTER	
GLOBAL_TRIGGER_COUNTER	
ABSOLUTE_BCID	
BCID_DIF	
NUMBER_LINE	
TEMP_ASU1	
TEMP_ASU2	
TEMP_DIF	
HEADER_LINE	
NUMBER_CHIPS	
LINE_SIZE	
TRAILER_LINE	
FRAME_HEADER	
MICROROC_HEADER	
BCID	
DATA	
FRAME_TRAILER	
GLOBAL_TRAILER	
CRC_MSB	
CRC_LSB	
SC_HEADER	
DIF_ID	
ASIC_HEADER	
SC_ASIC_SIZE	
SC_TRAILER	

Definition at line 14 of file Words.h.

```
{
  // Header
  DATA_FORMAT_VERSION    = 1,
  DAQ_SOFTWARE_VERSION   = 2,
  SDCC_FIRMWARE_VERSION  = 2,
  DIF_FIRMWARE_VERSION   = 2,
  TIMESTAMP_SECONDES     = 4,
  TIMESTAMP_MILLISECONDS = 4,
  // Payload
  GLOBAL_HEADER          = 1,
  DIF_IF                 = 1,
  DIF_TRIGGER_COUNTER    = 4,
  INFORMATION_COUNTER    = 4,
  GLOBAL_TRIGGER_COUNTER = 4,
  ABSOLUTE_BCID          = 6,
  BCID_DIF               = 3,
  NUMBER_LINE            = 1,
  TEMP_ASU1              = 4,
  TEMP_ASU2              = 4,
00034   TEMP_DIF                = 1,
00035   HEADER_LINE             = 1,
00036   NUMBER_CHIPS            = 1,
00037   LINE_SIZE               = 64 * 2,
00038   TRAILER_LINE            = 1,
00039   FRAME_HEADER            = 1,
00040   MICROROC_HEADER         = 1,
00041   BCID                    = 3,
00042   DATA                    = 16,
00043   FRAME_TRAILER           = 1,
00044   GLOBAL_TRAILER          = 1,
00045   CRC_MSB                 = 1,
00046   CRC_LSB                 = 1,
00047   // Slowcontrol
00048   SC_HEADER               = 1,
00049   DIF_ID                  = 1,
00050   ASIC_HEADER             = 1,
00051   SC_ASIC_SIZE            = 1,
00052   SC_TRAILER              = 1
00053 };
```

5.31.2.3 Value `enum class Value : std::uint8_t [strong]`

Enumerator

GLOBAL_HEADER	
GLOBAL_HEADER_TEMP	
HEADER_LINE	
TRAILER_LINE	
FRAME_HEADER	
FRAME_TRAILER	
FRAME_TRAILER_ERROR	
GLOBAL_TRAILER	
SC_HEADER	
SC_TRAILER	

Definition at line 59 of file Words.h.

```
00060 {
00061   GLOBAL_HEADER        = 0xb0,
00062   GLOBAL_HEADER_TEMP   = 0xbb,
00063   HEADER_LINE          = 0xc4,
00064   TRAILER_LINE         = 0xd4,
00065   FRAME_HEADER         = 0xb4,
00066   FRAME_TRAILER        = 0xa3,
00067   FRAME_TRAILER_ERROR = 0xc3,
00068   GLOBAL_TRAILER       = 0xa0,
00069   SC_HEADER            = 0xb1,
00070   SC_TRAILER           = 0xa1
00071 };
```

5.32 Words.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include <cstdint>
00008
00009 enum class Hardware :  std::uint8_t
00010 {
00011   NUMBER_PAD = 64,
00012 };
00013
00014 enum class Size :  std::uint8_t
00015 {
  // Header
  DATA_FORMAT_VERSION     = 1,
  DAQ_SOFTWARE_VERSION    = 2,
  SDCC_FIRMWARE_VERSION   = 2,
  DIF_FIRMWARE_VERSION    = 2,
  TIMESTAMP_SECONDES      = 4,
  TIMESTAMP_MILLISECONDS = 4,
  // Payload
  GLOBAL_HEADER           = 1,
  DIF_IF                  = 1,
  DIF_TRIGGER_COUNTER     = 4,
  INFORMATION_COUNTER     = 4,
  GLOBAL_TRIGGER_COUNTER = 4,
  ABSOLUTE_BCID           = 6,
  BCID_DIF                = 3,
  NUMBER_LINE             = 1,
  TEMP_ASU1               = 4,
  TEMP_ASU2               = 4,
  TEMP_DIF                = 1,
  HEADER_LINE             = 1,
  NUMBER_CHIPS            = 1,
  LINE_SIZE               = 64 * 2,
  TRAILER_LINE            = 1,
  FRAME_HEADER            = 1,
  MICROROC_HEADER         = 1,
  BCID                    = 3,
  DATA                    = 16,
  FRAME_TRAILER           = 1,
  GLOBAL_TRAILER          = 1,
  CRC_MSB                 = 1,
  CRC_LSB                 = 1,
  // Slowcontrol
  SC_HEADER               = 1,
  DIF_ID                  = 1,
  ASIC_HEADER             = 1,
  SC_ASIC_SIZE            = 1,
  SC_TRAILER              = 1
};

static inline std::uint32_t operator+(const Size& a, const Size& b) { return
static_cast<std::uint32_t>(a) + static_cast<std::uint32_t>(b); }
static inline std::uint32_t operator+(const std::uint32_t& a, const Size& b) { return a +
static_cast<std::uint32_t>(b); }
static inline std::uint32_t operator+(const Size& a) { return static_cast<std::uint32_t>(a); }

enum class Value :  std::uint8_t
{
  GLOBAL_HEADER        = 0xb0,
  GLOBAL_HEADER_TEMP   = 0xbb,
  HEADER_LINE          = 0xc4,
  TRAILER_LINE         = 0xd4,
  FRAME_HEADER         = 0xb4,
  FRAME_TRAILER        = 0xa3,
  FRAME_TRAILER_ERROR = 0xc3,
  GLOBAL_TRAILER       = 0xa0,
  SC_HEADER            = 0xb1,
  SC_TRAILER           = 0xa1
};
```

5.33 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.

5.33.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Bits.cc.

5.33.2 Function Documentation

5.33.2.1 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; }
```

5.34 Bits.cc

Go to the documentation of this file.

```
#include "Bits.h"

std::ostream& operator«(std::ostream& os, const bit8_t& c) { return os « c + 0; }
```

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

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

5.36 BufferLooperCounter.cc

Go to the documentation of this file.

```
#include "BufferLooperCounter.h"

#include <fmt/color.h>
#include <fmt/core.h>

void BufferLooperCounter::printAllCounters()
{
  fmt::print(fg(fmt::color::crimson) | fmt::emphasis::bold, "BUFFER LOOP FINAL STATISTICS : \n");
  printCounter("Start of DIF header", DIFStarter);
  printCounter("Value after DIF data are processed", DIFPtrValueAtReturnedPos);
  printCounter("Size remaining in buffer after end of DIF data", SizeAfterDIFPtr);
  fmt::print(fg(fmt::color::crimson) | fmt::emphasis::bold, "Number of Slow Control found {}  out of
which {} are bad\n", hasSlowControl, hasBadSlowControl);
  printCounter("Size remaining after all of data have been processed", SizeAfterAllData);
  printCounter("Number on non zero values in end of data buffer", NonZeroValusAtEndOfData);
}

void BufferLooperCounter::printCounter(const std::string& description, const std::map<int, int>& m)
{
  std::string out{"statistics for " + description + " :  \n"};
  for(std::map<int, int>::const_iterator it = m.begin(); it != m.end(); it++)
  {
    if(it != m.begin()) out += ",";
    out += " [" + std::to_string(it->first) + "]=" + std::to_string(it->second);
  }
  out += "\n";
  fmt::print(fg(fmt::color::crimson) | fmt::emphasis::bold, out);
}
```

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

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

Functions

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

5.37.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file DIFSlowControl.cc.

5.37.2 Function Documentation

5.37.2.1 to_string()

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

Definition at line 256 of file DIFSlowControl.cc.

```
{
  std::string ret;
  for(std::map<int, std::map<std::string, int»::const_iterator it = c.cbegin(); it != c.cend(); it++)
  {
    ret += "ASIC " + std::to_string(it->first) + " :\n";
    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";
  }
  return ret;
}
```

5.38 DIFSlowControl.cc

Go to the documentation of this file.

```

#include "DIFSlowControl.h"

DIFSlowControl::DIFSlowControl(const std::uint8_t& version, const std::uint8_t& DIfId, unsigned char*
cbuf) :  m_Version(version), m_DIFId(DIfId), m_AsicType(2)
{
  if(cbuf[0] != 0xb1) return;
  int header_shift{6};
  if(m_Version < 8) m_NbrAsic = cbuf[5];
  else
  {
    m_DIFId      = cbuf[1];
    m_NbrAsic    = cbuf[2];
    header_shift = 3;
  }
  int size_hardroc1 = m_NbrAsic * 72 + header_shift + 1;
  if(cbuf[size_hardroc1 - 1] != 0xa1) size_hardroc1 = 0;

  int size_hardroc2 = m_NbrAsic * 109 + header_shift + 1;
  if(cbuf[size_hardroc2 - 1] != 0xa1) size_hardroc2 = 0;
  if(size_hardroc1 != 0)
  {
    FillHR1(header_shift, cbuf);
    m_AsicType = 1;
  }
  else if(size_hardroc2 != 0)
    FillHR2(header_shift, cbuf);
  else
    return;
}

inline std::uint8_t DIFSlowControl::getDIFId() { return m_DIFId; }

inline std::map<int, std::map<std::string, int» DIFSlowControl::getChipsMap() { return m_MapSC; }

inline std::map<std::string, int> DIFSlowControl::getChipSlowControl(const int& asicid) { return m_MapSC[asicid]; }

inline int DIFSlowControl::getChipSlowControl(const std::int8_t& asicid, const std::string& param) { return getChipSlowControl(asicid)[param]; }

void DIFSlowControl::FillHR1(const int& header_shift, unsigned char* cbuf)
{
  int nasic{cbuf[header_shift - 1]};
  int idx{header_shift};
  for(int k = 0; k < nasic; k++)
  {
    std::bitset<72 * 8> bs;
    // printf("%x %x \n",cbuf[idx+k*72+69],cbuf[idx+k*72+70]);
    for(int l = 71; l >= 0; l--)
    {
      //  printf("%d %x :  %d -->",l,cbuf[idx+k*72+l],(71-l)*8);
      for(int m = 0; m < 8; m++)
      {
        if(((1 « m) & cbuf[idx + k * 72 + l]) != 0) bs.set((71 - l) * 8 + m, 1);
        else
          bs.set((71 - l) * 8 + m, 0);
        // printf("%d",(int) bs[(71-l)*8+m]);
      }
      // printf("\n");
    }
    FillAsicHR1(bs);
  }
}

void DIFSlowControl::FillHR2(const int& header_shift, unsigned char* cbuf)
{
  // int scsize1=cbuf[header_shift-1]*109+(header_shift-1)+2;
  int nasic{cbuf[header_shift - 1]};
  int idx{header_shift};
  // std::cout«" DIFSlowControl::FillHR nasic "«nasic«std::endl;
  for(int k = 0; k < nasic; k++)
  {
    std::bitset<109 * 8> bs;
    // printf("%x %x \n",cbuf[idx+k*109+69],cbuf[idx+k*109+70]);
    for(int l = 108; l >= 0; l--)
    {
      //  printf("%d %x :  %d -->",l,cbuf[idx+k*109+l],(71-l)*8);
      for(int m = 0; m < 8; m++)
      {
        if(((1 « m) & cbuf[idx + k * 109 + l]) != 0) bs.set((108 - l) * 8 + m, 1);
        else
          bs.set((108 - l) * 8 + m, 0);
        // printf("%d",(int) bs[(71-l)*8+m]);
      }
      // printf("\n");
    }
    FillAsicHR2(bs);
  }
}

void DIFSlowControl::FillAsicHR1(const std::bitset<72 * 8>& bs)
{
  // Asic Id
  int asicid{0};
  for(int j = 0; j < 8; j++)
    if(bs[j + 9] != 0) asicid += (1 « (7 - j));
  std::map<std::string, int> mAsic;
  // Slow Control
  mAsic["SSC0"]          = static_cast<int>(bs[575]);
  mAsic["SSC1"]          = static_cast<int>(bs[574]);
  mAsic["SSC2"]          = static_cast<int>(bs[573]);
  mAsic["Choix_caisson"] = static_cast<int>(bs[572]);
  mAsic["SW_50k"]        = static_cast<int>(bs[571]);
  mAsic["SW_100k"]       = static_cast<int>(bs[570]);
  mAsic["SW_100f"]       = static_cast<int>(bs[569]);
  mAsic["SW_50f"]        = static_cast<int>(bs[568]);

  mAsic["Valid_DC"]  = static_cast<int>(bs[567]);
  mAsic["ON_Discri"] = static_cast<int>(bs[566]);
  mAsic["ON_Fsb"]    = static_cast<int>(bs[565]);
  mAsic["ON_Otaq"]   = static_cast<int>(bs[564]);
  mAsic["ON_W"]      = static_cast<int>(bs[563]);
  mAsic["ON_Ss"]     = static_cast<int>(bs[562]);
  mAsic["ON_Buf"]    = static_cast<int>(bs[561]);
  mAsic["ON_Paf"]    = static_cast<int>(bs[560]);
  // Gain
  for(int i = 0; i < 64; i++)
  {
    int gain{0};
    for(int j = 0; j < 6; j++)
      if(bs[176 + i * 6 + j] != 0) gain += (1 « j);
    mAsic["Channel_" + std::to_string(i) + "_" + "Gain"]       = gain;
    mAsic["Channel_" + std::to_string(i) + "_" + "cTest"]      = bs[112 + i];
    mAsic["Channel_" + std::to_string(i) + "_" + "Valid_trig"] = static_cast<int>(bs[25 + i]);
  }

  mAsic["ON_Otabg"]  = static_cast<int>(bs[111]);
  mAsic["ON_Dac"]    = static_cast<int>(bs[110]);
  mAsic["ON_Otadac"] = static_cast<int>(bs[109]);
  // DAC
  int dac1{0};
  for(int j = 0; j < 10; j++)
    if(bs[j + 99] != 0) dac1 += (1 « j);
  mAsic["DAC1"] = dac1;
  int dac0{0};
  for(int j = 0; j < 10; j++)
    if(bs[j + 89] != 0) dac0 += (1 « j);
  mAsic["DAC0"]            = dac0;
  mAsic["EN_Raz_Ext"]      = static_cast<int>(bs[23]);
  mAsic["EN_Raz_Int"]      = static_cast<int>(bs[22]);
  mAsic["EN_Out_Raz_Int"]  = static_cast<int>(bs[21]);
  mAsic["EN_Trig_Ext"]     = static_cast<int>(bs[20]);
  mAsic["EN_Trig_Int"]     = static_cast<int>(bs[19]);
  mAsic["EN_Out_Trig_Int"] = static_cast<int>(bs[18]);
  mAsic["Bypass_Chip"]     = static_cast<int>(bs[17]);
  mAsic["HardrocHeader"]   = static_cast<int>(asicid);
  mAsic["EN_Out_Discri"]   = static_cast<int>(bs[8]);
  mAsic["EN_Transmit_On"]  = static_cast<int>(bs[7]);
  mAsic["EN_Dout"]         = static_cast<int>(bs[6]);
  mAsic["EN_RamFull"]      = static_cast<int>(bs[5]);
  m_MapSC[asicid]          = mAsic;
}

void DIFSlowControl::FillAsicHR2(const std::bitset<109 * 8>& bs)
{
  int asicid{0};
  for(int j = 0; j < 8; j++)
    if(bs[j + (108 - 7) * 8 + 2] != 0) asicid += (1 « (7 - j));
  std::map<std::string, int> mAsic;
  for(int i = 0; i < 64; i++)
  {
    int gain{0};
    int mask{0};
    mAsic["Channel_" + std::to_string(i) + "_" + "cTest"] = bs[i];
    for(int j = 0; j < 8; j++)
      if(bs[64 + i * 8 + j] != 0) gain += (1 « j);
    mAsic["Channel_" + std::to_string(i) + "_" + "Gain"] = gain;
    for(int j = 0; j < 3; j++)
      if(bs[8 * 77 + 2 + i * 3 + j] != 0) mask += (1 « j);
    mAsic["Channel_" + std::to_string(i) + "_" + "Mask"] = mask;
  }
  mAsic["PwrOnPA"] = static_cast<int>(bs[8 * 72]);
  mAsic["Cmdb3SS"] = static_cast<int>(bs[8 * 72 + 1]);
  mAsic["Cmdb2SS"] = static_cast<int>(bs[8 * 72 + 2]);
  mAsic["Cmdb1SS"] = static_cast<int>(bs[8 * 72 + 3]);
  mAsic["Cmdb0SS"] = static_cast<int>(bs[8 * 72 + 4]);
  mAsic["SwSsc0"]  = static_cast<int>(bs[8 * 72 + 5]);
  mAsic["SwSsc1"]  = static_cast<int>(bs[8 * 72 + 6]);
  mAsic["SwSsc2"]  = static_cast<int>(bs[8 * 72 + 7]);

  mAsic["PwrOnBuff"] = static_cast<int>(bs[8 * 73]);
  mAsic["PwrOnSS"]   = static_cast<int>(bs[8 * 73 + 1]);
  mAsic["PwrOnW"]    = static_cast<int>(bs[8 * 73 + 2]);
  mAsic["Cmdb3Fsb2"] = static_cast<int>(bs[8 * 73 + 3]);
  mAsic["Cmdb2Fsb2"] = static_cast<int>(bs[8 * 73 + 4]);
  mAsic["Cmdb1Fsb2"] = static_cast<int>(bs[8 * 73 + 5]);
  mAsic["Cmdb0Fsb2"] = static_cast<int>(bs[8 * 73 + 6]);
  mAsic["Sw50k2"]    = static_cast<int>(bs[8 * 73 + 7]);

  mAsic["Sw100k2"]   = static_cast<int>(bs[8 * 74]);
  mAsic["Sw100f2"]   = static_cast<int>(bs[8 * 74 + 1]);
  mAsic["Sw50f2"]    = static_cast<int>(bs[8 * 74 + 2]);
  mAsic["Cmdb3Fsb1"] = static_cast<int>(bs[8 * 74 + 3]);
  mAsic["Cmdb2Fsb1"] = static_cast<int>(bs[8 * 74 + 4]);
  mAsic["Cmdb1Fsb1"] = static_cast<int>(bs[8 * 74 + 5]);
  mAsic["Cmdb0Fsb1"] = static_cast<int>(bs[8 * 74 + 6]);
  mAsic["Sw50k1"]    = static_cast<int>(bs[8 * 74 + 7]);

  mAsic["Sw100k1"]   = static_cast<int>(bs[8 * 75]);
  mAsic["Sw100f1"]   = static_cast<int>(bs[8 * 75 + 1]);
  mAsic["Sw50f1"]    = static_cast<int>(bs[8 * 75 + 2]);
  mAsic["Sel0"]      = static_cast<int>(bs[8 * 75 + 3]);
  mAsic["Sel1"]      = static_cast<int>(bs[8 * 75 + 4]);
  mAsic["PwrOnFsb"]  = static_cast<int>(bs[8 * 75 + 5]);
  mAsic["PwrOnFsb1"] = static_cast<int>(bs[8 * 75 + 6]);
  mAsic["PwrOnFsb2"] = static_cast<int>(bs[8 * 75 + 7]);

  mAsic["Sw50k0"]      = static_cast<int>(bs[8 * 76]);
  mAsic["Sw100k0"]     = static_cast<int>(bs[8 * 76 + 1]);
  mAsic["Sw100f0"]     = static_cast<int>(bs[8 * 76 + 2]);
  mAsic["Sw50f0"]      = static_cast<int>(bs[8 * 76 + 3]);
  mAsic["EnOtaQ"]      = static_cast<int>(bs[8 * 76 + 4]);
  mAsic["OtaQ_PwrADC"] = static_cast<int>(bs[8 * 76 + 5]);
  mAsic["Discri_PwrA"] = static_cast<int>(bs[8 * 76 + 6]);
  mAsic["Discri2"]     = static_cast<int>(bs[8 * 76 + 7]);

  mAsic["Discri1"]      = static_cast<int>(bs[8 * 77]);
  mAsic["RS_or_Discri"] = static_cast<int>(bs[8 * 77 + 1]);

  mAsic["Header"] = asicid;
  for(int i = 0; i < 3; i++)
  {
    int B = 0;
    for(int j = 0; j < 10; j++)
      if(bs[8 * 102 + 2 + i * 10 + j] != 0) B += (1 « j);
    mAsic["B" + std::to_string(i)] = B;
  }

  mAsic["Smalldac"]  = static_cast<int>(bs[8 * 106]);
  mAsic["DacSw"]     = static_cast<int>(bs[8 * 106 + 1]);
  mAsic["OtagBgSw"]  = static_cast<int>(bs[8 * 106 + 2]);
  mAsic["Trig2b"]    = static_cast<int>(bs[8 * 106 + 3]);
  mAsic["Trig1b"]    = static_cast<int>(bs[8 * 106 + 4]);
  mAsic["Trig0b"]    = static_cast<int>(bs[8 * 106 + 5]);
  mAsic["EnTrigOut"] = static_cast<int>(bs[8 * 106 + 6]);
  mAsic["DiscrOrOr"] = static_cast<int>(bs[8 * 106 + 7]);

  mAsic["TrigExtVal"]    = static_cast<int>(bs[8 * 107]);
  mAsic["RazChnIntVal"] = static_cast<int>(bs[8 * 107 + 1]);
  mAsic["RazChnExtVal"] = static_cast<int>(bs[8 * 107 + 2]);
  mAsic["ScOn"]         = static_cast<int>(bs[8 * 107 + 3]);
  mAsic["CLKMux"]       = static_cast<int>(bs[8 * 107 + 4]);

  // EnOCDout1b   EnOCDout2b   EnOCTransmitOn1b   EnOCTransmitOn2b   EnOCChipsatb   SelStartReadout
SelEndReadout
  mAsic["SelEndReadout"]    = static_cast<int>(bs[8 * 108 + 1]);
  mAsic["SelStartReadout"]  = static_cast<int>(bs[8 * 108 + 2]);
  mAsic["EnOCChipsatb"]     = static_cast<int>(bs[8 * 108 + 3]);
  mAsic["EnOCTransmitOn2b"] = static_cast<int>(bs[8 * 108 + 4]);
  mAsic["EnOCTransmitOn1b"] = static_cast<int>(bs[8 * 108 + 5]);
  mAsic["EnOCDout2b"]       = static_cast<int>(bs[8 * 108 + 6]);
  mAsic["EnOCDout1b"]       = static_cast<int>(bs[8 * 108 + 7]);
  m_MapSC[asicid]           = mAsic;
}

std::string to_string(const DIFSlowControl& c)
{
  std::string ret;
  for(std::map<int, std::map<std::string, int»::const_iterator it = c.cbegin(); it != c.cend(); it++)
  {
    ret += "ASIC " + std::to_string(it->first) + " :\n";
    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";
  }
  return ret;
}
```

5.39 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)

5.39.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Filesystem.cc.

5.39.2 Function Documentation

5.39.2.1 extension()

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

Definition at line 13 of file Filesystem.cc.

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

5.39.2.2 filename()

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

Definition at line 19 of file Filesystem.cc.

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

5.39.2.3 path()

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

Definition at line 7 of file Filesystem.cc.

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

5.40 Filesystem.cc

Go to the documentation of this file.

```
#include "Filesystem.h"

std::string path(const std::string& file)
{
  std::size_t pos = file.find_last_of("\\/");
  return (std::string::npos == pos) ?  "" :  file.substr(0, pos);
}

std::string extension(const std::string& file)
{
  std::size_t position = file.find_last_of(".");
  return (std::string::npos == position || position == 0) ?  "" :  file.substr(position + 1);
}

std::string filename(const std::string& file)
{
  std::size_t position = file.find_last_of(".");
  std::size_t pos      = file.find_last_of("\\/");
  return (std::string::npos == pos) ?  file.substr(0, position) :  file.substr(pos + 1, position - pos
- 1);
}
```

5.41 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)

5.41.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Formatters.cc.

5.41.2 Function Documentation

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

Definition at line 71 of file Formatters.cc.

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

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

Definition at line 73 of file Formatters.cc.

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

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

Definition at line 75 of file Formatters.cc.

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

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

Definition at line 69 of file Formatters.cc.

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

5.41.2.5 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.

```
{
  std::size_t iend = end;
  if(iend == -1) iend = b.size();
  std::string ret;
  for(std::size_t k = begin; k < iend; k++)
  {
    ret += to_bin(b[k]);
    ret += " - ";
  }
  return ret;
}
```

5.41.2.6 to_dec() [1/5]

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

Definition at line 29 of file Formatters.cc.

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

5.41.2.7 to_dec() [2/5]

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

Definition at line 31 of file Formatters.cc.

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

5.41.2.8 to_dec() [3/5]

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

Definition at line 33 of file Formatters.cc.

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

5.41.2.9 to_dec() [4/5]

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

Definition at line 27 of file Formatters.cc.

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

5.41.2.10 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.

```
{
  std::size_t iend = end;
  if(iend == -1) iend = b.size();
  std::string ret;
  for(std::size_t k = begin; k < iend; k++)
  {
    ret += to_dec(b[k]);
    ret += " - ";
  }
  return ret;
}
```

5.41.2.11 to_hex() [1/5]

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

Definition at line 50 of file Formatters.cc.

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

5.41.2.12 to_hex() [2/5]

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

Definition at line 52 of file Formatters.cc.

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

5.41.2.13 to_hex() [3/5]

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

Definition at line 54 of file Formatters.cc.

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

5.41.2.14 to_hex() [4/5]

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

Definition at line 48 of file Formatters.cc.

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

5.41.2.15 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.

```
{
  std::size_t iend = end;
  if(iend == -1) iend = b.size();
  std::string ret;
  for(std::size_t k = begin; k < iend; k++)
  {
    ret += to_hex(b[k]);
    ret += " - ";
  }
  return ret;
}
```

5.41.2.16 to_oct() [1/5]

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

Definition at line 92 of file Formatters.cc.

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

5.41.2.17 to_oct() [2/5]

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

Definition at line 94 of file Formatters.cc.

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

5.41.2.18 to_oct() [3/5]

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

Definition at line 96 of file Formatters.cc.

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

5.41.2.19 to_oct() [4/5]

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

Definition at line 90 of file Formatters.cc.

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

5.41.2.20 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.

```
{
  std::size_t iend = end;
  if(iend == -1) iend = b.size();
  std::string ret;
  for(std::size_t k = begin; k < iend; k++)
  {
    ret += to_oct(b[k]);
    ret += " - ";
  }
  return ret;
}
```

5.42 Formatters.cc

Go to the documentation of this file.

```

#include "Formatters.h"

#include "Bits.h"
#include "Buffer.h"
#include "Words.h"

#include <fmt/format.h>

std::string to_dec(const Buffer& b, const std::size_t& begin, const std::size_t& end)
{
  std::size_t iend = end;
  if(iend == -1) iend = b.size();
  std::string ret;
  for(std::size_t k = begin; k < iend; k++)
  {
    ret += to_dec(b[k]);
    ret += " - ";
  }
  return ret;
}

std::string to_dec(const bit8_t& b) { return fmt::format("{:#d}", b); }

std::string to_dec(const bit16_t& b) { return fmt::format("{:#d}", b); }

std::string to_dec(const bit32_t& b) { return fmt::format("{:#d}", b); }

std::string to_dec(const bit64_t& b) { return fmt::format("{:#d}", b); }

std::string to_hex(const Buffer& b, const std::size_t& begin, const std::size_t& end)
{
  std::size_t iend = end;
  if(iend == -1) iend = b.size();
  std::string ret;
  for(std::size_t k = begin; k < iend; k++)
  {
    ret += to_hex(b[k]);
    ret += " - ";
  }
  return ret;
}

std::string to_hex(const bit8_t& b) { return fmt::format("{:#02x}", b); }

std::string to_hex(const bit16_t& b) { return fmt::format("{:#04x}", b); }

std::string to_hex(const bit32_t& b) { return fmt::format("{:#08x}", b); }

std::string to_hex(const bit64_t& b) { return fmt::format("{:#016x}", b); }

std::string to_bin(const Buffer& b, const std::size_t& begin, const std::size_t& end)
{
  std::size_t iend = end;
  if(iend == -1) iend = b.size();
  std::string ret;
  for(std::size_t k = begin; k < iend; k++)
  {
    ret += to_bin(b[k]);
    ret += " - ";
  }
  return ret;
}

std::string to_bin(const bit8_t& b) { return fmt::format("{:#08b}", b); }

std::string to_bin(const bit16_t& b) { return fmt::format("{:#016b}", b); }

std::string to_bin(const bit32_t& b) { return fmt::format("{:#032b}", b); }

std::string to_bin(const bit64_t& b) { return fmt::format("{:#064b}", b); }

std::string to_oct(const Buffer& b, const std::size_t& begin, const std::size_t& end)
{
  std::size_t iend = end;
  if(iend == -1) iend = b.size();
  std::string ret;
  for(std::size_t k = begin; k < iend; k++)
  {
    ret += to_oct(b[k]);
    ret += " - ";
  }
  return ret;
}

std::string to_oct(const bit8_t& b) { return fmt::format("{:#04o}", b); }

std::string to_oct(const bit16_t& b) { return fmt::format("{:#08o}", b); }

std::string to_oct(const bit32_t& b) { return fmt::format("{:#016o}", b); }

std::string to_oct(const bit64_t& b) { return fmt::format("{:#032o}", b); }
```

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

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

5.43.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file RawBufferNavigator.cc.

5.44 RawBufferNavigator.cc

Go to the documentation of this file.

```

#include "RawBufferNavigator.h"

#include "Words.h"

int RawBufferNavigator::m_Start = 92;

void RawBufferNavigator::StartAt(const int& start)
{
  if(start >= 0) m_Start = start;
}

RawBufferNavigator::RawBufferNavigator() {}

void RawBufferNavigator::setBuffer(const Buffer& b)
{
  m_Buffer           = b;
  m_StartPayload     = -1;
  m_StartPayloadDone = false;
}

std::uint8_t RawBufferNavigator::getDetectorID() { return m_Buffer[0]; }

bool RawBufferNavigator::findStartOfPayload()
{
  if(m_StartPayloadDone == true)
  {
    if(m_StartPayload == -1) return false;
    else
      return true;
  }
  else
  {
    m_StartPayloadDone = true;
    for(std::size_t i = m_Start; i < m_Buffer.size(); i++)
    {
      if(static_cast<std::uint8_t>(m_Buffer[i]) == static_cast<std::uint8_t>(Value::GLOBAL_HEADER) ||
static_cast<std::uint8_t>(m_Buffer[i]) == static_cast<std::uint8_t>(Value::GLOBAL_HEADER_TEMP))
      {
        m_StartPayload = i;
        return true;
      }
    }
    m_StartPayload = -1;
    return false;
  }
}

std::int32_t RawBufferNavigator::getStartOfPayload()
{
  findStartOfPayload();
  return m_StartPayload;
}

bool RawBufferNavigator::validPayload() { return m_StartPayload != -1; }

Buffer RawBufferNavigator::getPayload() { return Buffer(&(m_Buffer.begin()[m_StartPayload]),
m_Buffer.size() - m_StartPayload); }
```

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

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

5.45.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Version.cc.

5.46 Version.cc

Go to the documentation of this file.

```

#include "Version.h"

const static Version streamout_version;

std::uint8_t Version::getMajor() { return major; }

std::uint8_t Version::getMinor() { return minor; }

std::uint8_t Version::getPatch() { return patch; }

std::string Version::getPreRelease()
{
  switch(prerelease_type)
  {
    case semver::prerelease::alpha:  return "alpha";
    case semver::prerelease::beta:  return "beta";
    case semver::prerelease::rc:  return "rc";
    case semver::prerelease::none:  return "";
    default:  return "";
  }
}

std::uint8_t Version::getPreReleaseNumber() { return prerelease_number; }
```

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

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

Classes

- class textDump

5.47.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file textDump.h.

5.48 textDump.h

Go to the documentation of this file.

```

#pragma once

#include "Interface.h"
#include "PayloadParser.h"
#include "spdlog/sinks/stdout_color_sinks.h"

#include <memory>
#include <spdlog/logger.h>

class textDump :  public InterfaceWriter
{
public:
  textDump();
  void                               start();
  void                               processDIF(const PayloadParser&);
  void                               processFrame(const PayloadParser&, uint32_t frameIndex);
  void                               processPadInFrame(const PayloadParser&, uint32_t frameIndex,
uint32_t channelIndex);
  void                               processSlowControl(Buffer);
  void                               end();
  std::shared_ptr<spdlog::logger>& print() { return m_InternalLogger; }
  void                               setLevel(const spdlog::level::level_enum& level) {
m_InternalLogger->set_level(level); }

private:
  // This class is a dumb class to print on terminal so we need the logger + the standard one given by
the interface.
  std::shared_ptr<spdlog::logger> m_InternalLogger{nullptr};
};
```

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

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

5.49.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file textDump.cc.

5.50 textDump.cc

Go to the documentation of this file.

```
#include "textDump.h"

#include "PayloadParser.h"

textDump::textDump() :  InterfaceWriter("textDump", "1.0.0")
{
  m_InternalLogger = std::make_shared<spdlog::logger>("textDump",
std::make_shared<spdlog::sinks::stdout_color_sink_mt>());
  m_InternalLogger->set_level(spdlog::level::trace);
  addCompatibility("RawdataReader", ">=1.0.0");
  addCompatibility("DIFdataExample", ">=1.0.0");
}

void textDump::start() { print()->info("Will dump bunch of DIF data"); }

void textDump::processDIF(const PayloadParser& 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()); }

void textDump::processFrame(const PayloadParser& d, uint32_t frameIndex)
{
  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));
}

void textDump::processPadInFrame(const PayloadParser& d, uint32_t frameIndex, uint32_t channelIndex)
{
  if(d.getThresholdStatus(frameIndex, channelIndex) > 0) { print()->info("\t\tChannel {}, Threshold
{}", channelIndex, d.getThresholdStatus(frameIndex, channelIndex)); }
}

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

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

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

5.51.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file LCIOWriter.h.

5.52 LCIOWriter.h

Go to the documentation of this file.

```

#pragma once
```

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

5.53.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file LCIOWriter.cc.

5.54 LCIOWriter.cc

Go to the documentation of this file.

```

```

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

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

Classes

- class RawdataReader

5.55.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file RawdataReader.h.

5.56 RawdataReader.h

Go to the documentation of this file.

```
#pragma once

#include "Interface.h"

#include <array>
#include <cstdint>
#include <fstream>
#include <string>
#include <vector>

class Buffer;

class RawdataReader :  public InterfaceReader
{
public:
  explicit 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() { closeFile(); }
  static void setDefaultBufferSize(const std::size_t& size);

private:
  void                 uncompress();
  std::ifstream        m_FileStream;
  void                 setFileSize(const std::size_t& size);
  static std::size_t  m_BufferSize;
  std::size_t          m_FileSize{0};
  std::uint32_t        m_NumberOfDIF{0};
  std::uint32_t        m_EventNumber{0};
  std::vector<bit8_t> m_buf;
  std::string          m_Filename;
};
```

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

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

5.57.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file RawdataReader.cc.

5.58 RawdataReader.cc

Go to the documentation of this file.

```cpp

#include "RawdataReader.h"

#include "Exception.h"

#include <cstdint>
#include <cstring>
#include <stdexcept>
#include <zlib.h>

std::size_t RawdataReader::m_BufferSize = 0x100000;

void RawdataReader::setDefaultBufferSize(const std::size_t& size) { m_BufferSize = size; }

RawdataReader::RawdataReader(const char* fileName) :  InterfaceReader("RawdataReader", "1.0.0")
{
  m_buf.reserve(m_BufferSize);
  m_Filename = fileName;
}

void RawdataReader::start() { openFile(m_Filename); }

void RawdataReader::end() { closeFile(); }

void RawdataReader::uncompress()
{
  static const std::size_t size_buffer{0x20000};
  std::size_t              shift{3 * sizeof(std::uint32_t) + sizeof(std::uint64_t)};
  static bit8_t            obuf[size_buffer];
  unsigned long            size_buffer_end{0x20000};  // NOLINT(runtime/int)
  std::int8_t              rc = ::uncompress(obuf, &size_buffer_end, &m_Buffer[shift], m_Buffer.size() - shift);
  switch(rc)
  {
    case Z_OK: break;
    case Z_MEM_ERROR: throw Exception(Z_MEM_ERROR, "Not enough memory"); break;
    case Z_BUF_ERROR: throw Exception(Z_BUF_ERROR, "Not enough room in the output buffer"); break;
    case Z_DATA_ERROR: throw Exception(Z_DATA_ERROR, "The input data was corrupted or incomplete"); break;
    default:  throw Exception("The input data was corrupted or incomplete"); break;
  }
  memcpy(&m_Buffer[shift], obuf, size_buffer_end);
  m_Buffer.setSize(size_buffer_end + shift);
}

void RawdataReader::closeFile()
{
  try
  {
    if(m_FileStream.is_open()) m_FileStream.close();
  }
  catch(const std::ios_base::failure& e)
  {
    log()->error("Caught an ios_base::failure in closeFile :  {} {}", e.what(), e.code().value());
    throw;
  }
}

void RawdataReader::openFile(const std::string& fileName)
{
  try
  {
    m_FileStream.rdbuf()->pubsetbuf(0, 0);
    m_FileStream.exceptions(std::ifstream::failbit | std::ifstream::badbit);
    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
    m_FileStream.rdbuf()->pubsetbuf(0, 0);
    if(m_FileStream.is_open())
    {
      setFileSize(m_FileStream.tellg());
      m_FileStream.seekg(0, std::ios::beg);
    }
  }
  catch(const std::ios_base::failure& e)
  {
    log()->error("Caught an ios_base::failure in openFile :  {}", e.what());
    throw;
  }
}

bool RawdataReader::nextEvent()
{
  try
  {
    m_FileStream.read(reinterpret_cast<char*>(&m_EventNumber), sizeof(std::uint32_t));
    m_FileStream.read(reinterpret_cast<char*>(&m_NumberOfDIF), sizeof(std::uint32_t));
  }
  catch(const std::ios_base::failure& e)
  {
    return false;
  }
  return true;
}

bool RawdataReader::nextDIFbuffer()
{
  try
  {
    static int DIF_processed{0};
    if(DIF_processed >= m_NumberOfDIF)
    {
      DIF_processed = 0;
      return false;
    }
    else
    {
      DIF_processed++;
      std::uint32_t bsize{0};
      m_FileStream.read(reinterpret_cast<char*>(&bsize), sizeof(std::uint32_t));
      m_FileStream.read(reinterpret_cast<char*>(&m_buf[0]), bsize);
      m_Buffer = Buffer(m_buf);
    }
  }
  catch(const std::ios_base::failure& e)
  {
    log()->error("Caught an ios_base::failure in openFile : {}", e.what());
    return false;
  }
  return true;
}

const Buffer& RawdataReader::getBuffer()
{
  uncompress();
  return m_Buffer;
}

void RawdataReader::setFileSize(const std::size_t& size) { m_FileSize = size; }

float RawdataReader::getFileSize() { return m_FileSize; }
```

5.59 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

5.59.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file DIF.h.

5.59.2 Typedef Documentation

5.59.2.1 Hits_const_iterator

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

Definition at line 14 of file DIF.h.

5.60 DIF.h

Go to the documentation of this file.

```
#pragma once

#include "Hit.h"

#include <TObject.h>
#include <cstdint>
#include <map>
#include <vector>

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

class DIF :  public TObject
{
public:
  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;

private:
  std::uint8_t     m_ID{0};
  std::uint32_t    m_DTC{0};
  std::uint32_t    m_GTC{0};
  std::uint32_t    m_DIFBCID{0};
  std::uint64_t    m_AbsoluteBCID{0};
  std::vector<Hit> m_Hits;
  ClassDef(DIF, 1);
};
```

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

```
#include <vector>
```

5.61.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file DIFLinkDef.h.

5.62 DIFLinkDef.h

Go to the documentation of this file.

```
#pragma once
#include <vector>

#ifdef __CLING__
#pragma link C++ class DIF;
#pragma link C++ class Hit;
#pragma link C++ class std::vector < Hit>;
#endif
```

5.63 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

5.63.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Event.h.

5.63.2 Typedef Documentation

5.63.2.1 DIFs_const_iterator `using DIFs_const_iterator = std::map<std::uint8_t, DIF>::const_↩ iterator`

Definition at line 13 of file Event.h.

5.64 Event.h

Go to the documentation of this file.

```
#pragma once

#include "DIF.h"

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

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

class Event :  public TObject
{
public:
  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;

private:
  std::map<std::uint8_t, DIF> DIFs;
  ClassDef(Event, 1);
};
```

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

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

5.65.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file EventLinkDef.h.

5.66 EventLinkDef.h

Go to the documentation of this file.

```
#pragma once
#include <cstdint>
#include <map>
#include <vector>
#ifdef __CLING__
#pragma link C++ class DIF;
#pragma link C++ class std::vector < DIF>;
#pragma link C++ class Hit;
#pragma link C++ class std::vector < Hit>;
#pragma link C++ class Event;
#pragma link C++ class std::vector < Event>;
#pragma link C++ class std::map < std::uint8_t, DIF>;
#endif
```

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

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

Classes

- class Hit

5.67.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Hit.h.

5.68 Hit.h

Go to the documentation of this file.

```
#pragma once

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

class Hit :  public TObject
{
public:
  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;

private:
  std::uint8_t  m_DIF{0};
  std::uint8_t  m_ASIC{0};
  std::uint8_t  m_Channel{0};
  std::uint8_t  m_Threshold{0};
  std::uint32_t m_DTC{0};
  std::uint32_t m_GTC{0};
  std::uint32_t m_DIFBCID{0};
  std::uint32_t m_FrameBCID{0};
  std::uint32_t m_Timestamp{0};
  std::uint64_t m_AbsoluteBCID{0};
  ClassDef(Hit, 1);
};
```

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

5.69.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file HitLinkDef.h.

5.70 HitLinkDef.h

Go to the documentation of this file.

```
#pragma once
#ifdef __CLING__
#pragma link C++ class Hit;
#endif
```

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

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

Classes

- class ROOTWriter

5.72 ROOTWriter.h

Go to the documentation of this file.

```

#pragma once

#include "Buffer.h"
#include "Event.h"
#include "Interface.h"
#include "PayloadParser.h"

#include <TFile.h>
#include <TTree.h>
#include <string>
#include <vector>

class ROOTWriter :  public InterfaceWriter
{
public:
  ROOTWriter();

  void setFilename(const std::string&);

  void start();
  void processDIF(const PayloadParser&);
  void processFrame(const PayloadParser&, const std::uint32_t& frameIndex);
  void processPadInFrame(const PayloadParser&, const std::uint32_t& frameIndex, const std::uint32_t&
channelIndex);
  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();

private:
  TFile*      m_File{nullptr};
  TTree*      m_Tree{nullptr};
  Event*      m_Event{nullptr};
  DIF*        m_DIF{nullptr};
  Hit*        m_Hit{nullptr};
  std::string m_Filename;
};
```

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

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

5.73.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file DIF.cc.

5.74 DIF.cc

Go to the documentation of this file.

```

#include "DIF.h"

#include <cstdint>

void DIF::addHit(const Hit& hit) { m_Hits.push_back(hit); }

void DIF::setID(const std::uint8_t& id) { m_ID = id; }

std::uint8_t DIF::getID()const { return m_ID; }

void DIF::setDTC(const std::uint32_t& dtc) { m_DTC = dtc; }

std::uint32_t DIF::getDTC()const { return m_DTC; }

void DIF::setGTC(const std::uint32_t& gtc) { m_GTC = gtc; }

std::uint32_t DIF::getGTC()const { return m_GTC; }

void DIF::setDIFBCID(const std::uint32_t& difbcid) { m_DIFBCID = difbcid; }

std::uint32_t DIF::getDIFBCID()const { return m_DIFBCID; }

void DIF::setAbsoluteBCID(const std::uint64_t& absolutebcid) { m_AbsoluteBCID = absolutebcid; }

std::uint64_t DIF::getAbsoluteBCID()const { return m_AbsoluteBCID; }

std::vector<Hit>::const_iterator DIF::cbegin()const { return m_Hits.cbegin(); }

std::vector<Hit>::const_iterator DIF::cend()const { return m_Hits.cend(); }

void DIF::clear() { m_Hits.clear(); }
```

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

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

5.75.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Event.cc.

5.76 Event.cc

Go to the documentation of this file.

```
#include "Event.h"

void Event::clear() { DIFs.clear(); }

void Event::addDIF(const DIF& dif) { DIFs[dif.getID()] = dif; }

std::map<std::uint8_t, DIF>::const_iterator Event::cbegin()const { return DIFs.cbegin(); }

std::map<std::uint8_t, DIF>::const_iterator Event::cend()const { return DIFs.cend(); }
```

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

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

5.77.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file Hit.cc.

5.78 Hit.cc

Go to the documentation of this file.

```
#include "Hit.h"
void Hit::clear()
{
  m_DIF           = 0;
  m_ASIC          = 0;
  m_Channel       = 0;
  m_Threshold     = 0;
  m_DTC           = 0;
  m_GTC           = 0;
  m_DIFBCID       = 0;
  m_FrameBCID     = 0;
  m_Timestamp     = 0;
  m_AbsoluteBCID  = 0;
}

void Hit::setDIF(const std::uint8_t& dif) { m_DIF = dif; }

void Hit::setASIC(const std::uint8_t& asic) { m_ASIC = asic; }

void Hit::setChannel(const std::uint8_t& channel) { m_Channel = channel; }

void Hit::setThreshold(const std::uint8_t& threshold) { m_Threshold = threshold; }

void Hit::setDTC(const std::uint32_t& dtc) { m_DTC = dtc; }

void Hit::setGTC(const std::uint32_t& gtc) { m_GTC = gtc; }

void Hit::setDIFBCID(const std::uint32_t& difbcid) { m_DIFBCID = difbcid; }

void Hit::setFrameBCID(const std::uint32_t& framebcid) { m_FrameBCID = framebcid; }

void Hit::setTimestamp(const std::uint32_t& timestamp) { m_Timestamp = timestamp; }

void Hit::setAbsoluteBCID(const std::uint64_t& absolutebcid) { m_AbsoluteBCID = absolutebcid; }

std::uint8_t Hit::getDIFid()const { return m_DIF; }

```

```
std::uint8_t Hit::getASICid()const { return m_ASIC; }

std::uint8_t Hit::getChannel()const { return m_Channel; }

std::uint8_t Hit::getThreshold()const { return m_Threshold; }

std::uint32_t Hit::getDTC()const { return m_DTC; }

std::uint32_t Hit::getGTC()const { return m_GTC; }

std::uint32_t Hit::getDIFBCID()const { return m_DIFBCID; }

std::uint32_t Hit::getFrameBCID()const { return m_FrameBCID; }

std::uint32_t Hit::getTimestamp()const { return m_Timestamp; }

std::uint64_t Hit::getAbsoluteBCID()const { return m_AbsoluteBCID; }
```

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

`#include "ROOTWriter.h"`

5.79.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file ROOTWriter.cc.

5.80 ROOTWriter.cc

Go to the documentation of this file.

```

#include "ROOTWriter.h"

void ROOTWriter::setFilename(const std::string& filename) { m_Filename = filename; }

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

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

void ROOTWriter::end()
{
  if(m_Tree) m_Tree->Write();
  if(m_File)
  {
    m_File->Write();
    m_File->Close();
  }
  if(m_File) delete m_File;
}

void ROOTWriter::processDIF(const PayloadParser& d)
{
  m_DIF->setID(d.getDIFid());
  m_DIF->setDTC(d.getDTC());
  m_DIF->setGTC(d.getGTC());
  m_DIF->setDIFBCID(d.getBCID());
  m_DIF->setAbsoluteBCID(d.getAbsoluteBCID());
}

```

```
void ROOTWriter::processFrame(const PayloadParser& d, const std::uint32_t& frameIndex)
{
  m_Hit->setDIF(d.getDIFid());
  m_Hit->setASIC(d.getASICid(frameIndex));
  m_Hit->setDTC(d.getDTC());
  m_Hit->setGTC(d.getGTC());
  m_Hit->setDIFBCID(d.getBCID());
  m_Hit->setAbsoluteBCID(d.getAbsoluteBCID());
  m_Hit->setFrameBCID(d.getFrameBCID(frameIndex));
  m_Hit->setTimestamp(d.getFrameTimeToTrigger(frameIndex));
}

void ROOTWriter::processPadInFrame(const PayloadParser& d, const std::uint32_t& frameIndex, const
std::uint32_t& channelIndex)
{
  m_Hit->setChannel(channelIndex);
  m_Hit->setThreshold(static_cast<std::uint8_t>(d.getThresholdStatus(frameIndex, channelIndex)));
}

void ROOTWriter::startEvent()
{
  m_Event = new Event();
  // m_Event->clear();
}

void ROOTWriter::endEvent()
{
  m_Tree->Fill();
  if(m_Event) delete m_Event;
}

void ROOTWriter::startDIF()
{
  m_DIF = new DIF();
  // m_DIF->clear();
}

void ROOTWriter::endDIF()
{
  m_Event->addDIF(*m_DIF);
  delete m_DIF;
}

void ROOTWriter::startFrame()
{
  m_Hit = new Hit();
  // m_Hit->clear();
}

void ROOTWriter::endFrame()
{
  m_DIF->addHit(*m_Hit);
  delete m_Hit;
}

void ROOTWriter::startPad() {}

void ROOTWriter::endPad() {}
```