

streamout

Generated by Doxygen 1.9.3

<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy	1
<b>2 Class Index</b>	<b>1</b>
2.1 Class List	1
<b>3 File Index</b>	<b>2</b>
3.1 File List	2
<b>4 Class Documentation</b>	<b>3</b>
4.1 Buffer Class Reference	3
4.1.1 Detailed Description	4
4.1.2 Constructor & Destructor Documentation	4
4.1.3 Member Function Documentation	5
4.2 BufferLooper< SOURCE, DESTINATION > Class Template Reference	6
4.2.1 Detailed Description	7
4.2.2 Constructor & Destructor Documentation	7
4.2.3 Member Function Documentation	7
4.3 BufferLooperCounter Struct Reference	10
4.3.1 Detailed Description	11
4.3.2 Member Function Documentation	11
4.3.3 Member Data Documentation	11
4.4 DIF Class Reference	13
4.4.1 Detailed Description	13
4.4.2 Member Function Documentation	13
4.5 DIFPtr Class Reference	15
4.5.1 Detailed Description	16
4.5.2 Constructor & Destructor Documentation	18
4.5.3 Member Function Documentation	18
4.6 DIFSlowControl Class Reference	25
4.6.1 Detailed Description	25
4.6.2 Constructor & Destructor Documentation	25
4.6.3 Member Function Documentation	26
4.7 Event Class Reference	28
4.7.1 Detailed Description	28
4.7.2 Member Function Documentation	28
4.8 Exception Class Reference	29
4.8.1 Detailed Description	29
4.8.2 Constructor & Destructor Documentation	29
4.8.3 Member Function Documentation	29
4.9 Hit Class Reference	30
4.9.1 Detailed Description	31
4.9.2 Member Function Documentation	31

---

4.10 Interface Class Reference	34
4.10.1 Detailed Description	35
4.10.2 Constructor & Destructor Documentation	35
4.10.3 Member Function Documentation	35
4.11 InterfaceReader Class Reference	37
4.11.1 Detailed Description	37
4.11.2 Constructor & Destructor Documentation	38
4.11.3 Member Data Documentation	38
4.12 InterfaceWriter Class Reference	38
4.12.1 Detailed Description	39
4.12.2 Constructor & Destructor Documentation	39
4.12.3 Member Function Documentation	39
4.13 RawBufferNavigator Class Reference	40
4.13.1 Detailed Description	40
4.13.2 Constructor & Destructor Documentation	40
4.13.3 Member Function Documentation	41
4.14 RawdataReader Class Reference	42
4.14.1 Detailed Description	43
4.14.2 Constructor & Destructor Documentation	43
4.14.3 Member Function Documentation	43
4.15 ROOTWriter Class Reference	45
4.15.1 Detailed Description	46
4.15.2 Constructor & Destructor Documentation	46
4.15.3 Member Function Documentation	46
4.16 textDump Class Reference	49
4.16.1 Detailed Description	50
4.16.2 Constructor & Destructor Documentation	50
4.16.3 Member Function Documentation	50
4.17 Timer Class Reference	52
4.17.1 Detailed Description	52
4.17.2 Member Function Documentation	52
4.18 Version Class Reference	52
4.18.1 Detailed Description	53
4.18.2 Constructor & Destructor Documentation	53
4.18.3 Member Function Documentation	53
<b>5 File Documentation</b>	<b>54</b>
5.1 libs/core/include/Bits.h File Reference	54
5.1.1 Detailed Description	55
5.1.2 Typedef Documentation	55
5.1.3 Function Documentation	56
5.2 Bits.h	56

---

5.3 <a href="#">libs/core/include/Buffer.h File Reference</a>	56
5.3.1 Detailed Description	56
5.4 <a href="#">Buffer.h</a>	57
5.5 <a href="#">libs/core/include/BufferLooper.h File Reference</a>	57
5.5.1 Detailed Description	58
5.6 <a href="#">BufferLooper.h</a>	58
5.7 <a href="#">libs/core/include/BufferLooperCounter.h File Reference</a>	61
5.7.1 Detailed Description	61
5.8 <a href="#">BufferLooperCounter.h</a>	61
5.9 <a href="#">libs/core/include/DetectorId.h File Reference</a>	61
5.9.1 Detailed Description	62
5.9.2 Enumeration Type Documentation	62
5.10 <a href="#">DetectorId.h</a>	62
5.11 <a href="#">libs/core/include/DIFPtr.h File Reference</a>	62
5.11.1 Detailed Description	63
5.12 <a href="#">DIFPtr.h</a>	63
5.13 <a href="#">libs/core/include/DIFSlowControl.h File Reference</a>	66
5.13.1 Detailed Description	67
5.13.2 Function Documentation	67
5.14 <a href="#">DIFSlowControl.h</a>	67
5.15 <a href="#">libs/core/include/Exception.h File Reference</a>	68
5.15.1 Detailed Description	68
5.16 <a href="#">Exception.h</a>	68
5.17 <a href="#">libs/core/include/FileSystem.h File Reference</a>	69
5.17.1 Detailed Description	69
5.17.2 Function Documentation	69
5.18 <a href="#">FileSystem.h</a>	70
5.19 <a href="#">libs/core/include/Formatters.h File Reference</a>	70
5.19.1 Detailed Description	70
5.19.2 Function Documentation	70
5.20 <a href="#">Formatters.h</a>	74
5.21 <a href="#">libs/core/include/Interface.h File Reference</a>	74
5.21.1 Detailed Description	75
5.21.2 Enumeration Type Documentation	75
5.22 <a href="#">Interface.h</a>	76
5.23 <a href="#">libs/core/include/RawBufferNavigator.h File Reference</a>	77
5.23.1 Detailed Description	77
5.24 <a href="#">RawBufferNavigator.h</a>	77
5.25 <a href="#">libs/core/include/Timer.h File Reference</a>	78
5.25.1 Detailed Description	78
5.26 <a href="#">Timer.h</a>	78
5.27 <a href="#">libs/core/include/Utilities.h File Reference</a>	78

5.27.1 Detailed Description . . . . .	78
5.27.2 Function Documentation . . . . .	79
5.28 Utilities.h . . . . .	79
5.29 libs/core/include/Version.h File Reference . . . . .	79
5.29.1 Detailed Description . . . . .	79
5.30 Version.h . . . . .	80
5.31 libs/core/include/Words.h File Reference . . . . .	80
5.31.1 Detailed Description . . . . .	80
5.31.2 Enumeration Type Documentation . . . . .	80
5.32 Words.h . . . . .	81
5.33 libs/core/src/Bits.cc File Reference . . . . .	82
5.33.1 Detailed Description . . . . .	82
5.33.2 Function Documentation . . . . .	82
5.34 Bits.cc . . . . .	83
5.35 libs/core/src/BufferLooperCounter.cc File Reference . . . . .	83
5.36 BufferLooperCounter.cc . . . . .	83
5.37 libs/core/src/DIFSlowControl.cc File Reference . . . . .	83
5.37.1 Detailed Description . . . . .	83
5.37.2 Function Documentation . . . . .	84
5.38 DIFSlowControl.cc . . . . .	84
5.39 libs/core/src/Filesystem.cc File Reference . . . . .	87
5.39.1 Detailed Description . . . . .	87
5.39.2 Function Documentation . . . . .	87
5.40 Filesystem.cc . . . . .	88
5.41 libs/core/src/Formatters.cc File Reference . . . . .	88
5.41.1 Detailed Description . . . . .	89
5.41.2 Function Documentation . . . . .	89
5.42 Formatters.cc . . . . .	93
5.43 libs/core/src/RawBufferNavigator.cc File Reference . . . . .	94
5.43.1 Detailed Description . . . . .	94
5.44 RawBufferNavigator.cc . . . . .	94
5.45 libs/core/src/Version.cc File Reference . . . . .	95
5.45.1 Detailed Description . . . . .	95
5.46 Version.cc . . . . .	95
5.47 libs/interface/Dump/include/textDump.h File Reference . . . . .	96
5.47.1 Detailed Description . . . . .	96
5.48 textDump.h . . . . .	96
5.49 libs/interface/Dump/src/textDump.cc File Reference . . . . .	97
5.49.1 Detailed Description . . . . .	97
5.50 textDump.cc . . . . .	97
5.51 libs/interface/LCIO/include/LCIOWriter.h File Reference . . . . .	97
5.51.1 Detailed Description . . . . .	97

5.52	<a href="#">LCIOWriter.h</a>	98
5.53	<a href="#">libs/interface/LCIO/src/LCIOWriter.cc File Reference</a>	98
5.53.1	<a href="#">Detailed Description</a>	98
5.54	<a href="#">LCIOWriter.cc</a>	98
5.55	<a href="#">libs/interface/RawDataReader/include/RawdataReader.h File Reference</a>	98
5.55.1	<a href="#">Detailed Description</a>	98
5.56	<a href="#">RawdataReader.h</a>	99
5.57	<a href="#">libs/interface/RawDataReader/src/RawdataReader.cc File Reference</a>	99
5.57.1	<a href="#">Detailed Description</a>	99
5.58	<a href="#">RawdataReader.cc</a>	100
5.59	<a href="#">libs/interface/ROOT/include/DIF.h File Reference</a>	101
5.59.1	<a href="#">Detailed Description</a>	101
5.59.2	<a href="#">Typedef Documentation</a>	102
5.60	<a href="#">DIF.h</a>	102
5.61	<a href="#">libs/interface/ROOT/include/DIFLinkDef.h File Reference</a>	102
5.61.1	<a href="#">Detailed Description</a>	102
5.62	<a href="#">DIFLinkDef.h</a>	103
5.63	<a href="#">libs/interface/ROOT/include/Event.h File Reference</a>	103
5.63.1	<a href="#">Detailed Description</a>	103
5.63.2	<a href="#">Typedef Documentation</a>	103
5.64	<a href="#">Event.h</a>	104
5.65	<a href="#">libs/interface/ROOT/include/EventLinkDef.h File Reference</a>	104
5.65.1	<a href="#">Detailed Description</a>	104
5.66	<a href="#">EventLinkDef.h</a>	104
5.67	<a href="#">libs/interface/ROOT/include/Hit.h File Reference</a>	104
5.67.1	<a href="#">Detailed Description</a>	105
5.68	<a href="#">Hit.h</a>	105
5.69	<a href="#">libs/interface/ROOT/include/HitLinkDef.h File Reference</a>	105
5.69.1	<a href="#">Detailed Description</a>	105
5.70	<a href="#">HitLinkDef.h</a>	106
5.71	<a href="#">libs/interface/ROOT/include/ROOTWriter.h File Reference</a>	106
5.72	<a href="#">ROOTWriter.h</a>	106
5.73	<a href="#">libs/interface/ROOT/src/DIF.cc File Reference</a>	107
5.73.1	<a href="#">Detailed Description</a>	107
5.74	<a href="#">DIF.cc</a>	107
5.75	<a href="#">libs/interface/ROOT/src/Event.cc File Reference</a>	107
5.75.1	<a href="#">Detailed Description</a>	107
5.76	<a href="#">Event.cc</a>	108
5.77	<a href="#">libs/interface/ROOT/src/Hit.cc File Reference</a>	108
5.77.1	<a href="#">Detailed Description</a>	108
5.78	<a href="#">Hit.cc</a>	108
5.79	<a href="#">libs/interface/ROOT/src/ROOTWriter.cc File Reference</a>	109

<a href="#">5.79.1 Detailed Description</a> . . . . .	109
<a href="#">5.80 ROOTWriter.cc</a> . . . . .	109

## 1 Hierarchical Index

### 1.1 Class Hierarchy

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

<b>Buffer</b>	<a href="#">3</a>
<b>BufferLooper</b> < SOURCE, DESTINATION >	<a href="#">6</a>
<b>BufferLooperCounter</b>	<a href="#">10</a>
<b>DIFPtr</b>	<a href="#">15</a>
<b>DIFSlowControl</b>	<a href="#">25</a>
<b>Exception</b>	<a href="#">29</a>
<b>Interface</b>	<a href="#">34</a>
<b>InterfaceReader</b>	<a href="#">37</a>
<b>RawdataReader</b>	<a href="#">42</a>
<b>InterfaceWriter</b>	<a href="#">38</a>
<b>ROOTWriter</b>	<a href="#">45</a>
<b>textDump</b>	<a href="#">49</a>
<b>RawBufferNavigator</b>	<a href="#">40</a>
<b>Timer</b>	<a href="#">52</a>
<b>TObject</b>	
<b>DIF</b>	<a href="#">13</a>
<b>Event</b>	<a href="#">28</a>
<b>Hit</b>	<a href="#">30</a>
semver::version	
<b>Version</b>	<a href="#">52</a>

## 2 Class Index

### 2.1 Class List

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

<a href="#">Buffer</a>	<a href="#">3</a>
------------------------	-------------------

<a href="#">BufferLooper&lt; SOURCE, DESTINATION &gt;</a>	6
<a href="#">BufferLooperCounter</a>	10
<a href="#">DIF</a>	13
<a href="#">DIFPtr</a>	
M3 MICROROC and HARDROC2 dataformat	15
<a href="#">DIFSlowControl</a>	25
<a href="#">Event</a>	28
<a href="#">Exception</a>	29
<a href="#">Hit</a>	30
<a href="#">Interface</a>	34
<a href="#">InterfaceReader</a>	37
<a href="#">InterfaceWriter</a>	38
<a href="#">RawBufferNavigator</a>	
Class to navigate in the raw data buffer parse the header and send the payload as <a href="#">Buffer</a>	40
<a href="#">RawdataReader</a>	42
<a href="#">ROOTWriter</a>	45
<a href="#">textDump</a>	49
<a href="#">Timer</a>	52
<a href="#">Version</a>	52

## 3 File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">libs/core/include/Bits.h</a>	54
<a href="#">libs/core/include/Buffer.h</a>	56
<a href="#">libs/core/include/BufferLooper.h</a>	57
<a href="#">libs/core/include/BufferLooperCounter.h</a>	61
<a href="#">libs/core/include/DetectorId.h</a>	61
<a href="#">libs/core/include/DIFPtr.h</a>	62
<a href="#">libs/core/include/DIFSlowControl.h</a>	66
<a href="#">libs/core/include/Exception.h</a>	68
<a href="#">libs/core/include/Filesystem.h</a>	69



libs/core/include/ <a href="#">Formatters.h</a>	70
libs/core/include/ <a href="#">Interface.h</a>	74
libs/core/include/ <a href="#">RawBufferNavigator.h</a>	77
libs/core/include/ <a href="#">Timer.h</a>	78
libs/core/include/ <a href="#">Utilities.h</a>	78
libs/core/include/ <a href="#">Version.h</a>	79
libs/core/include/ <a href="#">Words.h</a>	80
libs/core/src/ <a href="#">Bits.cc</a>	82
libs/core/src/ <a href="#">BufferLooperCounter.cc</a>	83
libs/core/src/ <a href="#">DIFSlowControl.cc</a>	83
libs/core/src/ <a href="#">Filesystem.cc</a>	87
libs/core/src/ <a href="#">Formatters.cc</a>	88
libs/core/src/ <a href="#">RawBufferNavigator.cc</a>	94
libs/core/src/ <a href="#">Version.cc</a>	95
libs/interface/Dump/include/ <a href="#">textDump.h</a>	96
libs/interface/Dump/src/ <a href="#">textDump.cc</a>	97
libs/interface/LCIO/include/ <a href="#">LCIOWriter.h</a>	97
libs/interface/LCIO/src/ <a href="#">LCIOWriter.cc</a>	98
libs/interface/RawDataReader/include/ <a href="#">RawdataReader.h</a>	98
libs/interface/RawDataReader/src/ <a href="#">RawdataReader.cc</a>	99
libs/interface/ROOT/include/ <a href="#">DIF.h</a>	101
libs/interface/ROOT/include/ <a href="#">DIFLinkDef.h</a>	102
libs/interface/ROOT/include/ <a href="#">Event.h</a>	103
libs/interface/ROOT/include/ <a href="#">EventLinkDef.h</a>	104
libs/interface/ROOT/include/ <a href="#">Hit.h</a>	104
libs/interface/ROOT/include/ <a href="#">HitLinkDef.h</a>	105
libs/interface/ROOT/include/ <a href="#">ROOTWriter.h</a>	106
libs/interface/ROOT/src/ <a href="#">DIF.cc</a>	107
libs/interface/ROOT/src/ <a href="#">Event.cc</a>	107
libs/interface/ROOT/src/ <a href="#">Hit.cc</a>	108
libs/interface/ROOT/src/ <a href="#">ROOTWriter.cc</a>	109

## 4 Class Documentation

### 4.1 Buffer Class Reference

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

#### Public Member Functions

- [Buffer](#) ()
- virtual [~Buffer](#) ()
- [Buffer](#) (const [bit8\\_t](#) b[], const std::size\_t &i)
- [Buffer](#) (const char b[], const std::size\_t &i)
- template<typename T >  
  [Buffer](#) (const std::vector< T > &rawdata)
- template<typename T, std::size\_t N>  
  [Buffer](#) (const std::array< T, N > &rawdata)
- std::size\_t [size](#) () const
- std::size\_t [capacity](#) () const
- void [set](#) (unsigned char \*b)
- [bit8\\_t](#) \* [begin](#) () const
- [bit8\\_t](#) \* [end](#) () const
- [bit8\\_t](#) & [operator\[\]](#) (const std::size\_t &pos)
- [bit8\\_t](#) & [operator\[\]](#) (const std::size\_t &pos) const
- void [setSize](#) (const std::size\_t &[size](#))

#### 4.1.1 Detailed Description

Definition at line 14 of file [Buffer.h](#).

#### 4.1.2 Constructor & Destructor Documentation

##### 4.1.2.1 [Buffer\(\)](#) [1/5] `Buffer::Buffer ( ) [inline]`

Definition at line 17 of file [Buffer.h](#).

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

##### 4.1.2.2 [~Buffer\(\)](#) `virtual Buffer::~~Buffer ( ) [inline], [virtual]`

Definition at line 18 of file [Buffer.h](#).

```
00018 {}
```

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

Definition at line 19 of file [Buffer.h](#).

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

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

Definition at line 20 of file [Buffer.h](#).

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

**4.1.2.5 Buffer()** [4/5] `template<typename T >`  
`Buffer::Buffer (`  
`const std::vector< T > & rawdata ) [inline]`

Definition at line 21 of file [Buffer.h](#).

```
00021 : m_Buffer(const_cast<bit8_t*>(reinterpret_cast<const bit8_t*>(rawdata.data()))),  
m_Size(rawdata.size() * sizeof(T)), m_Capacity(rawdata.capacity() * sizeof(T)) {}
```

**4.1.2.6 Buffer()** [5/5] `template<typename T , std::size_t N>`  
`Buffer::Buffer (`  
`const std::array< T, N > & rawdata ) [inline]`

Definition at line 22 of file [Buffer.h](#).

```
00022 : m_Buffer(const_cast<bit8_t*>(reinterpret_cast<const bit8_t*>(rawdata.data()))),  
m_Size(rawdata.size() * sizeof(T)), m_Capacity(rawdata.size() * sizeof(T)) {}
```

## 4.1.3 Member Function Documentation

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

Definition at line 28 of file [Buffer.h](#).

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

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

Definition at line 25 of file [Buffer.h](#).

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

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

Definition at line 29 of file [Buffer.h](#).

```
00029 { return m_Buffer + m_Size; }
```

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

Definition at line 30 of file [Buffer.h](#).

```
00030 { return m_Buffer[pos]; }
```

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

Definition at line 31 of file [Buffer.h](#).

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

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

Definition at line 27 of file [Buffer.h](#).

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

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

Definition at line 33 of file [Buffer.h](#).

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

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

Definition at line 24 of file [Buffer.h](#).

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

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

- [libs/core/include/Buffer.h](#)

## 4.2 `BufferLooper< SOURCE, DESTINATION >` Class Template Reference

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

## Public Member Functions

- [BufferLooper](#) (SOURCE &source, DESTINATION &dest, bool debug=false)
- void [addSink](#) (const spdlog::sink\_ptr &sink, const spdlog::level::level\_enum &level=spdlog::get\_level())
- void [loop](#) (const std::uint32\_t &m\_NbrEventsToProcess=0)
- void [printAllCounters](#) ()
- std::shared\_ptr< spdlog::logger > [log](#) ()
- void [setDetectorIDs](#) (const std::vector< [DetectorID](#) > &detectorIDs)

### 4.2.1 Detailed Description

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

Definition at line 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](#).

```
00031                                     : m_Source(source),
    m_Destination(dest), m_Debug(debug)
00032 {
00033     m_Logger = spdlog::create<spdlog::sinks::null_sink_mt>("streamout");
00034     if(!spdlog::get("streamout")) { spdlog::register_logger(m_Logger); }
00035     m_Source.setLogger(m_Logger);
00036     m_Destination.setLogger(m_Logger);
00037 }
```

### 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](#).

```
00040 {
00041     sink->set_level(level);
00042     m_Sinks.push_back(sink);
00043     m_Logger = std::make_shared<spdlog::logger>("streamout", begin(m_Sinks), end(m_Sinks));
00044     m_Source.setLogger(m_Logger);
00045     m_Destination.setLogger(m_Logger);
00046 }
```

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

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

Definition at line 222 of file [BufferLooper.h](#).

```
00222 { 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](#).

```
00049 {
00050     // clang-format off
00051     fmt::print(fg(fmt::color::medium_orchid) | fmt::emphasis::bold,
00052         "\n"
00053         " SSSSSSSSSSSSSSS tttt
00054         tttt\n"
00055         "SS::::::::::::S ttt::t
00056         ttt::t\n"
00057         "S::::SSSSS::::S t::::t
00058         t::::t\n"
00059         "S::::S SSSSSS t::::t
00060         t::::t\n"
00061         "S::::S tttttt::::tttttt rrrrr rrrrrrrrr eeeeeeeeeee aaaaaaaaaaaaa
00062         mmmmmmm mmmmmmm oooooooooo uuuuuu uuuuuutttttt::::tttttt\n"
00063         "S::::S t::::t::::t r::rrrr::rr ee::::::::ee a:::::::::a
00064         mm:::::m m::::mm oo::::::::::oo u:::u u:::ut::::::::::t\n"
00065         " S::::SSSS t::::t::::t r::::::::::r e:::::eeeeee:::eaaaaaaaaa::::a
00066         m::::mm:::::mo:::::ou:::u u:::ut::::::::::t\n"
00067         " SS::::SSSSStttttt::::ttttt rr::::rrrrr::re::::e e::::e a::::a
00068         m::::mm:::::mo:::::oooo:::ou:::u u:::uttttt::::ttttt\n"
00069         " SSS::::SS t::::t r::::r r::re:::::eeeeee:::e aaaaaa::::a
00070         m::::mmmm:::::mmmm::::mo:::o o::::ou:::u u:::u t::::t\n"
00071         " SSSSSS::::S t::::t r::::r rrrrrrrre:::::e aa:::::::::a m:::m
00072         m:::m m:::mo:::o o::::ou:::u u:::u t::::t\n"
00073         " S::::S t::::t r::::r e:::::eeeeeeeeee a::::aaaa::::a m:::m
00074         m:::m m:::mo:::o o::::ou:::u u:::u t::::t\n"
00075         " S::::S t::::t ttttttr::r e:::::e a::::a a::::a m:::m
00076         m:::m m:::mo:::o o::::ou:::u u:::u t::::t tttttt\n"
00077         "SSSSSSS S::::S t::::ttt::tr::r e:::::e a::::a a::::a m:::m
00078         m:::m m:::mo::::oooo:::ou:::::uu t::::ttt::t\n"
00079         "S::::SSSSS::::S tt::::::::::tr::r e:::::eeeeeeea::::aaaa::::a m:::m
00080         m:::m m:::mo:::::o u:::::uu tt::::::::::t\n"
00081         "S::::SSSS ttttttr::r ee:::::e a:::::aa::am:::m
00082         m:::m m:::m oo:::::oo uu:::::uu tt::::tt\n"
00083         " SSSSSSSSSSSSSS tttttttttt rrrrrrr eeeeeeeeeeeee aaaaaaaaa aaaammmmmm
00084         mmmmmmm mmmmmmm oooooooooo uuuuuuuu uuuu tttttttttt {} \n"
00085         "\n",
00086         fmt::format(fg(fmt::color::red) | fmt::emphasis::bold, "v{}", streamout_version.to_string());
00087         // clang-format on
00088         log()->info("*****");
00089         log()->info("Streamout Version : {}", streamout_version.to_string());
00090         log()->info("Using InterfaceReader {} version {}", m_Source.getName(),
00091             m_Source.getVersion().to_string());
00092         log()->info("Using InterfaceWriter {} version {}", m_Destination.getName(),
00093             m_Destination.getVersion().to_string());
00094         if(!m_Destination.checkCompatibility(m_Source.getName(), m_Source.getVersion().to_string()))
00095         {
```

```

00079         log()->critical("{} version {} is not compatible with {} version {} ! ", m_Source.getName(),
m_Source.getVersion().to_string(), m_Destination.getName(), m_Destination.getVersion().to_string());
00080         log()->info("Compatible Interfaces for {} are", m_Destination.getName());
00081         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); }
00082         std::exit(-1);
00083     }
00084     if(!m_DetectorIDs.empty())
00085     {
00086         std::string ids;
00087         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)) + ";";
00088         log()->info("Detector ID(s) other than {} will be ignored", ids);
00089     }
00090     log()->info("*****");
00091     RawBufferNavigator bufferNavigator;
00092     Timer timer;
00093     timer.start();
00094     m_Source.start();
00095     m_Destination.start();
00096     while(m_Source.nextEvent() && m_NbrEventsToProcess >= m_NbrEvents)
00097     {
00098         m_Source.startEvent();
00099         m_Destination.startEvent();
00100
00101         m_Logger->warn("====* Event {} *====", m_NbrEvents);
00102         while(m_Source.nextDIFbuffer())
00103         {
00104             const Buffer& buffer = m_Source.getBuffer();
00105
00106             bufferNavigator.setBuffer(buffer);
00107             if(std::find(m_DetectorIDs.begin(), m_DetectorIDs.end(),
static_cast<DetectorID>(bufferNavigator.getDetectorID())) == m_DetectorIDs.end())
00108             {
00109                 m_Logger->debug("Ignoring detector ID : {}", bufferNavigator.getDetectorID());
00110                 continue;
00111             }
00112
00113             std::int32_t idstart = bufferNavigator.getStartOfPayload();
00114             if(m_Debug && idstart == -1) m_Logger->info(to_hex(buffer));
00115             c.DIFStarter[idstart]++;
00116             if(!bufferNavigator.validPayload())
00117             {
00118                 m_Logger->error("!bufferNavigator.validBuffer()");
00119                 continue;
00120             }
00121
00122             m_Source.startDIF();
00123             m_Destination.startDIF();
00124             DIFPtr d;
00125             // This is really a big error so skip DIF entirely if exception occurs
00126             try
00127             {
00128                 d.setBuffer(bufferNavigator.getPayload());
00129             }
00130             catch(const Exception& e)
00131             {
00132                 m_Logger->error!("{}", e.what());
00133                 continue;
00134             }
00135             bit8_t* debug_variable_1 = buffer.end();
00136             bit8_t* debug_variable_2 = d.end();
00137             if(debug_variable_1 != debug_variable_2) m_Logger->error("DIF BUFFER END {} {}",
fmt::ptr(debug_variable_1), fmt::ptr(debug_variable_2));
00138             if(m_Debug) assert(debug_variable_1 == debug_variable_2);
00139
00140             c.DIFPtrValueAtReturnedPos[d.begin()][d.getGetFramePtrReturn()]++;
00141             if(m_Debug) assert(d.begin()[d.getGetFramePtrReturn()] == 0xa0);
00142             c.SizeAfterDIFPtr[d.getGetFramePtrReturn()]++;
00143             m_Destination.processDIF(d);
00144             for(std::size_t i = 0; i < d.getNumberOfFrames(); ++i)
00145             {
00146                 m_Source.startFrame();
00147                 m_Destination.startFrame();
00148                 m_Destination.processFrame(d, i);
00149                 for(std::size_t j = 0; j < DU::NUMBER_PAD; ++j)
00150                 {
00151                     if(d.getThresholdStatus(i, j) != 0)
00152                     {
00153                         m_Source.startPad();
00154                         m_Destination.startPad();
00155                         m_Destination.processPadInFrame(d, i, j);
00156                         m_Source.endPad();
00157                         m_Destination.endPad();
00158                     }
00159                 }
00160             }
00161         }
00162     }

```

```

00167         m_Source.endFrame();
00168         m_Destination.endFrame();
00170     }
00171     // 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
00172     try
00173     {
00174         d.setSCBuffer();
00175     }
00176     catch(const Exception& e)
00177     {
00178         m_Logger->error("{} ", e.what());
00179     }
00180     bool processSC = false;
00181     if(d.hasSlowControlData())
00182     {
00183         c.hasSlowControl++;
00184         processSC = true;
00185     }
00186     if(d.badSCData())
00187     {
00188         c.hasBadSlowControl++;
00189         processSC = false;
00190     }
00191     if(processSC) { m_Destination.processSlowControl(d.getSCBuffer()); }
00192
00193     Buffer eod = d.getEndOfAllData();
00194     c.SizeAfterAllData[eod.size()]++;
00195     bit8_t* debug_variable_3 = eod.end();
00196     if(debug_variable_1 != debug_variable_3) m_Logger->info("END DATA BUFFER END {} {} ",
    fmt::ptr(debug_variable_1), fmt::ptr(debug_variable_3));
00197     if(m_Debug) assert(debug_variable_1 == debug_variable_3);
00198     if(eod.size() != 0) m_Logger->info("End of Data remaining stuff : {} ", to_hex(eod));
00199
00200     int nonzeroCount = 0;
00201     for(bit8_t* it = eod.begin(); it != eod.end(); it++)
00202         if(static_cast<int>(*it) != 0) nonzeroCount++;
00203     c.NonZeroValusAtEndOfData[nonzeroCount]++;
00204     m_Source.endDIF();
00205     m_Destination.endDIF();
00206 } // end of DIF while loop
00207 m_Logger->warn("====* Event {} *====", m_NbrEvents);
00208 m_NbrEvents++;
00209 m_Source.endEvent();
00210 m_Destination.endEvent();
00211 } // end of event while loop
00212 m_Destination.end();
00213 m_Source.end();
00214 timer.stop();
00215 fmt::print(fg(fmt::color::green) | fmt::emphasis::bold, "=== elapsed time {}ms ({}ms/event)
    ===\n", timer.getElapsedTime() / 1000, timer.getElapsedTime() / (1000 * m_NbrEvents));
00220 }

```

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

Definition at line 221 of file [BufferLooper.h](#).

```
00221 { 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 224 of file [BufferLooper.h](#).

```
00224 { 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](#).

```
00011 {
00012     fmt::print(fg(fmt::color::crimson) | fmt::emphasis::bold, "BUFFER LOOP FINAL STATISTICS : \n");
00013     printCounter("Start of DIF header", DIFStarter);
00014     printCounter("Value after DIF data are processed", DIFPtrValueAtReturnedPos);
00015     printCounter("Size remaining in buffer after end of DIF data", SizeAfterDIFPtr);
00016     fmt::print(fg(fmt::color::crimson) | fmt::emphasis::bold, "Number of Slow Control found {} out of
which {} are bad\n", hasSlowControl, hasBadSlowControl);
00017     printCounter("Size remaining after all of data have been processed", SizeAfterAllData);
00018     printCounter("Number on non zero values in end of data buffer", NonZeroValusAtEndOfData);
00019 }
```

##### 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](#).

```
00022 {
00023     std::string out{"statistics for " + description + " : \n"};
00024     for(std::map<int, int>::const_iterator it = m.begin(); it != m.end(); it++)
00025     {
00026         if(it != m.begin()) out += ", ";
00027         out += " [" + std::to\_string(it->first) + "]" = " + std::to\_string(it->second);
00028     }
00029     out += "\n";
00030     fmt::print(fg(fmt::color::crimson) | fmt::emphasis::bold, out);
00031 }
```

### 4.3.3 Member Data Documentation

**4.3.3.1 DIFPtrValueAtReturnedPos** `std::map<int, int> BufferLooperCounter::DIFPtrValueAtReturnedPos`

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::NonZeroValusAtEndOfData`

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](#).

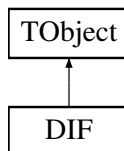
The documentation for this struct was generated from the following files:

- [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/DIFPtr.h>
```

## Public Member Functions

- `DIFPtr` ()=default
- void `setBuffer` (const `Buffer` &buffer)
- `bit8_t` \* `begin` () const
- `bit8_t` \* `end` () const
- std::uint32\_t `getSizeAfterDIFPtr` ()
- bool `hasSlowControlData` ()
- std::uint32\_t `getEndOfDIFData` ()
- bool `badSCData` ()
- void `setBuffer` (unsigned char \*, const std::uint32\_t &)
- `bit8_t` \* `getPtr` () const
- std::uint32\_t `getGetFramePtrReturn` () const
- std::vector< `bit8_t` \* > & `getFramesVector` ()
- std::vector< `bit8_t` \* > & `getLinesVector` ()
- std::uint32\_t `getID` () const
- std::uint32\_t `getDTC` () const
- std::uint32\_t `getGTC` () const
- std::uint64\_t `getAbsoluteBCID` () const
- std::uint32\_t `getBCID` () const
- std::uint32\_t `getLines` () const
- bool `hasLine` (const std::uint32\_t &) const
- std::uint32\_t `getTASU1` () const
- std::uint32\_t `getTASU2` () const
- std::uint32\_t `getTDIF` () const
- float `getTemperatureDIF` () const
- float `getTemperatureASU1` () const
- float `getTemperatureASU2` () const
- bool `hasTemperature` () const
- bool `hasAnalogReadout` () const
- std::uint32\_t `getNumberOfFrames` () const
- `bit8_t` \* `getFramePtr` (const std::uint32\_t &) const
- std::uint32\_t `getFrameAsicHeader` (const std::uint32\_t &) const
- std::uint32\_t `getFrameBCID` (const std::uint32\_t &) const
- std::uint32\_t `getFrameTimeToTrigger` (const std::uint32\_t &) const
- bool `getFrameLevel` (const std::uint32\_t &, const std::uint32\_t &, const std::uint32\_t &) const
- std::uint32\_t `getDIFid` () const
- std::uint32\_t `getASICid` (const std::uint32\_t &) const
- std::uint32\_t `getThresholdStatus` (const std::uint32\_t &, const std::uint32\_t &) const
- `Buffer` `getSCBuffer` ()
- `Buffer` `getEndOfAllData` ()
- std::uint32\_t `getDIF_CRC` ()
- void `setSCBuffer` ()

### 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)

daq software version (16 bits)

SDCC firmware version (16 bits)

DIF firmware version (16 bits)

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

timestamp (32bits) (milliseconde)



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) :

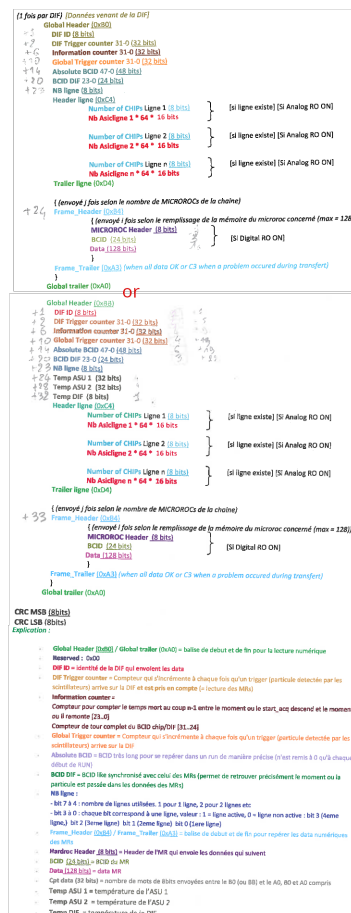


Figure 2 Data from the DIF analog or/and digital

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)

Definition at line 36 of file [DIFPtr.h](#).

## 4.5.2 Constructor & Destructor Documentation

### 4.5.2.1 DIFPtr() `DIFPtr::DIFPtr ( )` [default]

## 4.5.3 Member Function Documentation

### 4.5.3.1 badSCData() `bool DIFPtr::badSCData ( )` [inline]

Definition at line 55 of file [DIFPtr.h](#).

```
00056 {
00057     setSCBuffer();
00058     return m_BadSCdata;
00059 }
```



**4.5.3.2 begin()** `bit8_t * DIFPtr::begin ( ) const [inline]`

Definition at line 47 of file [DIFPtr.h](#).

```
00047 { return theDIF_; }
```

**4.5.3.3 end()** `bit8_t * DIFPtr::end ( ) const [inline]`

Definition at line 48 of file [DIFPtr.h](#).

```
00048 { return theDIF_ + theSize_; }
```

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

Definition at line 181 of file [DIFPtr.h](#).

```
00182 {
00183     std::uint64_t LBC = ((theDIF_[DU::ABCID_SHIFT] << 16) | (theDIF_[DU::ABCID_SHIFT + 1] << 8) |
    (theDIF_[DU::ABCID_SHIFT + 2])) * 16777216ULL /* to shift the value from the 24 first bits*/
00184     + ((theDIF_[DU::ABCID_SHIFT + 3] << 16) | (theDIF_[DU::ABCID_SHIFT + 4] << 8) |
    (theDIF_[DU::ABCID_SHIFT + 5]));
00185     return LBC;
00186 }
```

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

Definition at line 227 of file [DIFPtr.h](#).

```
00227 { return getFrameAsicHeader(i) & 0xFF; }
```

**4.5.3.6 getBCID()** `std::uint32_t DIFPtr::getBCID ( ) const [inline]`

Definition at line 188 of file [DIFPtr.h](#).

```
00188 { return (theDIF_[DU::BCID_SHIFT] << 16) + (theDIF_[DU::BCID_SHIFT + 1] << 8) + theDIF_[DU::BCID_SHIFT +
    2]; }
```

**4.5.3.7 getDIF\_CRC()** `std::uint32_t DIFPtr::getDIF_CRC ( ) [inline]`

Definition at line 103 of file [DIFPtr.h](#).

```
00104 {
00105     uint32_t i{getEndOfDIFData()};
00106     uint32_t ret{0};
00107     ret |= ((theDIF_[i - 2]) << 8);
00108     ret |= theDIF_[i - 1];
00109     return ret;
00110 }
```

#### 4.5.3.8 getDIFid() `uint32_t DIFPtr::getDIFid ( ) const [inline]`

Definition at line 225 of file [DIFPtr.h](#).

```
00225 { return getID() & 0xFF; }
```

#### 4.5.3.9 getDTC() `std::uint32_t DIFPtr::getDTC ( ) const [inline]`

Definition at line 177 of file [DIFPtr.h](#).

```
00177 { return (theDIF_[DU::DTC_SHIFT] << 24) + (theDIF_[DU::DTC_SHIFT + 1] << 16) + (theDIF_[DU::DTC_SHIFT + 2] << 8) + theDIF_[DU::DTC_SHIFT + 3]; }
```

#### 4.5.3.10 getEndOfAllData() `Buffer DIFPtr::getEndOfAllData ( ) [inline]`

Definition at line 96 of file [DIFPtr.h](#).

```
00097 {
00098     setSCBuffer();
00099     if(hasSlowControlData() && !m_BadSCdata) { return Buffer(&(m_SCbuffer.begin())[m_SCbuffer.size()]),
    getSizeAfterDIFPtr() - 3 - m_SCbuffer.size(); }
00100     else
00101         return Buffer(&(theDIF_[getEndOfDIFData()]), getSizeAfterDIFPtr() - 3); // remove the 2 bytes
    for CRC and the DIF trailer
00102 }
```

#### 4.5.3.11 getEndOfDIFData() `std::uint32_t DIFPtr::getEndOfDIFData ( ) [inline]`

Definition at line 53 of file [DIFPtr.h](#).

```
00053 { return getGetFramePtrReturn() + 3; }
```

#### 4.5.3.12 getFrameAsicHeader() `std::uint32_t DIFPtr::getFrameAsicHeader ( const std::uint32_t & i ) const [inline]`

Definition at line 214 of file [DIFPtr.h](#).

```
00214 { return getFrameAsicHeaderInternal(theFrames_[i]); }
```

#### 4.5.3.13 getFrameBCID() `std::uint32_t DIFPtr::getFrameBCID ( const std::uint32_t & i ) const [inline]`

Definition at line 216 of file [DIFPtr.h](#).

```
00216 { return GrayToBin((theFrames_[i][DU::FRAME_BCID_SHIFT] << 16) + (theFrames_[i][DU::FRAME_BCID_SHIFT + 1] << 8) + theFrames_[i][DU::FRAME_BCID_SHIFT + 2]); }
```

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

Definition at line 220 of file [DIFPtr.h](#).

```
00221 {
00222     return ((theFrames_[i][DU::FRAME_DATA_SHIFT + ((3 - ipad / 16) * 4 + (ipad % 16) / 4)] >> (7 -
    ((ipad % 16) % 4) * 2 + ilevel))) & 0x1);
00223 }
```

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

Definition at line 212 of file [DIFPtr.h](#).

```
00212 { return theFrames_[i]; }
```

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

Definition at line 171 of file [DIFPtr.h](#).

```
00171 { return theFrames_; }
```

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

Definition at line 218 of file [DIFPtr.h](#).

```
00218 { return getBCID() - getFrameBCID(i); }
```

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

Definition at line 169 of file [DIFPtr.h](#).

```
00169 { return theGetFramePtrReturn_; }
```

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

Definition at line 179 of file [DIFPtr.h](#).

```
00179 { return (theDIF_[DU::GTC_SHIFT] << 24) + (theDIF_[DU::GTC_SHIFT + 1] << 16) + (theDIF_[DU::GTC_SHIFT +
    2] << 8) + theDIF_[DU::GTC_SHIFT + 3]; }
```

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

Definition at line 175 of file [DIFPtr.h](#).

```
00175 { return theDIF_[DU::ID_SHIFT]; }
```

**4.5.3.21 getLines()** `std::uint32_t DIFPtr::getLines ( ) const [inline]`Definition at line 190 of file [DIFPtr.h](#).

```
00190 { return (theDIF_[DU::LINES_SHIFT] » 4) & 0x5; }
```

**4.5.3.22 getLinesVector()** `std::vector< bit8_t * > & DIFPtr::getLinesVector ( ) [inline]`Definition at line 173 of file [DIFPtr.h](#).

```
00173 { return theLines_; }
```

**4.5.3.23 getNumberOfFrames()** `std::uint32_t DIFPtr::getNumberOfFrames ( ) const [inline]`Definition at line 210 of file [DIFPtr.h](#).

```
00210 { return theFrames_.size(); }
```

**4.5.3.24 getPtr()** `bit8_t * DIFPtr::getPtr ( ) const [inline]`Definition at line 167 of file [DIFPtr.h](#).

```
00167 { return theDIF_; }
```

**4.5.3.25 getSCBuffer()** `Buffer DIFPtr::getSCBuffer ( ) [inline]`Definition at line 91 of file [DIFPtr.h](#).

```
00092 {  
00093     setSCBuffer();  
00094     return m_SCbuffer;  
00095 }
```

**4.5.3.26 getSizeAfterDIFPtr()** `std::uint32_t DIFPtr::getSizeAfterDIFPtr ( ) [inline]`Definition at line 50 of file [DIFPtr.h](#).

```
00050 { return theSize_ - getGetFramePtrReturn(); }
```

**4.5.3.27 getTASU1()** `std::uint32_t DIFPtr::getTASU1 ( ) const [inline]`Definition at line 194 of file [DIFPtr.h](#).

```
00194 { return (theDIF_[DU::TASU1_SHIFT] « 24) + (theDIF_[DU::TASU1_SHIFT + 1] « 16) +  
      (theDIF_[DU::TASU1_SHIFT + 2] « 8) + theDIF_[DU::TASU1_SHIFT + 3]; }
```

**4.5.3.28 getTASU2()** `std::uint32_t DIFPtr::getTASU2 ( ) const [inline]`Definition at line 196 of file [DIFPtr.h](#).

```
00196 { return (theDIF_[DU::TASU2_SHIFT] << 24) + (theDIF_[DU::TASU2_SHIFT + 1] << 16) +
      (theDIF_[DU::TASU2_SHIFT + 2] << 8) + theDIF_[DU::TASU2_SHIFT + 3]; }
```

**4.5.3.29 getTDIF()** `std::uint32_t DIFPtr::getTDIF ( ) const [inline]`Definition at line 198 of file [DIFPtr.h](#).

```
00198 { return theDIF_[DU::TDIF_SHIFT]; }
```

**4.5.3.30 getTemperatureASU1()** `float DIFPtr::getTemperatureASU1 ( ) const [inline]`Definition at line 202 of file [DIFPtr.h](#).

```
00202 { return (getTASU1() >> 3) * 0.0625; }
```

**4.5.3.31 getTemperatureASU2()** `float DIFPtr::getTemperatureASU2 ( ) const [inline]`Definition at line 204 of file [DIFPtr.h](#).

```
00204 { return (getTASU2() >> 3) * 0.0625; }
```

**4.5.3.32 getTemperatureDIF()** `float DIFPtr::getTemperatureDIF ( ) const [inline]`Definition at line 200 of file [DIFPtr.h](#).

```
00200 { return 0.508 * getTDIF() - 9.659; }
```

**4.5.3.33 getThresholdStatus()** `uint32_t DIFPtr::getThresholdStatus (
 const std::uint32_t & i,
 const std::uint32_t & ipad ) const [inline]`Definition at line 229 of file [DIFPtr.h](#).

```
00229 { return (((std::uint32_t) getFrameLevel(i, ipad, 1)) << 1) | ((std::uint32_t) getFrameLevel(i, ipad,
    0)); }
```

**4.5.3.34 hasAnalogReadout()** `bool DIFPtr::hasAnalogReadout ( ) const [inline]`Definition at line 208 of file [DIFPtr.h](#).

```
00208 { return getLines() != 0; }
```

**4.5.3.35 hasLine()** bool DIFPtr::hasLine (   
 const std::uint32\_t & line ) const [inline]

Definition at line 192 of file DIFPtr.h.

```
00192 { return ((theDIF_[DU::LINES_SHIFT] » line) & 0x1); }
```

**4.5.3.36 hasSlowControlData()** bool DIFPtr::hasSlowControlData ( ) [inline]

Definition at line 51 of file DIFPtr.h.

```
00051 { return theDIF_[getEndOfDIFData()] == 0xb1; }
```

**4.5.3.37 hasTemperature()** bool DIFPtr::hasTemperature ( ) const [inline]

Definition at line 206 of file DIFPtr.h.

```
00206 { return (theDIF_[0] == DU::START_OF_DIF_TEMP); }
```

**4.5.3.38 setBuffer()** [1/2] void DIFPtr::setBuffer (   
 const Buffer & buffer ) [inline]

Definition at line 41 of file DIFPtr.h.

```
00042 {
00043     setBuffer(buffer.begin(), buffer.size());
00044     m_BadSCdata = false;
00045 }
```

**4.5.3.39 setBuffer()** [2/2] void DIFPtr::setBuffer (   
 unsigned char \* ,   
 const std::uint32\_t & )

**4.5.3.40 setSCBuffer()** void DIFPtr::setSCBuffer ( ) [inline]

Definition at line 111 of file DIFPtr.h.

```
00112 {
00113     if(!hasSlowControlData()) return;
00114     if(m_SCbuffer.size() != 0) return; // deja fait
00115     if(m_BadSCdata) return;
00116     m_SCbuffer.set(&(theDIF_[getEndOfDIFData()]));
00117     // compute Slow Control size
00118     std::size_t maxsize{theSize_ - getEndOfDIFData() + 1}; // should I +1 here ?
00119     uint32_t k{1}; // SC Header
00120     uint32_t dif_ID{m_SCbuffer[1]};
00121     uint32_t chipSize{m_SCbuffer[3]};
00122     while((dif_ID != 0xal && m_SCbuffer[k] != 0xal && k < maxsize) || (dif_ID == 0xal && m_SCbuffer[k
+ 2] == chipSize && k < maxsize))
00123     {
00124         k += 2; // DIF ID + ASIC Header
00125         uint32_t scsize = m_SCbuffer[k];
00126         if(scsize != 74 && scsize != 109)
00127         {
00128             k = 0;
00129             m_BadSCdata = true;
00130             throw Exception(fmt::format("PROBLEM WITH SC SIZE {} ", scsize));

```

```

00131     }
00132     k++;          // skip size bit
00133     k += scsize;  // skip the data
00134 }
00135 if(m_Scbuffer[k] == 0xa1 && !m_BadSCdata) m_Scbuffer.setSize(k + 1); // add the trailer
00136 else
00137 {
00138     m_BadSCdata = true;
00139     throw Exception(fmt::format("PROBLEM SC TRAILER NOT FOUND "));
00140 }
00141 }

```

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

- `libs/core/include/DIFPtr.h`

## 4.6 DIFSlowControl Class Reference

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

### Public Member Functions

- [DIFSlowControl](#) (const std::uint8\_t &version, const std::uint8\_t &DIFid, unsigned char \*buf)  
*Constructor.*
- std::uint8\_t [getDIFid](#) ()  
*get DIF id*
- std::map< int, std::map< std::string, int > > [getChipsMap](#) ()  
*Get chips map.*
- std::map< std::string, int > [getChipSlowControl](#) (const int &asicid)  
*Get one chip map.*
- int [getChipSlowControl](#) (const std::int8\_t &asicid, const std::string &param)  
*Get one Chip value.*
- std::map< int, std::map< std::string, int > >::const\_iterator [cbegin](#) () const
- std::map< int, std::map< std::string, int > >::const\_iterator [cend](#) () const

### 4.6.1 Detailed Description

Definition at line 13 of file [DIFSlowControl.h](#).

### 4.6.2 Constructor & Destructor Documentation

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

Constructor.

## Parameters

<i>version</i>	Data format version
<i>DIFid</i>	<a href="#">DIF</a> id
<i>buf</i>	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] != 0xal) size_hardroc1 = 0;
00020
00021     int size_hardroc2 = m_NbrAsic * 109 + header_shift + 1;
00022     if(cbuf[size_hardroc2 - 1] != 0xal) size_hardroc2 = 0;
00023     if(size_hardroc1 != 0)
00024     {
00025         FillHR1(header_shift, cbuf);
00026         m_AsicType = 1;
00027     }
00028     else if(size_hardroc2 != 0)
00029         FillHR2(header_shift, cbuf);
00030     else
00031         return;
00032 }

```

## 4.6.3 Member Function Documentation

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

Definition at line 47 of file [DIFSlowControl.h](#).

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

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

Definition at line 49 of file [DIFSlowControl.h](#).

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

**4.6.3.3 getChipSlowControl()** `[1/2]` `std::map< std::string, int > DIFSlowControl::getChipSlowControl ( const int & asicid )` `[inline]`

Get one chip map.



## Parameters

<i>asid</i>	ASIC ID
-------------	---------

## Returns

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

Definition at line 38 of file [DIFSlowControl.cc](#).

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

**4.6.3.4 getChipSlowControl()** [2/2] `int DIFSlowControl::getChipSlowControl ( const std::int8_t & asid, const std::string & param ) [inline]`

Get one Chip value.

## Parameters

<i>asid</i>	ASic ID
<i>param</i>	Parameter name

Definition at line 40 of file [DIFSlowControl.cc](#).

```
00040 { return getChipSlowControl(asid)[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; }
```

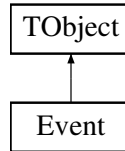
The documentation for this class was generated from the following files:

- [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(); }
```

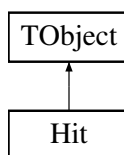
The documentation for this class was generated from the following file:

- [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; }
```

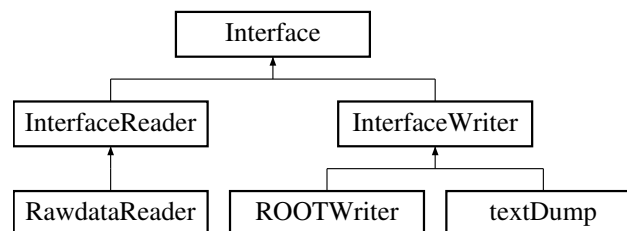
The documentation for this class was generated from the following files:

- [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 {}

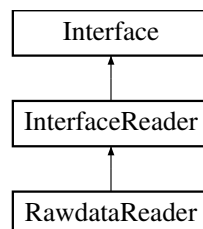
The documentation for this class was generated from the following file:

- [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](#).

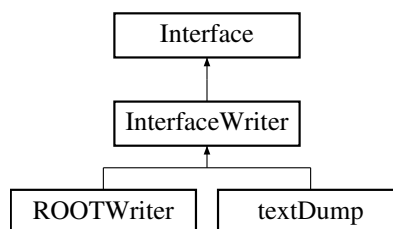
The documentation for this class was generated from the following file:

- [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](#).

```
00078 { 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](#).

```
00083 {  
00084     if(m_Compatible.find(name) != m_Compatible.end())  
00085     {  
00086         auto ran = semver::range::detail::range(m_Compatible[name]);  
00087         semver::version ver = semver::version(version);  
00088         if(ran.satisfies(ver, false)) return true;  
00089         else  
00090             return false;  
00091     }  
00092     else  
00093         return false;  
00094 }
```

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

Definition at line 80 of file [Interface.h](#).

```
00080 { return m_Compatible; }
```

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

- [libs/core/include/Interface.h](#)

## 4.13 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)

#### 4.13.1 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](#).

#### 4.13.2 Constructor & Destructor Documentation

**4.13.2.1 RawBufferNavigator()** `RawBufferNavigator::RawBufferNavigator ( )`

Definition at line 16 of file [RawBufferNavigator.cc](#).

```
00016 {}
```

**4.13.2.2** `~RawBufferNavigator()` `RawBufferNavigator::~RawBufferNavigator ( ) [default]`

### 4.13.3 Member Function Documentation

**4.13.3.1** `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(m_Buffer[i] == DU::START_OF_DIF || m_Buffer[i] == DU::START_OF_DIF_TEMP)
00041             {
00042                 m_StartPayload = i;
00043                 return true;
00044             }
00045         }
00046         m_StartPayload = -1;
00047         return false;
00048     }
00049 }
```

**4.13.3.2** `getDetectorID()` `std::uint8_t RawBufferNavigator::getDetectorID ( )`

Definition at line 25 of file [RawBufferNavigator.cc](#).

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

**4.13.3.3** `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); }
```

**4.13.3.4** `getStartOfPayload()` `std::int32_t RawBufferNavigator::getStartOfPayload ( )`

Definition at line 51 of file [RawBufferNavigator.cc](#).

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

**4.13.3.5 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 }
```

**4.13.3.6 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 }
```

**4.13.3.7 validPayload()** bool RawBufferNavigator::validPayload ( )

Definition at line 57 of file RawBufferNavigator.cc.

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

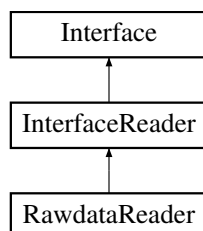
The documentation for this class was generated from the following files:

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

## 4.14 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

### 4.14.1 Detailed Description

Definition at line 17 of file [RawdataReader.h](#).

### 4.14.2 Constructor & Destructor Documentation

**4.14.2.1 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 }
```

**4.14.2.2 ~RawdataReader()** virtual RawdataReader::~RawdataReader ( ) [inline], [virtual]

Definition at line 29 of file [RawdataReader.h](#).

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

### 4.14.3 Member Function Documentation

**4.14.3.1 closeFile()** void RawdataReader::closeFile ( )

Definition at line 47 of file [RawdataReader.cc](#).

```
00048 {
00049     try
00050     {
00051         if(m_FileStream.is_open()) m_FileStream.close();
00052     }
00053     catch(const std::ios_base::failure& e)
00054     {
00055         log()->error("Caught an ios_base::failure in closeFile :  {} {}", e.what(), e.code().value());
00056         throw;
00057     }
00058 }
```

**4.14.3.2 end()** void RawdataReader::end ( )Definition at line 26 of file [RawdataReader.cc](#).

```
00026 { closeFile(); }
```

**4.14.3.3 getBuffer()** const Buffer & RawdataReader::getBuffer ( )Definition at line 122 of file [RawdataReader.cc](#).

```
00123 {
00124     uncompress();
00125     return m_Buffer;
00126 }
```

**4.14.3.4 getFileSize()** float RawdataReader::getFileSize ( )Definition at line 130 of file [RawdataReader.cc](#).

```
00130 { return m_FileSize; }
```

**4.14.3.5 nextDIFbuffer()** bool RawdataReader::nextDIFbuffer ( )Definition at line 95 of file [RawdataReader.cc](#).

```
00096 {
00097     try
00098     {
00099         static int DIF_processed{0};
00100         if(DIF_processed >= m_NumberOfDIF)
00101         {
00102             DIF_processed = 0;
00103             return false;
00104         }
00105         else
00106         {
00107             DIF_processed++;
00108             std::uint32_t bsize{0};
00109             m_FileStream.read(reinterpret_cast<char*>(&bsize), sizeof(std::uint32_t));
00110             m_FileStream.read(reinterpret_cast<char*>(&m_buf[0]), bsize);
00111             m_Buffer = Buffer(m_buf);
00112         }
00113     }
00114     catch(const std::ios_base::failure& e)
00115     {
00116         log()->error("Caught an ios_base::failure in openFile : {}", e.what());
00117         return false;
00118     }
00119     return true;
00120 }
```

**4.14.3.6 nextEvent()** bool RawdataReader::nextEvent ( )Definition at line 81 of file [RawdataReader.cc](#).

```
00082 {
00083     try
00084     {
00085         m_FileStream.read(reinterpret_cast<char*>(&m_EventNumber), sizeof(std::uint32_t));
00086         m_FileStream.read(reinterpret_cast<char*>(&m_NumberOfDIF), sizeof(std::uint32_t));
00087     }
00088     catch(const std::ios_base::failure& e)
00089     {
00090         return false;
00091     }
00092     return true;
00093 }
```

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

Definition at line 60 of file [RawdataReader.cc](#).

```
00061 {
00062     try
00063     {
00064         m_FileStream.rdbuf()->pubsetbuf(0, 0);
00065         m_FileStream.exceptions(std::ifstream::failbit | std::ifstream::badbit);
00066         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
00067         m_FileStream.rdbuf()->pubsetbuf(0, 0);
00068         if(m_FileStream.is_open())
00069         {
00070             setFileSize(m_FileStream.tellg());
00071             m_FileStream.seekg(0, std::ios::beg);
00072         }
00073     }
00074     catch(const std::ios_base::failure& e)
00075     {
00076         log()->error("Caught an ios_base::failure in openFile : {}", e.what());
00077         throw;
00078     }
00079 }
```

**4.14.3.8 setDefaultBufferSize()** void RawdataReader::setDefaultBufferSize (   
 const std::size\_t & size ) [static]

Definition at line 16 of file [RawdataReader.cc](#).

```
00016 { m_BufferSize = size; }
```

**4.14.3.9 start()** void RawdataReader::start ( )

Definition at line 24 of file [RawdataReader.cc](#).

```
00024 { openFile(m_Filename); }
```

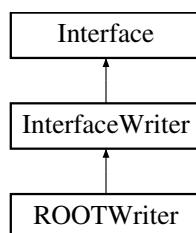
The documentation for this class was generated from the following files:

- [libs/interface/RawDataReader/include/RawdataReader.h](#)
- [libs/interface/RawDataReader/src/RawdataReader.cc](#)

## 4.15 ROOTWriter Class Reference

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

Inheritance diagram for ROOTWriter:



## Public Member Functions

- [ROOTWriter](#) ()
- void [setFilename](#) (const std::string &)
- void [start](#) ()
- void [processDIF](#) (const [DIFPtr](#) &)
- void [processFrame](#) (const [DIFPtr](#) &, const std::uint32\_t &frameIndex)
- void [processPadInFrame](#) (const [DIFPtr](#) &, const std::uint32\_t &frameIndex, const std::uint32\_t &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](#) ()

### 4.15.1 Detailed Description

Definition at line 18 of file [ROOTWriter.h](#).

### 4.15.2 Constructor & Destructor Documentation

#### 4.15.2.1 [ROOTWriter](#)() `ROOTWriter::ROOTWriter ( )`

Definition at line 10 of file [ROOTWriter.cc](#).

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

### 4.15.3 Member Function Documentation

#### 4.15.3.1 [end](#)() `void ROOTWriter::end ( )`

Definition at line 19 of file [ROOTWriter.cc](#).

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

**4.15.3.2 endDIF()** void ROOTWriter::endDIF ( ) [virtual]

Reimplemented from [Interface](#).

Definition at line 75 of file [ROOTWriter.cc](#).

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

**4.15.3.3 endEvent()** void ROOTWriter::endEvent ( ) [virtual]

Reimplemented from [Interface](#).

Definition at line 63 of file [ROOTWriter.cc](#).

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

**4.15.3.4 endFrame()** void ROOTWriter::endFrame ( ) [virtual]

Reimplemented from [Interface](#).

Definition at line 87 of file [ROOTWriter.cc](#).

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

**4.15.3.5 endPad()** void ROOTWriter::endPad ( ) [virtual]

Reimplemented from [Interface](#).

Definition at line 95 of file [ROOTWriter.cc](#).

```
00095 {}
```

**4.15.3.6 processDIF()** void ROOTWriter::processDIF (   
const [DIFPtr](#) & d )

Definition at line 30 of file [ROOTWriter.cc](#).

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

**4.15.3.7 processFrame()** void ROOTWriter::processFrame (

```
const DIFPtr & d,
const std::uint32_t & frameIndex )
```

Definition at line 39 of file [ROOTWriter.cc](#).

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

**4.15.3.8 processPadInFrame()** void ROOTWriter::processPadInFrame (

```
const DIFPtr & d,
const std::uint32_t & frameIndex,
const std::uint32_t & channelIndex )
```

Definition at line 51 of file [ROOTWriter.cc](#).

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

**4.15.3.9 processSlowControl()** void ROOTWriter::processSlowControl (

```
const Buffer & ) [inline]
```

Definition at line 29 of file [ROOTWriter.h](#).

```
00029 { ; }
```

**4.15.3.10 setFilename()** void ROOTWriter::setFilename (

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

Definition at line 8 of file [ROOTWriter.cc](#).

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

**4.15.3.11 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 }
```

**4.15.3.12 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 }
```

**4.15.3.13 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 }
```

**4.15.3.14 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 }
```

**4.15.3.15 startPad()** void ROOTWriter::startPad ( ) [virtual]

Reimplemented from [Interface](#).

Definition at line 93 of file [ROOTWriter.cc](#).

```
00093 {}
```

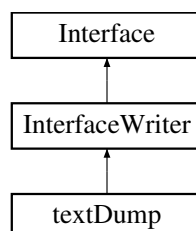
The documentation for this class was generated from the following files:

- [libs/interface/ROOT/include/ROOTWriter.h](#)
- [libs/interface/ROOT/src/ROOTWriter.cc](#)

## 4.16 textDump Class Reference

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

Inheritance diagram for textDump:



## Public Member Functions

- [textDump](#) ()
- void [start](#) ()
- void [processDIF](#) (const [DIFPtr](#) &)
- void [processFrame](#) (const [DIFPtr](#) &, uint32\_t frameIndex)
- void [processPadInFrame](#) (const [DIFPtr](#) &, uint32\_t frameIndex, uint32\_t channelIndex)
- void [processSlowControl](#) ([Buffer](#))
- void [end](#) ()
- std::shared\_ptr< spdlog::logger > & [print](#) ()
- void [setLevel](#) (const spdlog::level::level\_enum &level)

### 4.16.1 Detailed Description

Definition at line 14 of file [textDump.h](#).

### 4.16.2 Constructor & Destructor Documentation

#### 4.16.2.1 [textDump\(\)](#) `textDump::textDump ( )`

Definition at line 9 of file [textDump.cc](#).

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

### 4.16.3 Member Function Documentation

#### 4.16.3.1 [end\(\)](#) `void textDump::end ( )`

Definition at line 33 of file [textDump.cc](#).

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

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

Definition at line 24 of file [textDump.h](#).

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



**4.16.3.3 processDIF()** void textDump::processDIF (   
const [DIFPtr](#) & d )

Definition at line 19 of file [textDump.cc](#).

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

**4.16.3.4 processFrame()** void textDump::processFrame (   
const [DIFPtr](#) & d,   
uint32\_t frameIndex )

Definition at line 21 of file [textDump.cc](#).

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

**4.16.3.5 processPadInFrame()** void textDump::processPadInFrame (   
const [DIFPtr](#) & d,   
uint32\_t frameIndex,   
uint32\_t channelIndex )

Definition at line 26 of file [textDump.cc](#).

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

**4.16.3.6 processSlowControl()** void textDump::processSlowControl (   
[Buffer](#) )

Definition at line 31 of file [textDump.cc](#).

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

**4.16.3.7 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); }
```

**4.16.3.8 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](#)

## 4.17 Timer Class Reference

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

### Public Member Functions

- void [start](#) ()
- void [stop](#) ()
- float [getElapsedTime](#) ()

#### 4.17.1 Detailed Description

Definition at line 9 of file [Timer.h](#).

#### 4.17.2 Member Function Documentation

##### 4.17.2.1 [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(); }
```

##### 4.17.2.2 [start\(\)](#) void Timer::start ( ) [inline]

Definition at line 12 of file [Timer.h](#).

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

##### 4.17.2.3 [stop\(\)](#) void Timer::stop ( ) [inline]

Definition at line 13 of file [Timer.h](#).

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

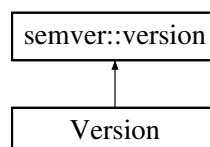
The documentation for this class was generated from the following file:

- [libs/core/include/Timer.h](#)

## 4.18 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 &prr=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](#) ()

### 4.18.1 Detailed Description

Definition at line 11 of file [Version.h](#).

### 4.18.2 Constructor & Destructor Documentation

**4.18.2.1 Version() [1/3]** `Version::Version (`  
    const std::uint8\_t & *mj*,  
    const std::uint8\_t & *mn*,  
    const std::uint8\_t & *pt*,  
    const semver::prerelease & *prr* = *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, prr, prn) {}
```

**4.18.2.2 Version() [2/3]** `Version::Version (`  
    const std::string\_view & *str* ) [inline], [explicit]

Definition at line 15 of file [Version.h](#).

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

**4.18.2.3 Version() [3/3]** `Version::Version ( ) [default]`

### 4.18.3 Member Function Documentation

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

Definition at line 9 of file [Version.cc](#).

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

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

Definition at line 11 of file [Version.cc](#).

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

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

Definition at line 13 of file [Version.cc](#).

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

#### 4.18.3.4 `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 }
```

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

## 5.3 libs/core/include/Buffer.h File Reference

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

### Classes

- class [Buffer](#)

### 5.3.1 Detailed Description

#### Copyright

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

#### See also

<https://github.com/apingault/Trivent4HEP>

Definition in file [Buffer.h](#).

## 5.4 Buffer.h

Go to the documentation of this file.

```

00001
00006 #pragma once
00007
00008 #include "Bits.h"
00009
00010 #include <array>
00011 #include <string>
00012 #include <vector>
00013
00014 class Buffer
00015 {
00016 public:
00017     Buffer() : m_Buffer(nullptr), m_Size(0), m_Capacity(0) {}
00018     virtual ~Buffer() {}
00019     Buffer(const bit8_t b[], const std::size_t& i) : m_Buffer(const_cast<bit8_t*>(&b[0])), m_Size(i),
m_Capacity(i) {}
00020     Buffer(const char b[], const std::size_t& i) : m_Buffer(const_cast<bit8_t*>(reinterpret_cast<const
bit8_t*>(&b[0])), m_Size(i * sizeof(char)), m_Capacity(i * sizeof(char)) {}
00021     template<typename T> Buffer(const std::vector<T>& rawdata) :
m_Buffer(const_cast<bit8_t*>(reinterpret_cast<const bit8_t*>(rawdata.data()))), m_Size(rawdata.size()
* sizeof(T)), m_Capacity(rawdata.capacity() * sizeof(T)) {}
00022     template<typename T, std::size_t N> Buffer(const std::array<T, N>& rawdata) :
m_Buffer(const_cast<bit8_t*>(reinterpret_cast<const bit8_t*>(rawdata.data()))), m_Size(rawdata.size()
* sizeof(T)), m_Capacity(rawdata.size() * sizeof(T)) {}
00023
00024     std::size_t size()const { return m_Size; }
00025     std::size_t capacity()const { return m_Capacity; }
00026
00027     void set(unsigned char* b) { m_Buffer = b; }
00028     bit8_t* begin()const { return m_Buffer; }
00029     bit8_t* end()const { return m_Buffer + m_Size; }
00030     bit8_t& operator[](const std::size_t& pos) { return m_Buffer[pos]; }
00031     bit8_t& operator[](const std::size_t& pos)const { return m_Buffer[pos]; }
00032
00033     void setSize(const std::size_t& size) { m_Size = size; }
00034
00035 private:
00036     bit8_t* m_Buffer{nullptr};
00037     std::size_t m_Size{0};
00038     std::size_t m_Capacity{0};
00039 };

```

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

```

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

```

### Classes

- class [BufferLooper< SOURCE, DESTINATION >](#)

### 5.5.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

Definition in file [BufferLooper.h](#).

## 5.6 BufferLooper.h

[Go to the documentation of this file.](#)

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

```



```

00061 "      SSS:::SS      t:::t      r:::r      r:::re:::eeee:::e aaaaaa:::a
m:::mm:::mm:::mo:::o      o:::ou:::u      u:::u      t:::t\n"
00062 "      SSSSS:::S      t:::t      r:::r      rrrrrre:::e aa:::a m:::m
m:::m      m:::mo:::o      o:::ou:::u      u:::u      t:::t\n"
00063 "      S:::S      t:::t      r:::r      e:::eeeeeeee a:::aaaa:::a m:::m
m:::m      m:::mo:::o      o:::ou:::u      u:::u      t:::t\n"
00064 "      S:::S      t:::t      tttttt:::r      e:::e      a:::a      a:::a m:::m
m:::m      m:::mo:::o      o:::ou:::uuu:::u      t:::t      ttttt\n"
00065 "SSSSSS      S:::S      t:::t      r:::r      e:::e      a:::a      a:::a m:::m
m:::m      m:::mo:::oooo:::ou:::uu      t:::t      tttt:::t\n"
00066 "S:::SSSSS:::S      tt:::tr:::r      e:::eeeeee:::aaaa:::a m:::m
m:::m      m:::mo:::o      o      u:::u      tt:::t\n"
00067 "S:::SS      tt:::tr:::r      ee:::e a:::aa:::am:::m
m:::m      m:::mo:::oo:::oo      uu:::uu      tt:::t\n"
00068 "SSSSSSSSSSSS      tttttttt rrrrrr      eeeeeeeeeee aaaaaaaa aaammmmmm
mmmmmm mmmmmmm oooooooooo uuuuuuuu uuuu      ttttttttt {} \n"
00069 "\n",
00070 fmt::format(fg(fmt::color::red) | fmt::emphasis::bold, "v{}", streamout_version.to_string());
00071 // clang-format on
00072 log()->info("*****");
00073 log()->info("Streamout Version : {}", streamout_version.to_string());
00074 log()->info("Using InterfaceReader {} version {}", m_Source.getName(),
m_Source.getVersion().to_string());
00075 log()->info("Using InterfaceWriter {} version {}", m_Destination.getName(),
m_Destination.getVersion().to_string());
00076
00077 if(!m_Destination.checkCompatibility(m_Source.getName(), m_Source.getVersion().to_string()))
00078 {
00079     log()->critical("{} version {} is not compatible with {} version {} ! ", m_Source.getName(),
m_Source.getVersion().to_string(), m_Destination.getName(), m_Destination.getVersion().to_string());
00080     log()->info("Compatible Interfaces for {} are", m_Destination.getName());
00081     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); }
00082     std::exit(-1);
00083 }
00084 if(!m_DetectorIDs.empty())
00085 {
00086     std::string ids;
00087     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)) + ";";
00088     log()->info("Detector ID(s) other than {} will be ignored", ids);
00089 }
00090 log()->info("*****");
00091 RawBufferNavigator bufferNavigator;
00092 Timer timer;
00093 timer.start();
00094 m_Source.start();
00095 m_Destination.start();
00096 while(m_Source.nextEvent() && m_NbrEventsToProcess >= m_NbrEvents)
00097 {
00098     m_Source.startEvent();
00099     m_Destination.startEvent();
00100
00101     m_Logger->warn("==== Event {} =====", m_NbrEvents);
00102     while(m_Source.nextDIFbuffer())
00103     {
00104         const Buffer& buffer = m_Source.getBuffer();
00105
00106         bufferNavigator.setBuffer(buffer);
00107         if(std::find(m_DetectorIDs.begin(), m_DetectorIDs.end(),
static_cast<DetectorID>(bufferNavigator.getDetectorID())) == m_DetectorIDs.end())
00108         {
00109             m_Logger->debug("Ignoring detector ID : {}", bufferNavigator.getDetectorID());
00110             continue;
00111         }
00112
00113         std::int32_t idstart = bufferNavigator.getStartOfPayload();
00114         if(m_Debug && idstart == -1) m_Logger->info(to_hex(buffer));
00115         c.DIFStarter[idstart]++;
00116         if(!bufferNavigator.validPayload())
00117         {
00118             m_Logger->error("!bufferNavigator.validBuffer()");
00119             continue;
00120         }
00121
00122         m_Source.startDIF();
00123         m_Destination.startDIF();
00124         DIFPtr d;
00125         // This is really a big error so skip DIF entirely if exception occurs
00126         try
00127         {
00128             d.setBuffer(bufferNavigator.getPayload());
00129         }
00130         catch(const Exception& e)
00131         {
00132             m_Logger->error("{} ", e.what());
00133         }
00134     }
00135 }

```

```

00137         continue;
00138     }
00139     bit8_t* debug_variable_1 = buffer.end();
00140     bit8_t* debug_variable_2 = d.end();
00141     if(debug_variable_1 != debug_variable_2) m_Logger->error("DIF BUFFER END {} {}",
fmt::ptr(debug_variable_1), fmt::ptr(debug_variable_2));
00142     if(m_Debug) assert(debug_variable_1 == debug_variable_2);
00143
00144     c.DIFPtrValueAtReturnedPos[d.begin()][d.getGetFramePtrReturn()]+=;
00145     if(m_Debug) assert(d.begin()[d.getGetFramePtrReturn()] == 0xa0);
00146     c.SizeAfterDIFPtr[d.getSizeAfterDIFPtr()]+=;
00147     m_Destination.processDIF(d);
00148     for(std::size_t i = 0; i < d.getNumberOfFrames(); ++i)
00149     {
00150         m_Source.startFrame();
00151         m_Destination.startFrame();
00152         m_Destination.processFrame(d, i);
00153         for(std::size_t j = 0; j < DU::NUMBER_PAD; ++j)
00154         {
00155             if(d.getThresholdStatus(i, j) != 0)
00156             {
00157                 m_Source.startPad();
00158                 m_Destination.startPad();
00159                 m_Destination.processPadInFrame(d, i, j);
00160                 m_Source.endPad();
00161                 m_Destination.endPad();
00162             }
00163         }
00164         m_Source.endFrame();
00165         m_Destination.endFrame();
00166     }
00167     // 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
00172     try
00173     {
00174         d.setSCBuffer();
00175     }
00176     catch(const Exception& e)
00177     {
00178         m_Logger->error("{} ", e.what());
00179     }
00180     bool processSC = false;
00181     if(d.hasSlowControlData())
00182     {
00183         c.hasSlowControl++;
00184         processSC = true;
00185     }
00186     if(d.badSCData())
00187     {
00188         c.hasBadSlowControl++;
00189         processSC = false;
00190     }
00191     if(processSC) { m_Destination.processSlowControl(d.getSCBuffer()); }
00192
00193     Buffer eod = d.getEndOfAllData();
00194     c.SizeAfterAllData[eod.size()]+=;
00195     bit8_t* debug_variable_3 = eod.end();
00196     if(debug_variable_1 != debug_variable_3) m_Logger->info("END DATA BUFFER END {} {}",
fmt::ptr(debug_variable_1), fmt::ptr(debug_variable_3));
00197     if(m_Debug) assert(debug_variable_1 == debug_variable_3);
00198     if(eod.size() != 0) m_Logger->info("End of Data remaining stuff : {} ", to_hex(eod));
00199
00200     int nonzeroCount = 0;
00201     for(bit8_t* it = eod.begin(); it != eod.end(); it++)
00202     {
00203         if(static_cast<int>(*it) != 0) nonzeroCount++;
00204     }
00205     c.NonZeroValusAtEndOfData[nonzeroCount]+=;
00206     m_Source.endDIF();
00207     m_Destination.endDIF();
00208     } // end of DIF while loop
00209     m_Logger->warn("***** Event {} *****", m_NbrEvents);
00210     m_NbrEvents++;
00211     m_Source.endEvent();
00212     m_Destination.endEvent();
00213     } // end of event while loop
00214     m_Destination.end();
00215     m_Source.end();
00216     timer.stop();
00217     fmt::print(fg(fmt::color::green) | fmt::emphasis::bold, "=== elapsed time {}ms ({}ms/event)
===\n", timer.getElapsedTime() / 1000, timer.getElapsedTime() / (1000 * m_NbrEvents));
00220 }
00221 void printAllCounters() { c.printAllCounters(); }
00222 std::shared_ptr<spdlog::logger> log() { return m_Logger; }
00223
00224 void setDetectorIDs(const std::vector<DetectorID>& detectorIDs) { m_DetectorIDs = detectorIDs; }
00225
00226 private:
00227     std::vector<DetectorID> m_DetectorIDs;

```

```

00228     std::shared_ptr<spdlog::logger> m_Logger{nullptr};
00229     std::vector<spdlog::sink_ptr>    m_Sinks;
00230     BufferLooperCounter              c;
00231     SOURCE&                         m_Source{nullptr};
00232     DESTINATION&                     m_Destination{nullptr};
00233     bool                             m_Debug{false};
00234     std::uint32_t                     m_NbrEvents{1};
00235 };

```

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

```

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

```

### Classes

- struct [BufferLooperCounter](#)

### 5.7.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

Definition in file [BufferLooperCounter.h](#).

## 5.8 BufferLooperCounter.h

[Go to the documentation of this file.](#)

```

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

```

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

```

#include <cstdint>

```

## Enumerations

- enum class [DetectorID](#) : std::uint16\_t { [HARDROC](#) = 100 , [HARDROC\\_NEW](#) = 150 , [RUNHEADER](#) = 255 }

### 5.9.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

Definition in file [DetectorId.h](#).

### 5.9.2 Enumeration Type Documentation

#### 5.9.2.1 [DetectorID](#) enum class [DetectorID](#) : std::uint16\_t [strong]

##### Enumerator

<a href="#">HARDROC</a>	
<a href="#">HARDROC_NEW</a>	
<a href="#">RUNHEADER</a>	

Definition at line 9 of file [DetectorId.h](#).

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

## 5.10 DetectorId.h

[Go to the documentation of this file.](#)

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

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

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

```
#include "Words.h"
#include <stdint>
#include <spdlog/spdlog.h>
#include <string>
#include <vector>
```

## Classes

- class [DIFPtr](#)  
*M3 MICROROC and HARDROC2 dataformat.*

### 5.11.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

Definition in file [DIFPtr.h](#).

## 5.12 DIFPtr.h

[Go to the documentation of this file.](#)

```
00001
00005 #pragma once
00006
00007 #include "Bits.h"
00008 #include "Buffer.h"
00009 #include "Exception.h"
00010 #include "Formatters.h"
00011 #include "Utilities.h"
00012 #include "Words.h"
00013
00014 #include <stdint>
00015 #include <spdlog/spdlog.h>
00016 #include <string>
00017 #include <vector>
00018
00036 class DIFPtr
00037 {
00038 public:
00039     DIFPtr() = default;
00040
00041     void setBuffer(const Buffer& buffer)
00042     {
00043         setBuffer(buffer.begin(), buffer.size());
00044         m_BadSCdata = false;
00045     }
00046
00047     bit8_t* begin()const { return theDIF_; }
00048     bit8_t* end()const { return theDIF_ + theSize_; }
00049
00050     std::uint32_t getSizeAfterDIFPtr() { return theSize_ - getGetFramePtrReturn(); }
00051     bool hasSlowControlData() { return theDIF_[getEndOfDIFData()] == 0xb1; }
00052
00053     std::uint32_t getEndOfDIFData() { return getGetFramePtrReturn() + 3; }
00054
00055     bool badSCData()
00056     {
00057         setSCBuffer();
00058         return m_BadSCdata;
00059     }
00060
00061     void setBuffer(unsigned char*, const std::uint32_t&);
00062     bit8_t* getPtr() const;
00063     std::uint32_t getGetFramePtrReturn() const;
00064     std::vector<bit8_t*> getFramesVector();
00065     std::vector<bit8_t*> getLinesVector();
```

```

00066     std::uint32_t         getID() const;
00067     std::uint32_t         getDTC() const;
00068     std::uint32_t         getGTC() const;
00069     std::uint64_t         getAbsoluteBCID() const;
00070     std::uint32_t         getBCID() const;
00071     std::uint32_t         getLines() const;
00072     bool                  hasLine(const std::uint32_t&) const;
00073     std::uint32_t         getTASU1() const;
00074     std::uint32_t         getTASU2() const;
00075     std::uint32_t         getTDIF() const;
00076     float                 getTemperatureDIF() const;
00077     float                 getTemperatureASU1() const;
00078     float                 getTemperatureASU2() const;
00079     bool                  hasTemperature() const;
00080     bool                  hasAnalogReadout() const;
00081     std::uint32_t         getNumberOfFrames() const;
00082     bit8_t*              getFramePtr(const std::uint32_t&) const;
00083     std::uint32_t         getFrameAsicHeader(const std::uint32_t&) const;
00084     std::uint32_t         getFrameBCID(const std::uint32_t&) const;
00085     std::uint32_t         getFrameTimeToTrigger(const std::uint32_t&) const;
00086     bool                  getFrameLevel(const std::uint32_t&, const std::uint32_t&, const
std::uint32_t&) const;
00087     // Addition by GG
00088     std::uint32_t         getDIFid() const;
00089     std::uint32_t         getASICid(const std::uint32_t&) const;
00090     std::uint32_t         getThresholdStatus(const std::uint32_t&, const std::uint32_t&) const;
00091     Buffer                getSCBuffer()
00092     {
00093         setSCBuffer();
00094         return m_SCbuffer;
00095     }
00096     Buffer getEndOfAllData()
00097     {
00098         setSCBuffer();
00099         if(hasSlowControlData() && !m_BadSCdata) { return Buffer(&(m_SCbuffer.begin())[m_SCbuffer.size()]),
getSizedAfterDIFPtr() - 3 - m_SCbuffer.size()); }
00100         else
00101             return Buffer(&(theDIF_[getEndOfDIFData()]), getSizedAfterDIFPtr() - 3); // remove the 2 bytes
for CRC and the DIF trailer
00102     }
00103     std::uint32_t getDIF_CRC()
00104     {
00105         uint32_t i{getEndOfDIFData()};
00106         uint32_t ret{0};
00107         ret |= ((theDIF_[i - 2]) << 8);
00108         ret |= theDIF_[i - 1];
00109         return ret;
00110     }
00111     void setSCBuffer()
00112     {
00113         if(!hasSlowControlData()) return;
00114         if(m_SCbuffer.size() != 0) return; // deja fait
00115         if(m_BadSCdata) return;
00116         m_SCbuffer.set(&(theDIF_[getEndOfDIFData()]));
00117         // compute Slow Control size
00118         std::size_t maxsize{theSize_ - getEndOfDIFData() + 1}; // should I +1 here ?
00119         uint32_t k{1}; // SC Header
00120         uint32_t dif_ID{m_SCbuffer[1]};
00121         uint32_t chipSize{m_SCbuffer[3]};
00122         while((dif_ID != 0xa1 && m_SCbuffer[k] != 0xa1 && k < maxsize) || (dif_ID == 0xa1 && m_SCbuffer[k
+ 2] == chipSize && k < maxsize))
00123         {
00124             k += 2; // DIF ID + ASIC Header
00125             uint32_t scsize = m_SCbuffer[k];
00126             if(scsize != 74 && scsize != 109)
00127             {
00128                 k
= 0;
00129                 m_BadSCdata = true;
00130                 throw Exception(fmt::format("PROBLEM WITH SC SIZE {} ", scsize));
00131             }
00132             k++; // skip size bit
00133             k += scsize; // skip the data
00134         }
00135         if(m_SCbuffer[k] == 0xa1 && !m_BadSCdata) m_SCbuffer.setSize(k + 1); // add the trailer
00136         else
00137         {
00138             m_BadSCdata = true;
00139             throw Exception(fmt::format("PROBLEM SC TRAILER NOT FOUND "));
00140         }
00141     }
00142 private:
00143     std::uint32_t         getAnalogPtr(const std::uint32_t& idx = 0);
00144     std::uint32_t         getFrameAsicHeaderInternal(const unsigned char* framePtr) const;
00145     std::uint32_t         getFramePtr();
00146     std::uint32_t         theSize_{0};
00147     theGetFramePtrReturn_{0};

```

```

00149     bit8_t*           theDIF_{nullptr};
00150     std::vector<bit8_t*> theFrames_;
00151     std::vector<bit8_t*> theLines_;
00152     bool              m_BadSCdata{false};
00153     Buffer            m_SCbuffer;
00154 };
00155
00156 inline std::uint32_t DIFPtr::getFrameAsicHeaderInternal(const bit8_t* framePtr) const { return
    (framePtr[DU::FRAME_ASIC_HEADER_SHIFT]); }
00157
00158 inline void DIFPtr::setBuffer(bit8_t* p, const std::uint32_t& max_size)
00159 {
00160     theFrames_.clear();
00161     theLines_.clear();
00162     theSize_      = max_size;
00163     theDIF_       = p;
00164     theGetFramePtrReturn_ = getFramePtr();
00165 }
00166
00167 inline bit8_t* DIFPtr::getPtr() const { return theDIF_; }
00168
00169 inline std::uint32_t DIFPtr::getGetFramePtrReturn() const { return theGetFramePtrReturn_; }
00170
00171 inline std::vector<bit8_t*>& DIFPtr::getFramesVector() { return theFrames_; }
00172
00173 inline std::vector<bit8_t*>& DIFPtr::getLinesVector() { return theLines_; }
00174
00175 inline std::uint32_t DIFPtr::getID() const { return theDIF_[DU::ID_SHIFT]; }
00176
00177 inline std::uint32_t DIFPtr::getDTC() const { return (theDIF_[DU::DTC_SHIFT] << 24) +
    (theDIF_[DU::DTC_SHIFT + 1] << 16) + (theDIF_[DU::DTC_SHIFT + 2] << 8) + theDIF_[DU::DTC_SHIFT + 3]; }
00178
00179 inline std::uint32_t DIFPtr::getGTC() const { return (theDIF_[DU::GTC_SHIFT] << 24) +
    (theDIF_[DU::GTC_SHIFT + 1] << 16) + (theDIF_[DU::GTC_SHIFT + 2] << 8) + theDIF_[DU::GTC_SHIFT + 3]; }
00180
00181 inline std::uint64_t DIFPtr::getAbsoluteBCID() const
00182 {
00183     std::uint64_t LBC = ((theDIF_[DU::ABCID_SHIFT] << 16) | (theDIF_[DU::ABCID_SHIFT + 1] << 8) |
    (theDIF_[DU::ABCID_SHIFT + 2])) * 16777216ULL /* to shift the value from the 24 first bits*/
00184     + ((theDIF_[DU::ABCID_SHIFT + 3] << 16) | (theDIF_[DU::ABCID_SHIFT + 4] << 8) |
    (theDIF_[DU::ABCID_SHIFT + 5]));
00185     return LBC;
00186 }
00187
00188 inline std::uint32_t DIFPtr::getBCID() const { return (theDIF_[DU::BCID_SHIFT] << 16) +
    (theDIF_[DU::BCID_SHIFT + 1] << 8) + theDIF_[DU::BCID_SHIFT + 2]; }
00189
00190 inline std::uint32_t DIFPtr::getLines() const { return (theDIF_[DU::LINES_SHIFT] >> 4) & 0x5; }
00191
00192 inline bool DIFPtr::hasLine(const std::uint32_t& line) const { return ((theDIF_[DU::LINES_SHIFT] >>
    line) & 0x1); }
00193
00194 inline std::uint32_t DIFPtr::getTASU1() const { return (theDIF_[DU::TASU1_SHIFT] << 24) +
    (theDIF_[DU::TASU1_SHIFT + 1] << 16) + (theDIF_[DU::TASU1_SHIFT + 2] << 8) + theDIF_[DU::TASU1_SHIFT +
    3]; }
00195
00196 inline std::uint32_t DIFPtr::getTASU2() const { return (theDIF_[DU::TASU2_SHIFT] << 24) +
    (theDIF_[DU::TASU2_SHIFT + 1] << 16) + (theDIF_[DU::TASU2_SHIFT + 2] << 8) + theDIF_[DU::TASU2_SHIFT +
    3]; }
00197
00198 inline std::uint32_t DIFPtr::getTDIF() const { return theDIF_[DU::TDIF_SHIFT]; }
00199
00200 inline float DIFPtr::getTemperatureDIF() const { return 0.508 * getTDIF() - 9.659; }
00201
00202 inline float DIFPtr::getTemperatureASU1() const { return (getTASU1() >> 3) * 0.0625; }
00203
00204 inline float DIFPtr::getTemperatureASU2() const { return (getTASU2() >> 3) * 0.0625; }
00205
00206 inline bool DIFPtr::hasTemperature() const { return (theDIF_[0] == DU::START_OF_DIF_TEMP); }
00207
00208 inline bool DIFPtr::hasAnalogReadout() const { return getLines() != 0; }
00209
00210 inline std::uint32_t DIFPtr::getNumberOfFrames() const { return theFrames_.size(); }
00211
00212 inline bit8_t* DIFPtr::getFramePtr(const std::uint32_t& i) const { return theFrames_[i]; }
00213
00214 inline std::uint32_t DIFPtr::getFrameAsicHeader(const std::uint32_t& i) const { return
    getFrameAsicHeaderInternal(theFrames_[i]); }
00215
00216 inline std::uint32_t DIFPtr::getFrameBCID(const std::uint32_t& i) const { return
    GrayToBin((theFrames_[i][DU::FRAME_BCID_SHIFT] << 16) + (theFrames_[i][DU::FRAME_BCID_SHIFT + 1] << 8) +
    theFrames_[i][DU::FRAME_BCID_SHIFT + 2]); }
00217
00218 inline std::uint32_t DIFPtr::getFrameTimeToTrigger(const std::uint32_t& i) const { return getBCID() -
    getFrameBCID(i); }
00219
00220 inline bool DIFPtr::getFrameLevel(const std::uint32_t& i, const std::uint32_t& ipad, const

```

```

        std::uint32_t& ilevel) const
00221 {
00222     return ((theFrames_[i][DU::FRAME_DATA_SHIFT + ((3 - ipad / 16) * 4 + (ipad % 16) / 4)] » (7 -
        ((ipad % 16) % 4) * 2 + ilevel))) & 0x1;
00223 }
00224 // Addition by GG
00225 inline uint32_t DIFPtr::getDIFid() const { return getID() & 0xFF; }
00226
00227 inline uint32_t DIFPtr::getASICid(const std::uint32_t& i) const { return getFrameAsicHeader(i) & 0xFF;
    }
00228
00229 inline uint32_t DIFPtr::getThresholdStatus(const std::uint32_t& i, const std::uint32_t& ipad) const {
    return (((std::uint32_t) getFrameLevel(i, ipad, 1)) « 1) | ((std::uint32_t) getFrameLevel(i, ipad, 0));
    }
00230
00231 inline std::uint32_t DIFPtr::getFramePtr()
00232 {
00233     std::uint32_t fshift{0};
00234     if(DATA_FORMAT_VERSION >= 13)
00235     {
00236         fshift = DU::LINES_SHIFT + 1;
00237         if(hasTemperature()) fshift = DU::TDIF_SHIFT + 1; // jenlev 1
00238         if(hasAnalogReadout()) fshift = getAnalogPtr(fshift); // to be implemented
00239     }
00240     else
00241         fshift = DU::BCID_SHIFT + 3;
00242     if(theDIF_[fshift] != DU::START_OF_FRAME) { throw Exception(fmt::format("This is not a start of
    frame {} ", to_hex(theDIF_[fshift]))); }
00243     do {
00244         if(theDIF_[fshift] == DU::END_OF_DIF) return fshift;
00245         if(theDIF_[fshift] == DU::START_OF_FRAME) fshift++;
00246         if(theDIF_[fshift] == DU::END_OF_FRAME)
00247         {
00248             fshift++;
00249             continue;
00250         }
00251         std::uint32_t header = getFrameAsicHeaderInternal(&theDIF_[fshift]);
00252         if(header == DU::END_OF_FRAME) return (fshift + 2);
00253         if(header < 1 || header > 48) { throw Exception(fmt::format("{} Header problem {} ", header,
    fshift)); }
00254         theFrames_.push_back(&theDIF_[fshift]);
00255         fshift += DU::FRAME_SIZE;
00256         if(fshift > theSize_) { throw Exception(fmt::format("fshift {} exceed {} ", fshift, theSize_)); }
00257         if(theDIF_[fshift] == DU::END_OF_FRAME) fshift++;
00258     } while(true);
00259 }
00260
00261 inline std::uint32_t DIFPtr::getAnalogPtr(const std::uint32_t& idx)
00262 {
00263     std::uint32_t fshift{idx};
00264     if(theDIF_[fshift] != DU::START_OF_LINES) return fshift;
00265     fshift++;
00266     while(theDIF_[fshift] != DU::END_OF_LINES)
00267     {
00268         theLines_.push_back(&theDIF_[fshift]);
00269         std::uint32_t nchip{theDIF_[fshift]};
00270         fshift += 1 + nchip * 64 * 2;
00271     }
00272     return fshift++;
00273 }

```

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

```

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

```

### Classes

- class [DIFSlowControl](#)



## Functions

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

### 5.13.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

Definition in file `DIFSlowControl.h`.

### 5.13.2 Function Documentation

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

Definition at line 256 of file `DIFSlowControl.cc`.

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

## 5.14 DIFSlowControl.h

[Go to the documentation of this file.](#)

```
00001
00005 #pragma once
00006
00007 #include <bitset>
00008 #include <cstdint>
00009 #include <iosfwd>
00010 #include <map>
00011 #include <string>
00012
00013 class DIFSlowControl
00014 {
00015 public:
00017     DIFSlowControl(const std::uint8_t& version, const std::uint8_t& DIFid, unsigned char* buf);
00023
00025     std::uint8_t getDIFId();
00026
00028
00031     std::map<int, std::map<std::string, int> getChipsMap();
00032
00034
00038     std::map<std::string, int> getChipSlowControl(const int& asicid);
00039
00041
00045     int getChipSlowControl(const std::int8_t& asicid, const std::string& param);
00046
00047     std::map<int, std::map<std::string, int>::const_iterator cbegin()const { return m_MapSC.cbegin(); }
00048
00049     std::map<int, std::map<std::string, int>::const_iterator cend()const { return m_MapSC.cend(); }
00050
00051 private:
00053     DIFSlowControl() = delete;
```

```

00055 void FillHR1(const int& header_shift, unsigned char* cbuf);
00057 void FillHR2(const int& header_shift, unsigned char* cbuf);
00059 void FillAsicHR1(const std::bitset<72 * 8>& bs);
00061 void FillAsicHR2(const std::bitset<109 * 8>& bs);
00062
00063 unsigned int m_DIFId{0};
00064 unsigned int m_Version{0};
00065 unsigned int m_AsicType{0}; // asicType_
00066 unsigned int m_NbrAsic{0};
00067 std::map<int, std::map<std::string, int> m_MapSC;
00068 };
00069
00070 std::string to_string(const DIFSlowControl& c);

```

## 5.15 libs/core/include/Exception.h File Reference

```

#include <stdint>
#include <exception>
#include <string>

```

### Classes

- class [Exception](#)

### 5.15.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

Definition in file [Exception.h](#).

## 5.16 Exception.h

[Go to the documentation of this file.](#)

```

00001
00005 #pragma once
00006
00007 #include <stdint>
00008 #include <exception>
00009 #include <string>
00010
00011 class Exception
00012 {
00013 public:
00014     virtual const char* what() const noexcept { return m_What.c_str(); }
00015     explicit Exception(const std::string& message) : m_Message(message) { constructWhat(); }
00016     Exception(const std::int32_t& error, const std::string& message) : m_Error(error),
00017         m_Message(message) { constructWhat(); }
00017     std::int32_t error() { return m_Error; }
00018     std::string message() { return m_Message; }
00019
00020 private:
00021     void constructWhat()
00022     {
00023         if(m_Error == 0) m_What = m_Message;
00024         else
00025             m_What = std::string("Error ") + std::to_string(m_Error) + std::string(" : ") + m_Message;
00026     }
00027     std::string m_What;
00028     std::string m_Message;
00029     std::int32_t m_Error{0};
00030 };

```

## 5.17 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 &`)

### 5.17.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

Definition in file [Filesystem.h](#).

### 5.17.2 Function Documentation

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

Definition at line 13 of file [Filesystem.cc](#).

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

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

Definition at line 19 of file [Filesystem.cc](#).

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

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

Definition at line 7 of file [Filesystem.cc](#).

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

## 5.18 Filesystem.h

[Go to the documentation of this file.](#)

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

## 5.19 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 &)`

### 5.19.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

Definition in file [Formatters.h](#).

### 5.19.2 Function Documentation

**5.19.2.1 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); }
```

**5.19.2.2 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); }
```

**5.19.2.3 to\_bin()** [3/5] std::string to\_bin (  
const bit64\_t & b )

Definition at line 75 of file [Formatters.cc](#).

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

**5.19.2.4 to\_bin()** [4/5] std::string to\_bin (  
const bit8\_t & b )

Definition at line 69 of file [Formatters.cc](#).

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

**5.19.2.5 to\_bin()** [5/5] std::string to\_bin (  
const Buffer & b,  
const std::size\_t & begin = 0,  
const std::size\_t & end = -1 )

Definition at line 56 of file [Formatters.cc](#).

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

**5.19.2.6 to\_dec()** [1/5] std::string to\_dec (  
const bit16\_t & b )

Definition at line 29 of file [Formatters.cc](#).

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

**5.19.2.7 to\_dec()** [2/5] std::string to\_dec (  
const bit32\_t & b )

Definition at line 31 of file [Formatters.cc](#).

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

**5.19.2.8 to\_dec()** [3/5] std::string to\_dec (  
const bit64\_t & b )

Definition at line 33 of file [Formatters.cc](#).

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

**5.19.2.9 to\_dec()** [4/5] std::string to\_dec (  
const bit8\_t & b )

Definition at line 27 of file [Formatters.cc](#).

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

**5.19.2.10 to\_dec()** [5/5] std::string to\_dec (  
const Buffer & b,  
const std::size\_t & begin = 0,  
const std::size\_t & end = -1 )

Definition at line 14 of file [Formatters.cc](#).

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

**5.19.2.11 to\_hex()** [1/5] std::string to\_hex (  
const bit16\_t & b )

Definition at line 50 of file [Formatters.cc](#).

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

**5.19.2.12 to\_hex()** [2/5] std::string to\_hex (  
const bit32\_t & b )

Definition at line 52 of file [Formatters.cc](#).

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

**5.19.2.13 to\_hex()** [3/5] std::string to\_hex (  
const bit64\_t & b )

Definition at line 54 of file [Formatters.cc](#).

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

**5.19.2.14 to\_hex()** [4/5] std::string to\_hex (  
const bit8\_t & b )

Definition at line 48 of file [Formatters.cc](#).

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

**5.19.2.15 to\_hex()** [5/5] std::string to\_hex (  
const Buffer & b,  
const std::size\_t & begin = 0,  
const std::size\_t & end = -1 )

Definition at line 35 of file [Formatters.cc](#).

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

**5.19.2.16 to\_oct()** [1/5] std::string to\_oct (  
const bit16\_t & b )

Definition at line 92 of file [Formatters.cc](#).

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

**5.19.2.17 to\_oct()** [2/5] std::string to\_oct (  
const bit32\_t & b )

Definition at line 94 of file [Formatters.cc](#).

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

**5.19.2.18 to\_oct()** [3/5] std::string to\_oct (  
const bit64\_t & b )

Definition at line 96 of file [Formatters.cc](#).

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

**5.19.2.19 to\_oct()** [4/5] std::string to\_oct (  
const bit8\_t & b )

Definition at line 90 of file [Formatters.cc](#).

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

**5.19.2.20 to\_oct()** [5/5] std::string to\_oct (  
const Buffer & b,  
const std::size\_t & begin = 0,  
const std::size\_t & end = -1 )

Definition at line 77 of file [Formatters.cc](#).

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

## 5.20 Formatters.h

[Go to the documentation of this file.](#)

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

## 5.21 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();*

### 5.21.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

Definition in file [Interface.h](#).

### 5.21.2 Enumeration Type Documentation

#### 5.21.2.1 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](#).

```

00032 {
00033     Unknown = 0,
00034     Reader  = 1,
00035     Writer  = 2
00036 };

```

## 5.22 Interface.h

[Go to the documentation of this file.](#)

```

00001
00004 #pragma once
00005
00006 #include "AppVersion.h"
00007 #include "Buffer.h"
00008 #include "Version.h"
00009
00010 #include <map>
00011 #include <memory>
00012 #include <semver.hpp>
00013 #include <spdlog/logger.h>
00014 #include <string>
00015
00031 enum class InterfaceType
00032 {
00033     Unknown = 0,
00034     Reader  = 1,
00035     Writer  = 2
00036 };
00037
00038 class Interface
00039 {
00040 public:
00041     Interface(const std::string& name, const std::string& version, const InterfaceType& type) :
00042         m_Name(name), m_Version(version) {}
00043     virtual ~Interface() = default;
00044     virtual void startEvent() {}
00045     virtual void endEvent() {}
00046     virtual void startDIF() {}
00047     virtual void endDIF() {}
00048     virtual void startFrame() {}
00049     virtual void endFrame() {}
00050     virtual void startPad() {}
00051     virtual void endPad() {}
00052     std::shared_ptr<spdlog::logger> log() { return m_Logger; }
00053     void setLogger(const std::shared_ptr<spdlog::logger> & logger) { m_Logger
00054         = logger; }
00055     std::string getName() { return m_Name; }
00056     Version getVersion() { return m_Version; }
00057 private:
00058     std::shared_ptr<spdlog::logger> m_Logger{nullptr};
00059     std::string m_Name;
00060     Version m_Version;
00061     InterfaceType m_Type{InterfaceType::Unknown};
00062 };
00063 class InterfaceReader : public Interface
00064 {
00065 public:
00066     InterfaceReader(const std::string& name, const std::string& version) : Interface(name, version,
00067         InterfaceType::Reader) {}
00068     virtual ~InterfaceReader() = default;
00069 protected:
00070     Buffer m_Buffer;
00071 };
00072
00073 class InterfaceWriter : public Interface
00074 {
00075 public:
00076     InterfaceWriter(const std::string& name, const std::string& version) : Interface(name, version,
00077         InterfaceType::Writer) {}
00078     void addCompatibility(const std::string& name, const std::string& version) { m_Compatible[name] =
00079         version; }
00080     std::map<std::string, std::string> getCompatibility() { return m_Compatible; }
00081     bool checkCompatibility(const std::string& name, const std::string& version)
00082     {
00083         if (m_Compatible.find(name) != m_Compatible.end())
00084         {
00085

```

```

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

```

## 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.](#)

```

00001
00005 #pragma once
00006
00007 #include "Buffer.h"
00008
00013 class RawBufferNavigator
00014 {
00015 public:
00016     static void StartAt(const int& start);
00017     RawBufferNavigator();
00018     ~RawBufferNavigator() = default;
00019     void        setBuffer(const Buffer&);
00020     std::uint8_t getDetectorID();
00021     bool        findStartOfPayload();
00022     std::int32_t getStartOfPayload();
00023     bool        validPayload();
00024     Buffer       getPayload();
00025
00026 private:
00027     static int    m_Start;
00028     Buffer         m_Buffer;
00029     bool          m_StartPayloadDone{false};
00030     std::int32_t  m_StartPayload{-1}; // -1 Means not found !
00031 };

```

## 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.](#)

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

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

```
#include <stdint>
```

### 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](#).

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

## 5.28 Utilities.h

[Go to the documentation of this file.](#)

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

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

```
#include <stdint>
#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.](#)

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

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

```
#include <cstdint>
```

### Enumerations

- enum [DU](#) : std::uint8\_t {  
[START\\_OF\\_DIF](#) = 0xB0 , [START\\_OF\\_DIF\\_TEMP](#) = 0xBB , [END\\_OF\\_DIF](#) = 0xA0 , [START\\_OF\\_LINES](#) =  
0xC4 ,  
[END\\_OF\\_LINES](#) = 0xD4 , [START\\_OF\\_FRAME](#) = 0xB4 , [END\\_OF\\_FRAME](#) = 0xA3 , [ID\\_SHIFT](#) = 1 ,  
[DTC\\_SHIFT](#) = 2 , [GTC\\_SHIFT](#) = 10 , [ABCID\\_SHIFT](#) = 14 , [BCID\\_SHIFT](#) = 20 ,  
[LINES\\_SHIFT](#) = 23 , [TASU1\\_SHIFT](#) = 24 , [TASU2\\_SHIFT](#) = 28 , [TDIF\\_SHIFT](#) = 32 ,  
[FRAME\\_ASIC\\_HEADER\\_SHIFT](#) = 0 , [FRAME\\_BCID\\_SHIFT](#) = 1 , [FRAME\\_DATA\\_SHIFT](#) = 4 , [FRAME\\_SIZE](#)  
= 20 ,  
[NUMBER\\_PAD](#) = 64 }

### 5.31.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

Definition in file [Words.h](#).

### 5.31.2 Enumeration Type Documentation

#### 5.31.2.1 [DU](#) enum [DU](#) : std::uint8\_t

## Enumerator

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

Definition at line 9 of file [Words.h](#).

```

00010 {
00011     START_OF_DIF      = 0xB0,
00012     START_OF_DIF_TEMP = 0xBB,
00013     END_OF_DIF        = 0xA0,
00014     START_OF_LINES    = 0xC4,
00015     END_OF_LINES      = 0xD4,
00016
00017     START_OF_FRAME    = 0xB4,
00018     END_OF_FRAME      = 0xA3,
00019
00020     ID_SHIFT          = 1,
00021     DTC_SHIFT         = 2,
00022     GTC_SHIFT         = 10,
00023     ABCID_SHIFT       = 14,
00024     BCID_SHIFT        = 20,
00025     LINES_SHIFT       = 23,
00026     TASU1_SHIFT       = 24,
00027     TASU2_SHIFT       = 28,
00028     TDIF_SHIFT        = 32,
00029
00030     FRAME_ASIC_HEADER_SHIFT = 0,
00031     FRAME_BCID_SHIFT      = 1,
00032     FRAME_DATA_SHIFT      = 4,
00033     FRAME_SIZE            = 20,
00034
00035     NUMBER_PAD = 64
00036 };

```

## 5.32 Words.h

[Go to the documentation of this file.](#)

```

00001
00005 #pragma once
00006
00007 #include <cstdint>
00008
00009 enum DU : std::uint8_t
00010 {
00011     START_OF_DIF      = 0xB0,

```

```

00012  START_OF_DIF_TEMP = 0xBB,
00013  END_OF_DIF         = 0xA0,
00014  START_OF_LINES     = 0xC4,
00015  END_OF_LINES       = 0xD4,
00016
00017  START_OF_FRAME     = 0xB4,
00018  END_OF_FRAME       = 0xA3,
00019
00020  ID_SHIFT           = 1,
00021  DTC_SHIFT          = 2,
00022  GTC_SHIFT          = 10,
00023  ABCID_SHIFT        = 14,
00024  BCID_SHIFT         = 20,
00025  LINES_SHIFT        = 23,
00026  TASU1_SHIFT        = 24,
00027  TASU2_SHIFT        = 28,
00028  TDIF_SHIFT         = 32,
00029
00030  FRAME_ASIC_HEADER_SHIFT = 0,
00031  FRAME_BCID_SHIFT       = 1,
00032  FRAME_DATA_SHIFT       = 4,
00033  FRAME_SIZE            = 20,
00034
00035  NUMBER_PAD = 64
00036 };

```

### 5.33 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.](#)

```
00001
00006 #include "Bits.h"
00007
00008 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.](#)

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

## 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.

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

## 5.38 DIFSlowControl.cc

[Go to the documentation of this file.](#)

```
00001
00005 #include "DIFSlowControl.h"
00006
00007 DIFSlowControl::DIFSlowControl(const std::uint8_t& version, const std::uint8_t& DIFId, unsigned char*
cbuf) : m_Version(version), m_DIFId(DIFId), m_AsicType(2)
00008 {
00009     if(cbuf[0] != 0xb1) return;
00010     int header_shift{6};
00011     if(m_Version < 8) m_NbrAsic = cbuf[5];
00012     else
00013     {
00014         m_DIFId = cbuf[1];
00015         m_NbrAsic = cbuf[2];
00016         header_shift = 3;
00017     }
00018     int size_hardroc1 = m_NbrAsic * 72 + header_shift + 1;
00019     if(cbuf[size_hardroc1 - 1] != 0xal) size_hardroc1 = 0;
00020
00021     int size_hardroc2 = m_NbrAsic * 109 + header_shift + 1;
00022     if(cbuf[size_hardroc2 - 1] != 0xal) size_hardroc2 = 0;
00023     if(size_hardroc1 != 0)
00024     {
00025         FillHR1(header_shift, cbuf);
00026         m_AsicType = 1;
00027     }
00028     else if(size_hardroc2 != 0)
00029         FillHR2(header_shift, cbuf);
00030     else
00031         return;
00032 }
00033
00034 inline std::uint8_t DIFSlowControl::getDIFId() { return m_DIFId; }
00035
00036 inline std::map<int, std::map<std::string, int> DIFSlowControl::getChipsMap() { return m_MapSC; }
00037
00038 inline std::map<std::string, int> DIFSlowControl::getChipSlowControl(const int& asicid) { return
m_MapSC[asicid]; }
00039
00040 inline int DIFSlowControl::getChipSlowControl(const std::int8_t& asicid, const std::string& param) {
return getChipSlowControl(asicid)[param]; }
00041
00042 void DIFSlowControl::FillHR1(const int& header_shift, unsigned char* cbuf)
00043 {
00044     int nasic{cbuf[header_shift - 1]};
00045     int idx{header_shift};
00046     for(int k = 0; k < nasic; k++)
00047     {
00048         std::bitset<72 * 8> bs;
00049         // printf("%x %x \n", cbuf[idx+k*72+69], cbuf[idx+k*72+70]);
00050         for(int l = 71; l >= 0; l--)
00051         {
00052             // printf("%d %x : %d -->", l, cbuf[idx+k*72+l], (71-l)*8);
00053             for(int m = 0; m < 8; m++)
00054             {
```

```

00055         if(((1 < m) & cbuf[idx + k * 72 + 1]) != 0) bs.set((71 - 1) * 8 + m, 1);
00056     else
00057         bs.set((71 - 1) * 8 + m, 0);
00058     // printf("%d", (int) bs[(71-1)*8+m]);
00059 }
00060 // printf("\n");
00061 }
00062 FillAsicHR1(bs);
00063 }
00064 }
00065
00066 void DIFSlowControl::FillHR2(const int& header_shift, unsigned char* cbuf)
00067 {
00068     // int scsizer=cbuf[header_shift-1]*109+(header_shift-1)+2;
00069     int nasic{cbuf[header_shift - 1]};
00070     int idx{header_shift};
00071     // std::cout<<" DIFSlowControl::FillHR nasic "<nasic<<std::endl;
00072     for(int k = 0; k < nasic; k++)
00073     {
00074         std::bitset<109 * 8> bs;
00075         // printf("%x %x \n",cbuf[idx+k*109+69],cbuf[idx+k*109+70]);
00076         for(int l = 108; l >= 0; l--)
00077         {
00078             // printf("%d %x : %d -->",l,cbuf[idx+k*109+1], (71-1)*8);
00079             for(int m = 0; m < 8; m++)
00080             {
00081                 if(((1 < m) & cbuf[idx + k * 109 + 1]) != 0) bs.set((108 - 1) * 8 + m, 1);
00082             else
00083                 bs.set((108 - 1) * 8 + m, 0);
00084             // printf("%d", (int) bs[(71-1)*8+m]);
00085         }
00086         // printf("\n");
00087     }
00088     FillAsicHR2(bs);
00089 }
00090 }
00091
00092 void DIFSlowControl::FillAsicHR1(const std::bitset<72 * 8>& bs)
00093 {
00094     // Asic Id
00095     int asicid{0};
00096     for(int j = 0; j < 8; j++)
00097         if(bs[j + 9] != 0) asicid += (1 << (7 - j));
00098     std::map<std::string, int> mAsic;
00099     // Slow Control
00100     mAsic["SSC0"] = static_cast<int>(bs[575]);
00101     mAsic["SSC1"] = static_cast<int>(bs[574]);
00102     mAsic["SSC2"] = static_cast<int>(bs[573]);
00103     mAsic["Choix_caisson"] = static_cast<int>(bs[572]);
00104     mAsic["SW_50k"] = static_cast<int>(bs[571]);
00105     mAsic["SW_100k"] = static_cast<int>(bs[570]);
00106     mAsic["SW_100f"] = static_cast<int>(bs[569]);
00107     mAsic["SW_50f"] = static_cast<int>(bs[568]);
00108
00109     mAsic["Valid_DC"] = static_cast<int>(bs[567]);
00110     mAsic["ON_Discri"] = static_cast<int>(bs[566]);
00111     mAsic["ON_Fsb"] = static_cast<int>(bs[565]);
00112     mAsic["ON_Otaq"] = static_cast<int>(bs[564]);
00113     mAsic["ON_W"] = static_cast<int>(bs[563]);
00114     mAsic["ON_Ss"] = static_cast<int>(bs[562]);
00115     mAsic["ON_Buf"] = static_cast<int>(bs[561]);
00116     mAsic["ON_Paf"] = static_cast<int>(bs[560]);
00117     // Gain
00118     for(int i = 0; i < 64; i++)
00119     {
00120         int gain{0};
00121         for(int j = 0; j < 6; j++)
00122             if(bs[176 + i * 6 + j] != 0) gain += (1 << j);
00123         mAsic["Channel_" + std::to_string(i) + "_" + "Gain"] = gain;
00124         mAsic["Channel_" + std::to_string(i) + "_" + "cTest"] = bs[112 + i];
00125         mAsic["Channel_" + std::to_string(i) + "_" + "Valid_trig"] = static_cast<int>(bs[25 + i]);
00126     }
00127
00128     mAsic["ON_Otabg"] = static_cast<int>(bs[111]);
00129     mAsic["ON_Dac"] = static_cast<int>(bs[110]);
00130     mAsic["ON_Otadac"] = static_cast<int>(bs[109]);
00131     // DAC
00132     int dac1{0};
00133     for(int j = 0; j < 10; j++)
00134         if(bs[j + 99] != 0) dac1 += (1 << j);
00135     mAsic["DAC1"] = dac1;
00136     int dac0{0};
00137     for(int j = 0; j < 10; j++)
00138         if(bs[j + 89] != 0) dac0 += (1 << j);
00139     mAsic["DAC0"] = dac0;
00140     mAsic["EN_Raz_Ext"] = static_cast<int>(bs[23]);
00141     mAsic["EN_Raz_Int"] = static_cast<int>(bs[22]);

```

```

00142 mAsic["EN_Out_Raz_Int"] = static_cast<int>(bs[21]);
00143 mAsic["EN_Trig_Ext"] = static_cast<int>(bs[20]);
00144 mAsic["EN_Trig_Int"] = static_cast<int>(bs[19]);
00145 mAsic["EN_Out_Trig_Int"] = static_cast<int>(bs[18]);
00146 mAsic["Bypass_Chip"] = static_cast<int>(bs[17]);
00147 mAsic["HardrocHeader"] = static_cast<int>(asicid);
00148 mAsic["EN_Out_Discr"] = static_cast<int>(bs[8]);
00149 mAsic["EN_Transmit_On"] = static_cast<int>(bs[7]);
00150 mAsic["EN_Dout"] = static_cast<int>(bs[6]);
00151 mAsic["EN_RamFull"] = static_cast<int>(bs[5]);
00152 m_MapSC[asicid] = mAsic;
00153 }
00154
00155 void DIFSlowControl::FillAsicHR2(const std::bitset<109 * 8>& bs)
00156 {
00157     int asicid{0};
00158     for(int j = 0; j < 8; j++)
00159         if(bs[j + (108 - 7) * 8 + 2] != 0) asicid += (1 << (7 - j));
00160     std::map<std::string, int> mAsic;
00161     for(int i = 0; i < 64; i++)
00162     {
00163         int gain{0};
00164         int mask{0};
00165         mAsic["Channel_" + std::to_string(i) + "_" + "cTest"] = bs[i];
00166         for(int j = 0; j < 8; j++)
00167             if(bs[64 + i * 8 + j] != 0) gain += (1 << j);
00168         mAsic["Channel_" + std::to_string(i) + "_" + "Gain"] = gain;
00169         for(int j = 0; j < 3; j++)
00170             if(bs[8 * 77 + 2 + i * 3 + j] != 0) mask += (1 << j);
00171         mAsic["Channel_" + std::to_string(i) + "_" + "Mask"] = mask;
00172     }
00173     mAsic["PwrOnPA"] = static_cast<int>(bs[8 * 72]);
00174     mAsic["Cmdb3SS"] = static_cast<int>(bs[8 * 72 + 1]);
00175     mAsic["Cmdb2SS"] = static_cast<int>(bs[8 * 72 + 2]);
00176     mAsic["Cmdb1SS"] = static_cast<int>(bs[8 * 72 + 3]);
00177     mAsic["Cmdb0SS"] = static_cast<int>(bs[8 * 72 + 4]);
00178     mAsic["SwSsc0"] = static_cast<int>(bs[8 * 72 + 5]);
00179     mAsic["SwSsc1"] = static_cast<int>(bs[8 * 72 + 6]);
00180     mAsic["SwSsc2"] = static_cast<int>(bs[8 * 72 + 7]);
00181
00182     mAsic["PwrOnBuff"] = static_cast<int>(bs[8 * 73]);
00183     mAsic["PwrOnSS"] = static_cast<int>(bs[8 * 73 + 1]);
00184     mAsic["PwrOnW"] = static_cast<int>(bs[8 * 73 + 2]);
00185     mAsic["Cmdb3Fsb2"] = static_cast<int>(bs[8 * 73 + 3]);
00186     mAsic["Cmdb2Fsb2"] = static_cast<int>(bs[8 * 73 + 4]);
00187     mAsic["Cmdb1Fsb2"] = static_cast<int>(bs[8 * 73 + 5]);
00188     mAsic["Cmdb0Fsb2"] = static_cast<int>(bs[8 * 73 + 6]);
00189     mAsic["Sw50k2"] = static_cast<int>(bs[8 * 73 + 7]);
00190
00191     mAsic["Sw100k2"] = static_cast<int>(bs[8 * 74]);
00192     mAsic["Sw100f2"] = static_cast<int>(bs[8 * 74 + 1]);
00193     mAsic["Sw50f2"] = static_cast<int>(bs[8 * 74 + 2]);
00194     mAsic["Cmdb3Fsb1"] = static_cast<int>(bs[8 * 74 + 3]);
00195     mAsic["Cmdb2Fsb1"] = static_cast<int>(bs[8 * 74 + 4]);
00196     mAsic["Cmdb1Fsb1"] = static_cast<int>(bs[8 * 74 + 5]);
00197     mAsic["Cmdb0Fsb1"] = static_cast<int>(bs[8 * 74 + 6]);
00198     mAsic["Sw50k1"] = static_cast<int>(bs[8 * 74 + 7]);
00199
00200     mAsic["Sw100k1"] = static_cast<int>(bs[8 * 75]);
00201     mAsic["Sw100f1"] = static_cast<int>(bs[8 * 75 + 1]);
00202     mAsic["Sw50f1"] = static_cast<int>(bs[8 * 75 + 2]);
00203     mAsic["Sel0"] = static_cast<int>(bs[8 * 75 + 3]);
00204     mAsic["Sel11"] = static_cast<int>(bs[8 * 75 + 4]);
00205     mAsic["PwrOnFsb"] = static_cast<int>(bs[8 * 75 + 5]);
00206     mAsic["PwrOnFsb1"] = static_cast<int>(bs[8 * 75 + 6]);
00207     mAsic["PwrOnFsb2"] = static_cast<int>(bs[8 * 75 + 7]);
00208
00209     mAsic["Sw50k0"] = static_cast<int>(bs[8 * 76]);
00210     mAsic["Sw100k0"] = static_cast<int>(bs[8 * 76 + 1]);
00211     mAsic["Sw100f0"] = static_cast<int>(bs[8 * 76 + 2]);
00212     mAsic["Sw50f0"] = static_cast<int>(bs[8 * 76 + 3]);
00213     mAsic["EnOtaQ"] = static_cast<int>(bs[8 * 76 + 4]);
00214     mAsic["OtaQ_PwrADC"] = static_cast<int>(bs[8 * 76 + 5]);
00215     mAsic["Discr_PwrA"] = static_cast<int>(bs[8 * 76 + 6]);
00216     mAsic["Discr2"] = static_cast<int>(bs[8 * 76 + 7]);
00217
00218     mAsic["Discr1"] = static_cast<int>(bs[8 * 77]);
00219     mAsic["RS_or_Discr"] = static_cast<int>(bs[8 * 77 + 1]);
00220
00221     mAsic["Header"] = asicid;
00222     for(int i = 0; i < 3; i++)
00223     {
00224         int B = 0;
00225         for(int j = 0; j < 10; j++)
00226             if(bs[8 * 102 + 2 + i * 10 + j] != 0) B += (1 << j);
00227         mAsic["B" + std::to_string(i)] = B;
00228     }

```

```

00229
00230 mAsic["Smallldac"] = static_cast<int>(bs[8 * 106]);
00231 mAsic["DacSw"] = static_cast<int>(bs[8 * 106 + 1]);
00232 mAsic["OtagBgSw"] = static_cast<int>(bs[8 * 106 + 2]);
00233 mAsic["Trig2b"] = static_cast<int>(bs[8 * 106 + 3]);
00234 mAsic["Trig1b"] = static_cast<int>(bs[8 * 106 + 4]);
00235 mAsic["Trig0b"] = static_cast<int>(bs[8 * 106 + 5]);
00236 mAsic["EnTrigOut"] = static_cast<int>(bs[8 * 106 + 6]);
00237 mAsic["DiscrOrOr"] = static_cast<int>(bs[8 * 106 + 7]);
00238
00239 mAsic["TrigExtVal"] = static_cast<int>(bs[8 * 107]);
00240 mAsic["RazChnIntVal"] = static_cast<int>(bs[8 * 107 + 1]);
00241 mAsic["RazChnExtVal"] = static_cast<int>(bs[8 * 107 + 2]);
00242 mAsic["ScOn"] = static_cast<int>(bs[8 * 107 + 3]);
00243 mAsic["CLKMux"] = static_cast<int>(bs[8 * 107 + 4]);
00244
00245 // EnOCDout1b EnOCDout2b EnOCTransmitOn1b EnOCTransmitOn2b EnOCChipsatb SelStartReadout
SelEndReadout
00246 mAsic["SelEndReadout"] = static_cast<int>(bs[8 * 108 + 1]);
00247 mAsic["SelStartReadout"] = static_cast<int>(bs[8 * 108 + 2]);
00248 mAsic["EnOCChipsatb"] = static_cast<int>(bs[8 * 108 + 3]);
00249 mAsic["EnOCTransmitOn2b"] = static_cast<int>(bs[8 * 108 + 4]);
00250 mAsic["EnOCTransmitOn1b"] = static_cast<int>(bs[8 * 108 + 5]);
00251 mAsic["EnOCDout2b"] = static_cast<int>(bs[8 * 108 + 6]);
00252 mAsic["EnOCDout1b"] = static_cast<int>(bs[8 * 108 + 7]);
00253 m_MapSC[asid] = mAsic;
00254 }
00255
00256 std::string to_string(const DIFSlowControl& c)
00257 {
00258     std::string ret;
00259     for(std::map<int, std::map<std::string, int>::const_iterator it = c.cbegin(); it != c.cend(); it++)
00260     {
00261         ret += "ASIC " + std::to_string(it->first) + " :\n";
00262         for(std::map<std::string, int>::const_iterator jt = (it->second).begin(); jt !=
(it->second).end(); jt++) ret += jt->first + " : " + std::to_string(jt->second) + "\n";
00263     }
00264     return ret;
00265 }

```

## 5.39 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](#).

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

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

Definition at line 19 of file [Filesystem.cc](#).

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

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

Definition at line 7 of file [Filesystem.cc](#).

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

## 5.40 Filesystem.cc

[Go to the documentation of this file.](#)

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

## 5.41 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`.

```
00071 { 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`.

```
00073 { 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](#).

```
00075 { 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](#).

```
00069 { 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](#).

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

**5.41.2.6 to\_dec()** [1/5] std::string to\_dec (  
const bit16\_t & b )

Definition at line 29 of file [Formatters.cc](#).

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

**5.41.2.7 to\_dec()** [2/5] std::string to\_dec (  
const bit32\_t & b )

Definition at line 31 of file [Formatters.cc](#).

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

**5.41.2.8 to\_dec()** [3/5] std::string to\_dec (  
const bit64\_t & b )

Definition at line 33 of file [Formatters.cc](#).

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



**5.41.2.9 to\_dec()** [4/5] std::string to\_dec (  
const bit8\_t & b )

Definition at line 27 of file [Formatters.cc](#).

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

**5.41.2.10 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](#).

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

**5.41.2.11 to\_hex()** [1/5] std::string to\_hex (  
const bit16\_t & b )

Definition at line 50 of file [Formatters.cc](#).

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

**5.41.2.12 to\_hex()** [2/5] std::string to\_hex (  
const bit32\_t & b )

Definition at line 52 of file [Formatters.cc](#).

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

**5.41.2.13 to\_hex()** [3/5] std::string to\_hex (  
const bit64\_t & b )

Definition at line 54 of file [Formatters.cc](#).

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

**5.41.2.14 to\_hex()** [4/5] std::string to\_hex (  
const bit8\_t & b )

Definition at line 48 of file [Formatters.cc](#).

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

**5.41.2.15 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](#).

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

**5.41.2.16 to\_oct()** [1/5] std::string to\_oct (  
    const bit16\_t & b )

Definition at line 92 of file [Formatters.cc](#).

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

**5.41.2.17 to\_oct()** [2/5] std::string to\_oct (  
    const bit32\_t & b )

Definition at line 94 of file [Formatters.cc](#).

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

**5.41.2.18 to\_oct()** [3/5] std::string to\_oct (  
    const bit64\_t & b )

Definition at line 96 of file [Formatters.cc](#).

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

**5.41.2.19 to\_oct()** [4/5] std::string to\_oct (  
    const bit8\_t & b )

Definition at line 90 of file [Formatters.cc](#).

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

**5.41.2.20 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](#).

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

## 5.42 Formatters.cc

[Go to the documentation of this file.](#)

```
00001
00006 #include "Formatters.h"
00007
00008 #include "Bits.h"
00009 #include "Buffer.h"
00010 #include "Words.h"
00011
00012 #include <fmt/format.h>
00013
00014 std::string to_dec(const Buffer& b, const std::size_t& begin, const std::size_t& end)
00015 {
00016     std::size_t iend = end;
00017     if(iend == -1) iend = b.size();
00018     std::string ret;
00019     for(std::size_t k = begin; k < iend; k++)
00020     {
00021         ret += to_dec(b[k]);
00022         ret += " - ";
00023     }
00024     return ret;
00025 }
00026
00027 std::string to_dec(const bit8_t& b) { return fmt::format("{:d}", b); }
00028
00029 std::string to_dec(const bit16_t& b) { return fmt::format("{:d}", b); }
00030
00031 std::string to_dec(const bit32_t& b) { return fmt::format("{:d}", b); }
00032
00033 std::string to_dec(const bit64_t& b) { return fmt::format("{:d}", b); }
00034
00035 std::string to_hex(const Buffer& b, const std::size_t& begin, const std::size_t& end)
00036 {
00037     std::size_t iend = end;
00038     if(iend == -1) iend = b.size();
00039     std::string ret;
00040     for(std::size_t k = begin; k < iend; k++)
00041     {
00042         ret += to_hex(b[k]);
00043         ret += " - ";
00044     }
00045     return ret;
00046 }
00047
00048 std::string to_hex(const bit8_t& b) { return fmt::format("{:02x}", b); }
00049
00050 std::string to_hex(const bit16_t& b) { return fmt::format("{:04x}", b); }
00051
00052 std::string to_hex(const bit32_t& b) { return fmt::format("{:08x}", b); }
00053
00054 std::string to_hex(const bit64_t& b) { return fmt::format("{:016x}", b); }
00055
00056 std::string to_bin(const Buffer& b, const std::size_t& begin, const std::size_t& end)
00057 {
00058     std::size_t iend = end;
00059     if(iend == -1) iend = b.size();
00060     std::string ret;
00061     for(std::size_t k = begin; k < iend; k++)
00062     {
00063         ret += to_bin(b[k]);
00064     }
00065 }
```

```

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

```

## 5.43 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.](#)

```

00001
00005 #include "RawBufferNavigator.h"
00006
00007 #include "Words.h"
00008
00009 int RawBufferNavigator::m_Start = 92;
00010
00011 void RawBufferNavigator::StartAt(const int& start)
00012 {
00013     if(start >= 0) m_Start = start;
00014 }
00015
00016 RawBufferNavigator::RawBufferNavigator() {}
00017
00018 void RawBufferNavigator::setBuffer(const Buffer& b)
00019 {
00020     m_Buffer = b;
00021     m_StartPayload = -1;

```

```

00022     m_StartPayloadDone = false;
00023 }
00024
00025 std::uint8_t RawBufferNavigator::getDetectorID() { return m_Buffer[0]; }
00026
00027 bool RawBufferNavigator::findStartOfPayload()
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(m_Buffer[i] == DU::START_OF_DIF || m_Buffer[i] == DU::START_OF_DIF_TEMP)
00041             {
00042                 m_StartPayload = i;
00043                 return true;
00044             }
00045         }
00046         m_StartPayload = -1;
00047         return false;
00048     }
00049 }
00050
00051 std::int32_t RawBufferNavigator::getStartOfPayload()
00052 {
00053     findStartOfPayload();
00054     return m_StartPayload;
00055 }
00056
00057 bool RawBufferNavigator::validPayload() { return m_StartPayload != -1; }
00058
00059 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.](#)

```

00001
00005 #include "Version.h"
00006
00007 const static Version streamout_version;
00008
00009 std::uint8_t Version::getMajor() { return major; }
00010
00011 std::uint8_t Version::getMinor() { return minor; }
00012
00013 std::uint8_t Version::getPatch() { return patch; }
00014
00015 std::string Version::getPreRelease()
00016 {
00017     switch(prerelease_type)

```

```

00018 {
00019     case semver::prerelease::alpha: return "alpha";
00020     case semver::prerelease::beta: return "beta";
00021     case semver::prerelease::rc: return "rc";
00022     case semver::prerelease::none: return "";
00023     default: return "";
00024 }
00025 }
00026
00027 std::uint8_t Version::getPreReleaseNumber() { return prerelease_number; }

```

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

```

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

```

### Classes

- class [textDump](#)

### 5.47.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

Definition in file [textDump.h](#).

## 5.48 textDump.h

[Go to the documentation of this file.](#)

```

00001
00005 #pragma once
00006
00007 #include "DIFPtr.h"
00008 #include "Interface.h"
00009 #include "spdlog/sinks/stdout_color_sinks.h"
00010
00011 #include <memory>
00012 #include <spdlog/logger.h>
00013
00014 class textDump : public InterfaceWriter
00015 {
00016 public:
00017     textDump();
00018     void start();
00019     void processDIF(const DIFPtr&);
00020     void processFrame(const DIFPtr&, uint32_t frameIndex);
00021     void processPadInFrame(const DIFPtr&, uint32_t frameIndex, uint32_t
channelIndex);
00022     void processSlowControl(Buffer);
00023     void end();
00024     std::shared_ptr<spdlog::logger> print() { return m_InternalLogger; }
00025     void setLevel(const spdlog::level::level_enum& level) {
m_InternalLogger->set_level(level); }
00026
00027 private:
00028     // This class is a dumb class to print on terminal so we need the logger + the standard one given by
the interface.
00029     std::shared_ptr<spdlog::logger> m_InternalLogger{nullptr};
00030 };

```

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

```
#include "textDump.h"
#include "DIFPtr.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.](#)

```
00001
00005 #include "textDump.h"
00006
00007 #include "DIFPtr.h"
00008
00009 textDump::textDump() : InterfaceWriter("textDump", "1.0.0")
00010 {
00011     m_InternalLogger = std::make_shared<spdlog::logger>("textDump",
00012         std::make_shared<spdlog::sinks::stdout_color_sink_mt>());
00013     m_InternalLogger->set_level(spdlog::level::trace);
00014     addCompatibility("RawdataReader", ">=1.0.0");
00015     addCompatibility("DIFdataExample", ">=1.0.0");
00016 }
00017 void textDump::start() { print()->info("Will dump bunch of DIF data"); }
00018
00019 void textDump::processDIF(const DIFPtr& d) { print()->info("DIF_ID : {}, DTC : {}, GTC : {}, DIF BCID
00020 {}, Absolute BCID : {}, Nbr frames {}", d.getDIFid(), d.getDTC(), d.getGTC(), d.getBCID(),
00021 d.getAbsoluteBCID(), d.getNumberOfFrames()); }
00022
00023 void textDump::processFrame(const DIFPtr& d, uint32_t frameIndex)
00024 {
00025     print()->info("\tDisplaying frame number {} : ASIC ID {}, Frame BCID {}, Frame Time To Trigger
00026 (a.k.a timestamp) is {}", frameIndex, d.getASID(), d.getFrameBCID(frameIndex),
00027 d.getFrameTimeToTrigger(frameIndex));
00028 }
00029
00030 void textDump::processPadInFrame(const DIFPtr& d, uint32_t frameIndex, uint32_t channelIndex)
00031 {
00032     if(d.getThresholdStatus(frameIndex, channelIndex) > 0) { print()->info("\t\tChannel {}, Threshold
00033 {}, channelIndex, d.getThresholdStatus(frameIndex, channelIndex)); }
00034 }
00035
00036 void textDump::processSlowControl(Buffer) { print()->error("textDump::processSlowControl not
00037 implemented yet."); }
00038
00039 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.](#)

```
00001
00005 #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.](#)

```
00001
```

## 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.](#)

```

00001
00005 #pragma once
00006
00007 #include "Interface.h"
00008
00009 #include <array>
00010 #include <cstdint>
00011 #include <fstream>
00012 #include <string>
00013 #include <vector>
00014
00015 class Buffer;
00016
00017 class RawdataReader : public InterfaceReader
00018 {
00019 public:
00020     explicit RawdataReader(const char* fileName);
00021     void start();
00022     void end();
00023     float getFileSize();
00024     void openFile(const std::string& fileName);
00025     void closeFile();
00026     bool nextEvent();
00027     bool nextDIFbuffer();
00028     const Buffer& getBuffer();
00029     virtual ~RawdataReader() { closeFile(); }
00030     static void setDefaultBufferSize(const std::size_t& size);
00031
00032 private:
00033     void uncompress();
00034     std::ifstream m_FileStream;
00035     void setFileSize(const std::size_t& size);
00036     static std::size_t m_BufferSize;
00037     std::size_t m_FileSize{0};
00038     std::uint32_t m_NumberOfDIF{0};
00039     std::uint32_t m_EventNumber{0};
00040     std::vector<bit8_t> m_buf;
00041     std::string m_Filename;
00042 };

```

## 5.57 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.](#)

```

00001
00004 #include "RawdataReader.h"
00005
00006 #include "Exception.h"
00007
00008 #include <stdint>
00009 #include <cstring>
00010 #include <stdexcept>
00011 #include <zlib.h>
00012
00014 std::size_t RawdataReader::m_BufferSize = 0x100000;
00015
00016 void RawdataReader::setDefaultBufferSize(const std::size_t& size) { m_BufferSize = size; }
00017
00018 RawdataReader::RawdataReader(const char* fileName) : InterfaceReader("RawdataReader", "1.0.0")
00019 {
00020     m_buf.reserve(m_BufferSize);
00021     m_Filename = fileName;
00022 }
00023
00024 void RawdataReader::start() { openFile(m_Filename); }
00025
00026 void RawdataReader::end() { closeFile(); }
00027
00028 void RawdataReader::uncompress()
00029 {
00030     static const std::size_t size_buffer{0x20000};
00031     std::size_t shift{3 * sizeof(std::uint32_t) + sizeof(std::uint64_t)};
00032     static bit8_t obuf[size_buffer];
00033     unsigned long size_buffer_end{0x20000}; // NOLINT(runtime/int)
00034     std::int8_t rc = ::uncompress(obuf, &size_buffer_end, &m_Buffer[shift], m_Buffer.size()
- shift);
00035     switch(rc)
00036     {
00037         case Z_OK: break;
00038         case Z_MEM_ERROR: throw Exception(Z_MEM_ERROR, "Not enough memory"); break;
00039         case Z_BUF_ERROR: throw Exception(Z_BUF_ERROR, "Not enough room in the output buffer"); break;
00040         case Z_DATA_ERROR: throw Exception(Z_DATA_ERROR, "The input data was corrupted or incomplete");
break;
00041         default: throw Exception("The input data was corrupted or incomplete"); break;
00042     }
00043     memcpy(&m_Buffer[shift], obuf, size_buffer_end);
00044     m_Buffer.setSize(size_buffer_end + shift);
00045 }
00046
00047 void RawdataReader::closeFile()
00048 {
00049     try
00050     {
00051         if(m_FileStream.is_open()) m_FileStream.close();
00052     }
00053     catch(const std::ios_base::failure& e)
00054     {
00055         log()->error("Caught an ios_base::failure in closeFile : {} {}", e.what(), e.code().value());
00056         throw;
00057     }
00058 }
00059
00060 void RawdataReader::openFile(const std::string& fileName)
00061 {
00062     try
00063     {
00064         m_FileStream.rdbuf()->pubsetbuf(0, 0);
00065         m_FileStream.exceptions(std::ifstream::failbit | std::ifstream::badbit);
00066         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
00067         m_FileStream.rdbuf()->pubsetbuf(0, 0);
00068         if(m_FileStream.is_open())
00069         {
00070             setFileSize(m_FileStream.tellg());
00071             m_FileStream.seekg(0, std::ios::beg);
00072         }
00073     }
00074     catch(const std::ios_base::failure& e)
00075     {
00076         log()->error("Caught an ios_base::failure in openFile : {} {}", e.what());
00077         throw;
00078     }
00079 }
00080
00081 bool RawdataReader::nextEvent()
00082 {
00083     try

```

```

00084 {
00085     m_FileStream.read(reinterpret_cast<char*>(&m_EventNumber), sizeof(std::uint32_t));
00086     m_FileStream.read(reinterpret_cast<char*>(&m_NumberOfDIF), sizeof(std::uint32_t));
00087 }
00088 catch(const std::ios_base::failure& e)
00089 {
00090     return false;
00091 }
00092 return true;
00093 }
00094
00095 bool RawdataReader::nextDIFbuffer()
00096 {
00097     try
00098     {
00099         static int DIF_processed{0};
00100         if(DIF_processed >= m_NumberOfDIF)
00101         {
00102             DIF_processed = 0;
00103             return false;
00104         }
00105         else
00106         {
00107             DIF_processed++;
00108             std::uint32_t bsize{0};
00109             m_FileStream.read(reinterpret_cast<char*>(&bsize), sizeof(std::uint32_t));
00110             m_FileStream.read(reinterpret_cast<char*>(&m_buf[0]), bsize);
00111             m_Buffer = Buffer(m_buf);
00112         }
00113     }
00114     catch(const std::ios_base::failure& e)
00115     {
00116         log()->error("Caught an ios_base::failure in openFile : {}", e.what());
00117         return false;
00118     }
00119     return true;
00120 }
00121
00122 const Buffer& RawdataReader::getBuffer()
00123 {
00124     uncompress();
00125     return m_Buffer;
00126 }
00127
00128 void RawdataReader::setFileSize(const std::size_t& size) { m_FileSize = size; }
00129
00130 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.](#)

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

## 5.61 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.](#)

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

## 5.63 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.](#)

```

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

```

## 5.65 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.](#)

```

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

```

## 5.67 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.](#)

```

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

```

**5.69 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.](#)

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

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

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

### Classes

- class [ROOTWriter](#)

## 5.72 ROOTWriter.h

[Go to the documentation of this file.](#)

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



## 5.73 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.](#)

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

## 5.75 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.](#)

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

## 5.77 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.](#)

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

```

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

```

## 5.79 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.](#)

```

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

```

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