

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>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 Buffer Class Reference	7
4.1.1 Detailed Description	7
4.1.2 Constructor & Destructor Documentation	7
4.1.2.1 Buffer() [1/5]	8
4.1.2.2 ~Buffer()	8
4.1.2.3 Buffer() [2/5]	8
4.1.2.4 Buffer() [3/5]	8
4.1.2.5 Buffer() [4/5]	8
4.1.2.6 Buffer() [5/5]	9
4.1.3 Member Function Documentation	9
4.1.3.1 begin()	9
4.1.3.2 capacity()	9
4.1.3.3 end()	9
4.1.3.4 operator[]() [1/2]	9
4.1.3.5 operator[]() [2/2]	10
4.1.3.6 set()	10
4.1.3.7 setSize()	10
4.1.3.8 size()	10
4.2 BufferLooper< SOURCE, DESTINATION > Class Template Reference	10
4.2.1 Detailed Description	11
4.2.2 Constructor & Destructor Documentation	11
4.2.2.1 BufferLooper()	11
4.2.3 Member Function Documentation	11
4.2.3.1 addSink()	11
4.2.3.2 log()	12
4.2.3.3 loop()	12
4.2.3.4 printAllCounters()	13
4.2.3.5 setDetectorIDs()	14
4.3 BufferLooperCounter Struct Reference	14
4.3.1 Detailed Description	14
4.3.2 Member Function Documentation	14
4.3.2.1 printAllCounters()	15
4.3.2.2 printCounter()	15

4.3.3 Member Data Documentation	15
4.3.3.1 DIFPtrValueAtReturnedPos	15
4.3.3.2 DIFStarter	15
4.3.3.3 hasBadSlowControl	16
4.3.3.4 hasSlowControl	16
4.3.3.5 NonZeroValusAtEndOfData	16
4.3.3.6 SizeAfterAllData	16
4.3.3.7 SizeAfterDIFPtr	16
4.4 DIF Class Reference	17
4.4.1 Detailed Description	17
4.4.2 Member Function Documentation	17
4.4.2.1 addHit()	17
4.4.2.2 getAbsoluteBCID()	18
4.4.2.3 getDIFBCID()	18
4.4.2.4 getDTC()	18
4.4.2.5 getGTC()	18
4.4.2.6 getID()	18
4.4.2.7 setAbsoluteBCID()	18
4.4.2.8 setDIFBCID()	19
4.4.2.9 setDTC()	19
4.4.2.10 setGTC()	19
4.4.2.11 setID()	19
4.5 DIFPtr Class Reference	19
4.5.1 Detailed Description	20
4.5.2 Member Function Documentation	20
4.5.2.1 getAbsoluteBCID()	20
4.5.2.2 getASICid()	21
4.5.2.3 getBCID()	21
4.5.2.4 getDIFid()	21
4.5.2.5 getDTC()	21
4.5.2.6 getFrameAsicHeader()	21
4.5.2.7 getFrameBCID()	22
4.5.2.8 getFrameLevel()	22
4.5.2.9 getFramePtr()	22
4.5.2.10 getFramesVector()	22
4.5.2.11 getFrameTimeToTrigger()	22
4.5.2.12 getGetFramePtrReturn()	23
4.5.2.13 getGTC()	23
4.5.2.14 getID()	23
4.5.2.15 getLines()	23
4.5.2.16 getLinesVector()	23
4.5.2.17 getNumberOfFrames()	23

4.5.2.18 getPtr()	24
4.5.2.19 getTASU1()	24
4.5.2.20 getTASU2()	24
4.5.2.21 getTDIF()	24
4.5.2.22 getTemperatureASU1()	24
4.5.2.23 getTemperatureASU2()	24
4.5.2.24 getTemperatureDIF()	25
4.5.2.25 getThresholdStatus()	25
4.5.2.26 hasAnalogReadout()	25
4.5.2.27 hasLine()	25
4.5.2.28 hasTemperature()	25
4.5.2.29 setBuffer()	26
4.6 DIFSlowControl Class Reference	26
4.6.1 Detailed Description	26
4.6.2 Constructor & Destructor Documentation	27
4.6.2.1 DIFSlowControl()	27
4.6.3 Member Function Documentation	27
4.6.3.1 Dump()	27
4.6.3.2 getChipSlowControl() [1/2]	28
4.6.3.3 getChipSlowControl() [2/2]	28
4.6.3.4 getChipsMap()	28
4.6.3.5 getDIFId()	29
4.7 DIFUnpacker Class Reference	29
4.7.1 Detailed Description	30
4.7.2 Member Function Documentation	30
4.7.2.1 getAbsoluteBCID()	30
4.7.2.2 getAnalogPtr()	30
4.7.2.3 getBCID()	30
4.7.2.4 getDTC()	31
4.7.2.5 getFrameAsicHeader()	31
4.7.2.6 getFrameBCID()	31
4.7.2.7 getFrameLevel()	31
4.7.2.8 getFramePAD()	32
4.7.2.9 getFramePtr()	32
4.7.2.10 getGTC()	33
4.7.2.11 getID()	33
4.7.2.12 getLines()	33
4.7.2.13 getStartOfDIF()	33
4.7.2.14 getTASU1()	34
4.7.2.15 getTASU2()	34
4.7.2.16 getTDIF()	34
4.7.2.17 GrayToBin()	34

4.7.2.18 hasAnalogReadout()	35
4.7.2.19 hasLine()	35
4.7.2.20 hasTemperature()	35
4.8 Event Class Reference	35
4.8.1 Detailed Description	36
4.8.2 Member Function Documentation	36
4.8.2.1 addDIF()	36
4.8.2.2 clear()	36
4.9 Hit Class Reference	36
4.9.1 Detailed Description	37
4.9.2 Member Function Documentation	37
4.9.2.1 getAbsoluteBCID()	37
4.9.2.2 getASICid()	37
4.9.2.3 getChannelId()	38
4.9.2.4 getDIFBCID()	38
4.9.2.5 getDIFid()	38
4.9.2.6 getDTC()	38
4.9.2.7 setFrameBCID()	38
4.9.2.8 getGTC()	38
4.9.2.9 getThreshold()	39
4.9.2.10 getTimestamp()	39
4.9.2.11 setAbsoluteBCID()	39
4.9.2.12 setASIC()	39
4.9.2.13 setChannel()	39
4.9.2.14 setDIF()	40
4.9.2.15 setDIFBCID()	40
4.9.2.16 setDTC()	40
4.9.2.17 setFrameBCID()	40
4.9.2.18 setGTC()	40
4.9.2.19 setThreshold()	41
4.9.2.20 setTimestamp()	41
4.10 Interface Class Reference	41
4.10.1 Detailed Description	42
4.10.2 Constructor & Destructor Documentation	42
4.10.2.1 Interface()	42
4.10.2.2 ~Interface()	42
4.10.3 Member Function Documentation	42
4.10.3.1 endDIF()	42
4.10.3.2 endEvent()	43
4.10.3.3 endFrame()	43
4.10.3.4 endPad()	43
4.10.3.5 log()	43

4.10.3.6 setLogger()	43
4.10.3.7 startDIF()	44
4.10.3.8 startEvent()	44
4.10.3.9 startFrame()	44
4.10.3.10 startPad()	44
4.11 RawBufferNavigator Class Reference	44
4.11.1 Detailed Description	45
4.11.2 Constructor & Destructor Documentation	45
4.11.2.1 RawBufferNavigator() [1/2]	45
4.11.2.2 ~RawBufferNavigator()	45
4.11.2.3 RawBufferNavigator() [2/2]	46
4.11.3 Member Function Documentation	46
4.11.3.1 badSCData()	46
4.11.3.2 getDetectorID()	46
4.11.3.3 getDIF_CRC()	46
4.11.3.4 getDIFBuffer()	46
4.11.3.5 getDIFBufferSize()	47
4.11.3.6 getDIFBufferStart()	47
4.11.3.7 getDIFPtr()	47
4.11.3.8 getEndOfAllData()	47
4.11.3.9 getEndOfDIFData()	47
4.11.3.10 getSCBuffer()	48
4.11.3.11 getSizeAfterDIFPtr()	48
4.11.3.12 getStartOfDIF()	48
4.11.3.13 hasSlowControlData()	48
4.11.3.14 setBuffer()	48
4.11.3.15 StartAt()	49
4.11.3.16 validBuffer()	49
4.12 RawdataReader Class Reference	49
4.12.1 Detailed Description	50
4.12.2 Constructor & Destructor Documentation	50
4.12.2.1 RawdataReader()	50
4.12.2.2 ~RawdataReader()	50
4.12.3 Member Function Documentation	50
4.12.3.1 closeFile()	50
4.12.3.2 end()	51
4.12.3.3 getFileSize()	51
4.12.3.4 getSDHCALBuffer()	51
4.12.3.5 nextDIFbuffer()	51
4.12.3.6 nextEvent()	52
4.12.3.7 openFile()	52
4.12.3.8 setDefaultBufferSize()	52

4.12.3.9 start()	53
4.13 ROOTWriter Class Reference	53
4.13.1 Detailed Description	53
4.13.2 Constructor & Destructor Documentation	54
4.13.2.1 ROOTWriter()	54
4.13.3 Member Function Documentation	54
4.13.3.1 end()	54
4.13.3.2 endDIF()	54
4.13.3.3 endEvent()	54
4.13.3.4 endFrame()	55
4.13.3.5 endPad()	55
4.13.3.6 processDIF()	55
4.13.3.7 processFrame()	55
4.13.3.8 processPadInFrame()	56
4.13.3.9 processSlowControl()	56
4.13.3.10 setFilename()	56
4.13.3.11 start()	56
4.13.3.12 startDIF()	56
4.13.3.13 startEvent()	57
4.13.3.14 startFrame()	57
4.13.3.15 startPad()	57
4.14 textDump Class Reference	57
4.14.1 Detailed Description	58
4.14.2 Constructor & Destructor Documentation	58
4.14.2.1 textDump()	58
4.14.3 Member Function Documentation	58
4.14.3.1 end()	58
4.14.3.2 print()	58
4.14.3.3 processDIF()	59
4.14.3.4 processFrame()	59
4.14.3.5 processPadInFrame()	59
4.14.3.6 processSlowControl()	59
4.14.3.7 setLevel()	60
4.14.3.8 start()	60
4.15 Timer Class Reference	60
4.15.1 Detailed Description	60
4.15.2 Member Function Documentation	60
4.15.2.1 getElapsedTime()	60
4.15.2.2 start()	61
4.15.2.3 stop()	61

## 5 File Documentation

63



5.1 <a href="#">libs/core/include/Bits.h File Reference</a>	63
5.1.1 Detailed Description	63
5.1.2 Typedef Documentation	63
5.1.2.1 <code>bit16_t</code>	64
5.1.2.2 <code>bit32_t</code>	64
5.1.2.3 <code>bit64_t</code>	64
5.1.2.4 <code>bit8_t</code>	64
5.1.3 Function Documentation	64
5.1.3.1 <code>operator&lt;&lt;()</code>	64
5.2 <a href="#">Bits.h</a>	65
5.3 <a href="#">libs/core/include/Buffer.h File Reference</a>	65
5.3.1 Detailed Description	65
5.4 <a href="#">Buffer.h</a>	65
5.5 <a href="#">libs/core/include/BufferLooper.h File Reference</a>	66
5.5.1 Detailed Description	66
5.6 <a href="#">BufferLooper.h</a>	67
5.7 <a href="#">libs/core/include/BufferLooperCounter.h File Reference</a>	68
5.7.1 Detailed Description	69
5.8 <a href="#">BufferLooperCounter.h</a>	69
5.9 <a href="#">libs/core/include/DetectorId.h File Reference</a>	69
5.9.1 Detailed Description	69
5.9.2 Enumeration Type Documentation	70
5.9.2.1 <code>DetectorID</code>	70
5.10 <a href="#">DetectorId.h</a>	70
5.11 <a href="#">libs/core/include/DIFPtr.h File Reference</a>	70
5.11.1 Detailed Description	71
5.12 <a href="#">DIFPtr.h</a>	71
5.13 <a href="#">libs/core/include/DIFSlowControl.h File Reference</a>	72
5.13.1 Detailed Description	72
5.14 <a href="#">DIFSlowControl.h</a>	73
5.15 <a href="#">libs/core/include/DIFUnpacker.h File Reference</a>	73
5.15.1 Detailed Description	73
5.16 <a href="#">DIFUnpacker.h</a>	74
5.17 <a href="#">libs/core/include/Formatters.h File Reference</a>	74
5.17.1 Detailed Description	75
5.17.2 Function Documentation	75
5.17.2.1 <code>to_bin()</code> [1/5]	75
5.17.2.2 <code>to_bin()</code> [2/5]	75
5.17.2.3 <code>to_bin()</code> [3/5]	75
5.17.2.4 <code>to_bin()</code> [4/5]	76
5.17.2.5 <code>to_bin()</code> [5/5]	76
5.17.2.6 <code>to_dec()</code> [1/5]	76

5.17.2.7 to_dec() [2/5]	76
5.17.2.8 to_dec() [3/5]	77
5.17.2.9 to_dec() [4/5]	77
5.17.2.10 to_dec() [5/5]	77
5.17.2.11 to_hex() [1/5]	77
5.17.2.12 to_hex() [2/5]	78
5.17.2.13 to_hex() [3/5]	78
5.17.2.14 to_hex() [4/5]	78
5.17.2.15 to_hex() [5/5]	78
5.17.2.16 to_oct() [1/5]	79
5.17.2.17 to_oct() [2/5]	79
5.17.2.18 to_oct() [3/5]	79
5.17.2.19 to_oct() [4/5]	79
5.17.2.20 to_oct() [5/5]	79
5.18 Formatters.h	80
5.19 libs/core/include/Interface.h File Reference	80
5.19.1 Detailed Description	80
5.20 Interface.h	81
5.21 libs/core/include/RawBufferNavigator.h File Reference	81
5.21.1 Detailed Description	81
5.22 RawBufferNavigator.h	82
5.23 libs/core/include/Timer.h File Reference	82
5.23.1 Detailed Description	82
5.24 Timer.h	83
5.25 libs/core/include/Words.h File Reference	83
5.25.1 Detailed Description	83
5.25.2 Enumeration Type Documentation	83
5.25.2.1 DU	83
5.26 Words.h	84
5.27 libs/core/src/Bits.cc File Reference	85
5.27.1 Detailed Description	85
5.27.2 Function Documentation	85
5.27.2.1 operator<<()	85
5.28 Bits.cc	86
5.29 libs/core/src/Buffer.cc File Reference	86
5.30 Buffer.cc	86
5.31 libs/core/src/BufferLooperCounter.cc File Reference	86
5.32 BufferLooperCounter.cc	86
5.33 libs/core/src/DIFSlowControl.cc File Reference	87
5.33.1 Detailed Description	87
5.34 DIFSlowControl.cc	87
5.35 libs/core/src/DIFUnpacker.cc File Reference	90

5.35.1 Detailed Description	90
5.36 DIFUnpacker.cc	91
5.37 libs/core/src/Formatters.cc File Reference	93
5.37.1 Detailed Description	93
5.37.2 Function Documentation	93
5.37.2.1 to_bin() [1/5]	93
5.37.2.2 to_bin() [2/5]	94
5.37.2.3 to_bin() [3/5]	94
5.37.2.4 to_bin() [4/5]	94
5.37.2.5 to_bin() [5/5]	94
5.37.2.6 to_dec() [1/5]	95
5.37.2.7 to_dec() [2/5]	95
5.37.2.8 to_dec() [3/5]	95
5.37.2.9 to_dec() [4/5]	95
5.37.2.10 to_dec() [5/5]	95
5.37.2.11 to_hex() [1/5]	96
5.37.2.12 to_hex() [2/5]	96
5.37.2.13 to_hex() [3/5]	96
5.37.2.14 to_hex() [4/5]	96
5.37.2.15 to_hex() [5/5]	96
5.37.2.16 to_oct() [1/5]	97
5.37.2.17 to_oct() [2/5]	97
5.37.2.18 to_oct() [3/5]	97
5.37.2.19 to_oct() [4/5]	97
5.37.2.20 to_oct() [5/5]	97
5.38 Formatters.cc	98
5.39 libs/core/src/RawBufferNavigator.cc File Reference	99
5.39.1 Detailed Description	99
5.40 RawBufferNavigator.cc	99
5.41 libs/interface/Dump/include/textDump.h File Reference	100
5.41.1 Detailed Description	101
5.42 textDump.h	101
5.43 libs/interface/Dump/src/textDump.cc File Reference	101
5.43.1 Detailed Description	101
5.44 textDump.cc	102
5.45 libs/interface/LCIO/include/LCIOWriter.h File Reference	102
5.45.1 Detailed Description	102
5.46 LCIOWriter.h	102
5.47 libs/interface/LCIO/src/LCIOWriter.cc File Reference	102
5.47.1 Detailed Description	102
5.48 LCIOWriter.cc	103
5.49 libs/interface/RawDataReader/include/RawdataReader.h File Reference	103

5.49.1 Detailed Description . . . . .	103
5.50 RawdataReader.h . . . . .	103
5.51 libs/interface/RawDataReader/src/RawdataReader.cc File Reference . . . . .	104
5.51.1 Detailed Description . . . . .	104
5.52 RawdataReader.cc . . . . .	104
5.53 libs/interface/ROOT/include/DIF.h File Reference . . . . .	106
5.53.1 Detailed Description . . . . .	106
5.54 DIF.h . . . . .	106
5.55 libs/interface/ROOT/include/DIFLinkDef.h File Reference . . . . .	107
5.55.1 Detailed Description . . . . .	107
5.56 DIFLinkDef.h . . . . .	107
5.57 libs/interface/ROOT/include/Event.h File Reference . . . . .	107
5.57.1 Detailed Description . . . . .	107
5.58 Event.h . . . . .	108
5.59 libs/interface/ROOT/include/EventLinkDef.h File Reference . . . . .	108
5.59.1 Detailed Description . . . . .	108
5.60 EventLinkDef.h . . . . .	108
5.61 libs/interface/ROOT/include/Hit.h File Reference . . . . .	108
5.61.1 Detailed Description . . . . .	109
5.62 Hit.h . . . . .	109
5.63 libs/interface/ROOT/include/HitLinkDef.h File Reference . . . . .	109
5.63.1 Detailed Description . . . . .	109
5.64 HitLinkDef.h . . . . .	110
5.65 libs/interface/ROOT/include/ROOTWriter.h File Reference . . . . .	110
5.66 ROOTWriter.h . . . . .	110
5.67 libs/interface/ROOT/src/DIF.cc File Reference . . . . .	111
5.67.1 Detailed Description . . . . .	111
5.68 DIF.cc . . . . .	111
5.69 libs/interface/ROOT/src/Event.cc File Reference . . . . .	111
5.69.1 Detailed Description . . . . .	112
5.70 Event.cc . . . . .	112
5.71 libs/interface/ROOT/src/Hit.cc File Reference . . . . .	112
5.71.1 Detailed Description . . . . .	112
5.72 Hit.cc . . . . .	112
5.73 libs/interface/ROOT/src/ROOTWriter.cc File Reference . . . . .	113
5.73.1 Detailed Description . . . . .	113
5.74 ROOTWriter.cc . . . . .	113

# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

Buffer	7
BufferLooper< SOURCE, DESTINATION >	10
BufferLooperCounter	14
DIFPtr	19
DIFSlowControl	26
DIFUnpacker	29
Interface	41
ROOTWriter	53
RawdataReader	49
textDump	57
RawBufferNavigator	44
Timer	60
TObject	
DIF	17
Event	35
Hit	36



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">Buffer</a>	7
<a href="#">BufferLooper&lt; SOURCE, DESTINATION &gt;</a>	10
<a href="#">BufferLooperCounter</a>	14
<a href="#">DIF</a>	17
<a href="#">DIFPtr</a>	19
<a href="#">DIFSlowControl</a>	
Handler of <a href="#">DIF</a> Slow Control info	26
<a href="#">DIFUnpacker</a>	29
<a href="#">Event</a>	35
<a href="#">Hit</a>	36
<a href="#">Interface</a>	
Template class should implement void SOURCE::start(); bool SOURCE::next(); void SOURCE↔ ::end(); const <a href="#">Buffer</a> & SOURCE::getSDHCALBuffer();	41
<a href="#">RawBufferNavigator</a>	44
<a href="#">RawdataReader</a>	49
<a href="#">ROOTWriter</a>	53
<a href="#">textDump</a>	57
<a href="#">Timer</a>	60





## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

libs/core/include/Bits.h	63
libs/core/include/Buffer.h	65
libs/core/include/BufferLooper.h	66
libs/core/include/BufferLooperCounter.h	68
libs/core/include/DetectorId.h	69
libs/core/include/DIFPtr.h	70
libs/core/include/DIFSlowControl.h	72
libs/core/include/DIFUnpacker.h	73
libs/core/include/Formatters.h	74
libs/core/include/Interface.h	80
libs/core/include/RawBufferNavigator.h	81
libs/core/include/Timer.h	82
libs/core/include/Words.h	83
libs/core/src/Bits.cc	85
libs/core/src/Buffer.cc	86
libs/core/src/BufferLooperCounter.cc	86
libs/core/src/DIFSlowControl.cc	87
libs/core/src/DIFUnpacker.cc	90
libs/core/src/Formatters.cc	93
libs/core/src/RawBufferNavigator.cc	99
libs/interface/Dump/include/textDump.h	100
libs/interface/Dump/src/textDump.cc	101
libs/interface/LCIO/include/LCIOWriter.h	102
libs/interface/LCIO/src/LCIOWriter.cc	102
libs/interface/RawDataReader/include/RawdataReader.h	103
libs/interface/RawDataReader/src/RawdataReader.cc	104
libs/interface/ROOT/include/DIF.h	106
libs/interface/ROOT/include/DIFLinkDef.h	107
libs/interface/ROOT/include/Event.h	107
libs/interface/ROOT/include/EventLinkDef.h	108
libs/interface/ROOT/include/Hit.h	108
libs/interface/ROOT/include/HitLinkDef.h	109
libs/interface/ROOT/include/ROOTWriter.h	110
libs/interface/ROOT/src/DIF.cc	111
libs/interface/ROOT/src/Event.cc	111
libs/interface/ROOT/src/Hit.cc	112
libs/interface/ROOT/src/ROOTWriter.cc	113



## Chapter 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 [13](#) of file [Buffer.h](#).

#### 4.1.2 Constructor & Destructor Documentation

**4.1.2.1 Buffer()** [1/5]

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

Definition at line 16 of file [Buffer.h](#).

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

**4.1.2.2 ~Buffer()**

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

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

```
00017 {}
```

**4.1.2.3 Buffer()** [2/5]

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

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

```
00018 : 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 19 of file [Buffer.h](#).

```
00019 : 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 20 of file [Buffer.h](#).

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

### 4.1.3 Member Function Documentation

#### 4.1.3.1 begin()

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

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

```
00027 { return m_Buffer; }
```

#### 4.1.3.2 capacity()

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

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

```
00024 { return m_Capacity; }
```

#### 4.1.3.3 end()

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

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

```
00028 { 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 29 of file [Buffer.h](#).

```
00029 { 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 30 of file [Buffer.h](#).

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

#### 4.1.3.6 set()

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

Definition at line 26 of file [Buffer.h](#).

```
00026 { m_Buffer = b; }
```

#### 4.1.3.7 setSize()

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

Definition at line 32 of file [Buffer.h](#).

```
00032 { m_Size = size; }
```

#### 4.1.3.8 size()

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

Definition at line 23 of file [Buffer.h](#).

```
00023 { 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 23 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 26 of file [BufferLooper.h](#).

```
00026                                     : m_Source(source),
    m_Destination(dest), m_Debug(debug)
00027 {
00028     m_Logger = spdlog::create<spdlog::sinks::null_sink_mt>("streamout");
00029     if(!spdlog::get("streamout")) { spdlog::register_logger(m_Logger); }
00030     m_Source.setLogger(m_Logger);
00031     m_Destination.setLogger(m_Logger);
00032 }
```

### 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 34 of file [BufferLooper.h](#).

```
00035 {
00036     sink->set_level(level);
00037     m_Sinks.push_back(sink);
00038     m_Logger = std::make_shared<spdlog::logger>("streamout", begin(m_Sinks), end(m_Sinks));
00039     m_Source.setLogger(m_Logger);
00040     m_Destination.setLogger(m_Logger);
00041 }
```

#### 4.2.3.2 log()

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

Definition at line 155 of file [BufferLooper.h](#).

```
00155 { 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 43 of file [BufferLooper.h](#).

```
00044 {
00045     Timer timer;
00046     timer.start();
00047     m_Source.start();
00048     m_Destination.start();
00049     RawBufferNavigator bufferNavigator;
00050     while(m_Source.nextEvent() && m_NbrEventsToProcess >= m_NbrEvents)
00051     {
00053         m_Source.startEvent();
00054         m_Destination.startEvent();
00056
00057         m_Logger->warn("====* Event number {} *====", m_NbrEvents);
00058         while(m_Source.nextDIFbuffer())
00059         {
00060             const Buffer& buffer = m_Source.getSDHCALBuffer();
00061             bufferNavigator.setBuffer(buffer);
00062
00063             bit8_t* debug_variable_1 = buffer.end();
00064             bit8_t* debug_variable_2 = bufferNavigator.getDIFBuffer().end();
00065             if(debug_variable_1 != debug_variable_2) m_Logger->info("DIF BUFFER END {} {}",
fmt::ptr(debug_variable_1), fmt::ptr(debug_variable_2));
00066             if(m_Debug) assert(debug_variable_1 == debug_variable_2);
00067
00068             if(std::find(m_DetectorIDs.begin(), m_DetectorIDs.end(),
static_cast<DetectorID>(bufferNavigator.getDetectorID())) == m_DetectorIDs.end())
00069             {
00070                 m_Logger->trace("{} ", bufferNavigator.getDetectorID());
00071                 continue;
00072             }
00073
00075             m_Source.startDIF();
00076             m_Destination.startDIF();
00078
00079             uint32_t idstart = bufferNavigator.getStartOfDIF();
00080             if(m_Debug && idstart == 0) m_Logger->info(to_hex(buffer));
00081             c.DIFStarter[idstart]++;
00082             if(!bufferNavigator.validBuffer())
00083             {
00084                 m_Logger->error("!bufferNavigator.validBuffer()");
00085                 continue;
00086             }
00087             DIFPtr& d = bufferNavigator.getDIFPtr();
```



```

00088         c.DIFPtrValueAtReturnedPos[bufferNavigator.getDIFBufferStart() [d.getGetFramePtrReturn()]]++;
00089         if(m_Debug) assert(bufferNavigator.getDIFBufferStart() [d.getGetFramePtrReturn()] == 0xa0);
00090         c.SizeAfterDIFPtr[bufferNavigator.getSizeAfterDIFPtr()];
00091         m_Destination.processDIF(d);
00092         for(std::size_t i = 0; i < d.getNumberOfFrames(); ++i)
00093         {
00094             m_Source.startFrame();
00095             m_Destination.startFrame();
00096             m_Destination.processFrame(d, i);
00097             for(std::size_t j = 0; j < DU::NUMBER_PAD; ++j)
00098             {
00099                 m_Source.startPad();
00100                 m_Destination.startPad();
00101                 m_Destination.processPadInFrame(d, i, j);
00102                 m_Source.endPad();
00103                 m_Destination.endPad();
00104             }
00105             m_Source.endFrame();
00106             m_Destination.endFrame();
00107         }
00108         bool processSC = false;
00109         if(bufferNavigator.hasSlowControlData())
00110         {
00111             c.hasSlowControl++;
00112             processSC = true;
00113         }
00114         if(bufferNavigator.badSCData())
00115         {
00116             c.hasBadSlowControl++;
00117             processSC = false;
00118         }
00119         if(processSC) { m_Destination.processSlowControl(bufferNavigator.getSCBuffer()); }
00120
00121         Buffer eod = bufferNavigator.getEndOfAllData();
00122         c.SizeAfterAllData[eod.size()];
00123         bit8_t* debug_variable_3 = eod.end();
00124         if(debug_variable_1 != debug_variable_3) m_Logger->info("END DATA BUFFER END {} {}",
00125             fmt::ptr(debug_variable_1), fmt::ptr(debug_variable_3));
00126         if(m_Debug) assert(debug_variable_1 == debug_variable_3);
00127         if(eod.size() != 0) m_Logger->info("End of Data remaining stuff : {}", to_hex(eod));
00128
00129         int nonzeroCount = 0;
00130         for(bit8_t* it = eod.begin(); it != eod.end(); it++)
00131             if(static_cast<int>(*it) != 0) nonzeroCount++;
00132         c.NonZeroValueAtEndOfData[nonzeroCount]++;
00133         m_Source.endDIF();
00134         m_Destination.endDIF();
00135     } // end of DIF while loop
00136     m_Logger->warn("***** Event number {} *****", m_NbrEvents);
00137     m_NbrEvents++;
00138     m_Source.endEvent();
00139     m_Destination.endEvent();
00140 } // end of event while loop
00141 m_Destination.end();
00142 m_Source.end();
00143 timer.stop();
00144 fmt::print("=== elapsed time {}ms ({}ms/event) ===\n", timer.getElapsedTime() / 1000,
00145     timer.getElapsedTime() / (1000 * m_NbrEvents));
00146 }

```

#### 4.2.3.4 printAllCounters()

```

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

```

Definition at line 154 of file [BufferLooper.h](#).

```
00154 { 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 157 of file [BufferLooper.h](#).

```
00157 { m_DetectorIDs = detectorIDs; }
```

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

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

### 4.3 BufferLooperCounter Struct Reference

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

#### Public Member Functions

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

#### Public Attributes

- int [hasSlowControl](#) = 0
- int [hasBadSlowControl](#) = 0
- std::map< int, int > [DIFStarter](#)
- std::map< int, int > [DIFPtrValueAtReturnedPos](#)
- std::map< int, int > [SizeAfterDIFPtr](#)
- std::map< int, int > [SizeAfterAllData](#)
- std::map< int, int > [NonZeroValusAtEndOfData](#)

#### 4.3.1 Detailed Description

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

#### 4.3.2 Member Function Documentation

### 4.3.2.1 printAllCounters()

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

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

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

### 4.3.2.2 printCounter()

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

Definition at line 20 of file [BufferLooperCounter.cc](#).

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

## 4.3.3 Member Data Documentation

### 4.3.3.1 DIFPtrValueAtReturnedPos

```
std::map<int, int> BufferLooperCounter::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](#).

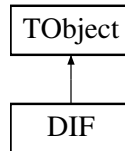
The documentation for this struct was generated from the following files:

- [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 [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

### 4.4.1 Detailed Description

Definition at line 13 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 getAbsoluteBCID()

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

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

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

#### 4.4.2.3 getDIFBCID()

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

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

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

#### 4.4.2.4 getDTC()

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

Definition at line 18 of file [DIF.cc](#).

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

#### 4.4.2.5 getGTC()

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

Definition at line 22 of file [DIF.cc](#).

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

#### 4.4.2.6 getID()

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

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

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

#### 4.4.2.7 setAbsoluteBCID()

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

Definition at line 28 of file [DIF.cc](#).

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

#### 4.4.2.8 setDIFBCID()

```
void DIF::setDIFBCID (
    const std::uint32_t & difbcid )
```

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

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

#### 4.4.2.9 setDTC()

```
void DIF::setDTC (
    const std::uint32_t & dtc )
```

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

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

#### 4.4.2.10 setGTC()

```
void DIF::setGTC (
    const std::uint32_t & gtc )
```

Definition at line 20 of file [DIF.cc](#).

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

#### 4.4.2.11 setID()

```
void DIF::setID (
    const std::uint8_t & id )
```

Definition at line 12 of file [DIF.cc](#).

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

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

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

## 4.5 DIFPtr Class Reference

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

## Public Member Functions

- void [setBuffer](#) (unsigned char \*, const std::uint32\_t &)
- unsigned char \* [getPtr](#) () const
- std::uint32\_t [getGetFramePtrReturn](#) () const
- std::vector< unsigned char \* > & [getFramesVector](#) ()
- std::vector< unsigned char \* > & [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
- unsigned char \* [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
- uint32\_t [getDIFid](#) () const
- uint32\_t [getASICid](#) (const std::uint32\_t &) const
- uint32\_t [getThresholdStatus](#) (const std::uint32\_t &, const std::uint32\_t &) const

### 4.5.1 Detailed Description

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

### 4.5.2 Member Function Documentation

#### 4.5.2.1 [getAbsoluteBCID\(\)](#)

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

Definition at line 79 of file [DIFPtr.h](#).

```
00079 { return DIFUnpacker::getAbsoluteBCID(theDIF_); }
```



#### 4.5.2.2 getASICid()

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

Definition at line 99 of file [DIFPtr.h](#).

```
00099 { return getFrameAsicHeader(i) & 0xFF; }
```

#### 4.5.2.3 getBCID()

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

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

```
00080 { return DIFUnpacker::getBCID(theDIF_); }
```

#### 4.5.2.4 getDIFid()

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

Definition at line 98 of file [DIFPtr.h](#).

```
00098 { return getID() & 0xFF; }
```

#### 4.5.2.5 getDTC()

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

Definition at line 77 of file [DIFPtr.h](#).

```
00077 { return DIFUnpacker::getDTC(theDIF_); }
```

#### 4.5.2.6 getFrameAsicHeader()

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

Definition at line 93 of file [DIFPtr.h](#).

```
00093 { return DIFUnpacker::getFrameAsicHeader(theFrames_[i]); }
```

#### 4.5.2.7 getFrameBCID()

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

Definition at line 94 of file [DIFPtr.h](#).

```
00094 { return DIFUnpacker::getFrameBCID(theFrames_[i]); }
```

#### 4.5.2.8 getFrameLevel()

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

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

```
00096 { return DIFUnpacker::getFrameLevel(theFrames_[i], ipad, ilevel); }
```

#### 4.5.2.9 getFramePtr()

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

Definition at line 92 of file [DIFPtr.h](#).

```
00092 { return theFrames_[i]; }
```

#### 4.5.2.10 getFramesVector()

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

Definition at line 74 of file [DIFPtr.h](#).

```
00074 { return theFrames_; }
```

#### 4.5.2.11 getFrameTimeToTrigger()

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

Definition at line 95 of file [DIFPtr.h](#).

```
00095 { return getBCID() - getFrameBCID(i); }
```

#### 4.5.2.12 getGetFramePtrReturn()

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

Definition at line 73 of file [DIFPtr.h](#).

```
00073 { return theGetFramePtrReturn_; }
```

#### 4.5.2.13 getGTC()

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

Definition at line 78 of file [DIFPtr.h](#).

```
00078 { return DIFUnpacker::getGTC(theDIF_); }
```

#### 4.5.2.14 getID()

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

Definition at line 76 of file [DIFPtr.h](#).

```
00076 { return DIFUnpacker::getID(theDIF_); }
```

#### 4.5.2.15 getLines()

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

Definition at line 81 of file [DIFPtr.h](#).

```
00081 { return DIFUnpacker::getLines(theDIF_); }
```

#### 4.5.2.16 getLinesVector()

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

Definition at line 75 of file [DIFPtr.h](#).

```
00075 { return theLines_; }
```

#### 4.5.2.17 getNumberOfFrames()

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

Definition at line 91 of file [DIFPtr.h](#).

```
00091 { return theFrames_.size(); }
```

#### 4.5.2.18 getPtr()

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

Definition at line 72 of file [DIFPtr.h](#).

```
00072 { return theDIF_; }
```

#### 4.5.2.19 getTASU1()

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

Definition at line 83 of file [DIFPtr.h](#).

```
00083 { return DIFUnpacker::getTASU1(theDIF_); }
```

#### 4.5.2.20 getTASU2()

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

Definition at line 84 of file [DIFPtr.h](#).

```
00084 { return DIFUnpacker::getTASU2(theDIF_); }
```

#### 4.5.2.21 getTDIF()

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

Definition at line 85 of file [DIFPtr.h](#).

```
00085 { return DIFUnpacker::getTDIF(theDIF_); }
```

#### 4.5.2.22 getTemperatureASU1()

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

Definition at line 87 of file [DIFPtr.h](#).

```
00087 { return (getTASU1() » 3) * 0.0625; }
```

#### 4.5.2.23 getTemperatureASU2()

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

Definition at line 88 of file [DIFPtr.h](#).

```
00088 { return (getTASU2() » 3) * 0.0625; }
```

#### 4.5.2.24 getTemperatureDIF()

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

Definition at line 86 of file [DIFPtr.h](#).

```
00086 { return 0.508 * getTDIF() - 9.659; }
```

#### 4.5.2.25 getThresholdStatus()

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

Definition at line 100 of file [DIFPtr.h](#).

```
00100 { return (((uint32_t)getFrameLevel(i, ipad, 1)) < 1) | ((uint32_t)getFrameLevel(i, ipad, 0)); }
```

#### 4.5.2.26 hasAnalogReadout()

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

Definition at line 90 of file [DIFPtr.h](#).

```
00090 { return DIFUnpacker::hasAnalogReadout(theDIF_); }
```

#### 4.5.2.27 hasLine()

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

Definition at line 82 of file [DIFPtr.h](#).

```
00082 { return DIFUnpacker::hasLine(line, theDIF_); }
```

#### 4.5.2.28 hasTemperature()

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

Definition at line 89 of file [DIFPtr.h](#).

```
00089 { return DIFUnpacker::hasTemperature(theDIF_); }
```

#### 4.5.2.29 setBuffer()

```
void DIFPtr::setBuffer (
    unsigned char * p,
    const std::uint32_t & max_size ) [inline]
```

Definition at line 56 of file [DIFPtr.h](#).

```
00057 {
00058     theFrames_.clear();
00059     theLines_.clear();
00060     theSize_ = max_size;
00061     theDIF_ = p;
00062     try
00063     {
00064         theGetFramePtrReturn_ = DIFUnpacker::getFramePtr(theFrames_, theLines_, theSize_, theDIF_);
00065     }
00066     catch(const std::string& e)
00067     {
00068         spdlog::get("streamout")->error(" DIF {} T ? {} {} ", getID(), hasTemperature(), e);
00069     }
00070 }
```

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

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

## 4.6 DIFSlowControl Class Reference

Handler of [DIF](#) Slow Control info.

```
#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.*
- void [Dump](#) ()  
*print out full map*

#### 4.6.1 Detailed Description

Handler of [DIF](#) Slow Control info.

Author

L.Mirabito

Date

March 2010

Version

1.0

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

## 4.6.2 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 10 of file [DIFSlowControl.cc](#).

```
00010 : m_Version(version), m_DIFid(DifId), m_AsicType(2)
00011 {
00012     if(cbuf[0] != 0xb1) return;
00013     int header_shift{6};
00014     if(m_Version < 8) m_NbrAsic = cbuf[5];
00015     else
00016     {
00017         m_DIFid      = cbuf[1];
00018         m_NbrAsic     = cbuf[2];
00019         header_shift = 3;
00020     }
00021     int size_hardroc1 = m_NbrAsic * 72 + header_shift + 1;
00022     if(cbuf[size_hardroc1 - 1] != 0xal) size_hardroc1 = 0;
00023
00024     int size_hardroc2 = m_NbrAsic * 109 + header_shift + 1;
00025     if(cbuf[size_hardroc2 - 1] != 0xal) size_hardroc2 = 0;
00026     if(size_hardroc1 != 0)
00027     {
00028         FillHR1(header_shift, cbuf);
00029         m_AsicType = 1;
00030     }
00031     else if(size_hardroc2 != 0)
00032         FillHR2(header_shift, cbuf);
00033     else
00034         return;
00035 }
```

## 4.6.3 Member Function Documentation

### 4.6.3.1 Dump()

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

print out full map

Definition at line 45 of file [DIFSlowControl.cc](#).

```
00046 {
```

```

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

```

#### 4.6.3.2 getChipSlowControl() [1/2]

```

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

```

Get one chip map.

##### Parameters

<i>asicid</i>	ASIC ID
---------------	---------

##### Returns

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

Definition at line 41 of file [DIFSlowControl.cc](#).

```

00041 { return m_MapSC[asicid]; }

```

#### 4.6.3.3 getChipSlowControl() [2/2]

```

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

```

Get one Chip value.

##### Parameters

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

Definition at line 43 of file [DIFSlowControl.cc](#).

```

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

```

#### 4.6.3.4 getChipsMap()

```

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

```

Get chips map.



**Returns**

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

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

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

**4.6.3.5 getDIFId()**

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

get DIF id

Definition at line 37 of file [DIFSlowControl.cc](#).

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

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

- [libs/core/include/DIFSlowControl.h](#)
- [libs/core/src/DIFSlowControl.cc](#)

**4.7 DIFUnpacker Class Reference**

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

**Static Public Member Functions**

- static std::uint64\_t [GrayToBin](#) (const std::uint64\_t &n)
- static std::uint32\_t [getStartOfDIF](#) (const unsigned char \*cbuf, const std::uint32\_t &size\_buf, const std::uint32\_t &start=92)
- static std::uint32\_t [getId](#) (const unsigned char \*cb, const std::uint32\_t &idx=0)
- static std::uint32\_t [getDTC](#) (const unsigned char \*cb, const std::uint32\_t &idx=0)
- static std::uint32\_t [getGTC](#) (const unsigned char \*cb, const std::uint32\_t &idx=0)
- static std::uint64\_t [getAbsoluteBCID](#) (const unsigned char \*cb, const std::uint32\_t &idx=0)
- static std::uint32\_t [getBCID](#) (const unsigned char \*cb, const std::uint32\_t &idx=0)
- static std::uint32\_t [getLines](#) (const unsigned char \*cb, const std::uint32\_t &idx=0)
- static bool [hasLine](#) (const std::uint32\_t &line, const unsigned char \*cb, const std::uint32\_t &idx=0)
- static std::uint32\_t [getTASU1](#) (const unsigned char \*cb, const std::uint32\_t &idx=0)
- static std::uint32\_t [getTASU2](#) (const unsigned char \*cb, const std::uint32\_t &idx=0)
- static std::uint32\_t [getTDIF](#) (const unsigned char \*cb, const std::uint32\_t &idx=0)
- static bool [hasTemperature](#) (const unsigned char \*cb, const std::uint32\_t &idx=0)
- static bool [hasAnalogReadout](#) (const unsigned char \*cb, const std::uint32\_t &idx=0)
- static std::uint32\_t [getFrameAsicHeader](#) (const unsigned char \*framePtr)
- static std::uint32\_t [getFrameBCID](#) (const unsigned char \*framePtr)
- static bool [getFramePAD](#) (const unsigned char \*framePtr, const std::uint32\_t &ip)
- static bool [getFrameLevel](#) (const unsigned char \*framePtr, const std::uint32\_t &ip, const std::uint32\_t &level)
- static std::uint32\_t [getAnalogPtr](#) (std::vector< unsigned char \* > &vLines, unsigned char \*cb, const std::uint32\_t &idx=0)
- static std::uint32\_t [getFramePtr](#) (std::vector< unsigned char \* > &vFrame, std::vector< unsigned char \* > &vLines, const std::uint32\_t &max\_size, unsigned char \*cb, const std::uint32\_t &idx=0)

### 4.7.1 Detailed Description

Definition at line 10 of file [DIFUnpacker.h](#).

### 4.7.2 Member Function Documentation

#### 4.7.2.1 getAbsoluteBCID()

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

Definition at line 53 of file [DIFUnpacker.cc](#).

```
00054 {
00055     std::uint64_t Shift{16777216ULL}; // to shift the value from the 24 first bits
00056     std::uint64_t pos{idx + DU::BCID_SHIFT};
00057     std::uint64_t LBC = ((cb[pos] << 16) | (cb[pos + 1] << 8) | (cb[pos + 2])) * Shift + ((cb[pos + 3] <<
16) | (cb[pos + 4] << 8) | (cb[pos + 5]));
00058     return LBC;
00059 }
```

#### 4.7.2.2 getAnalogPtr()

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

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

```
00093 {
00094     std::uint32_t fshift{idx};
00095     if(cb[fshift] != DU::START_OF_LINES) return fshift;
00096     fshift++;
00097     while(cb[fshift] != DU::END_OF_LINES)
00098     {
00099         vLines.push_back(&cb[fshift]);
00100         std::uint32_t nchip{cb[fshift]};
00101         fshift += 1 + nchip * 64 * 2;
00102     }
00103     return fshift++;
00104 }
```

#### 4.7.2.3 getBCID()

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

Definition at line 61 of file [DIFUnpacker.cc](#).

```
00061 { return (cb[idx + DU::BCID_SHIFT] << 16) + (cb[idx + DU::BCID_SHIFT + 1] << 8) + cb[idx +
DU::BCID_SHIFT + 2]; }
```

#### 4.7.2.4 getDTC()

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

Definition at line 49 of file [DIFUnpacker.cc](#).

```
00049 { return (cb[idx + DU::DTC_SHIFT] « 24) + (cb[idx + DU::DTC_SHIFT + 1] « 16) + (cb[idx + DU::DTC_SHIFT
+ 2] « 8) + cb[idx + DU::DTC_SHIFT + 3]; }
```

#### 4.7.2.5 getFrameAsicHeader()

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

Definition at line 76 of file [DIFUnpacker.cc](#).

```
00076 { return (framePtr[DU::FRAME_ASIC_HEADER_SHIFT]); }
```

#### 4.7.2.6 getFrameBCID()

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

Definition at line 78 of file [DIFUnpacker.cc](#).

```
00079 {
00080     std::uint32_t igray = (framePtr[DU::FRAME_BCID_SHIFT] « 16) + (framePtr[DU::FRAME_BCID_SHIFT + 1] «
8) + framePtr[DU::FRAME_BCID_SHIFT + 2];
00081     return DIFUnpacker::GrayToBin(igray);
00082 }
```

#### 4.7.2.7 getFrameLevel()

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

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

```
00090 { return ((framePtr[DU::FRAME_DATA_SHIFT + ((3 - ip / 16) * 4 + (ip % 16) / 4)] » (7 - ((ip % 16) %
4) * 2 + level))) & 0x1); }
```

#### 4.7.2.8 getFramePAD()

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

Definition at line 84 of file [DIFUnpacker.cc](#).

```
00085 {
00086     std::uint32_t* iframe{(std::uint32_t*)&framePtr[DU::FRAME_DATA_SHIFT]};
00087     return ((iframe[3 - ip / 32] » (ip % 32)) & 0x1);
00088 }
```

#### 4.7.2.9 getFramePtr()

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

Definition at line 106 of file [DIFUnpacker.cc](#).

```
00107 {
00108     std::uint32_t fshift{0};
00109     if(DATA_FORMAT_VERSION >= 13)
00110     {
00111         fshift = idx + DU::LINES_SHIFT + 1;
00112         if(DIFUnpacker::hasTemperature(cb, idx)) fshift = idx + DU::TDIF_SHIFT + 1;
00113         // jenlev 1
00114         if(DIFUnpacker::hasAnalogReadout(cb, idx)) fshift = DIFUnpacker::getAnalogPtr(vLines, cb, fshift);
00115         // to be implemented
00116     }
00117     else
00118     {
00119         fshift = idx + DU::BCID_SHIFT + 3;
00120         if(cb[fshift] != DU::START_OF_FRAME)
00121         {
00122             std::cout << "This is not a start of frame " << to_hex(cb[fshift]) << " \n";
00123             return fshift;
00124         }
00125         do {
00126             // printf("fshift %d and %d \n", fshift, max_size);
00127             if(cb[fshift] == DU::END_OF_DIF) return fshift;
00128             if(cb[fshift] == DU::START_OF_FRAME) fshift++;
00129             if(cb[fshift] == DU::END_OF_FRAME)
00130             {
00131                 fshift++;
00132                 continue;
00133             }
00134             std::uint32_t header = DIFUnpacker::getFrameAsicHeader(&cb[fshift]);
00135             if(header == DU::END_OF_FRAME) return (fshift + 2);
00136             // std::cout<<header<< " " << fshift << std::endl;
00137             if(header < 1 || header > 48) { throw header + " Header problem " + fshift; }
00138             vFrame.push_back(&cb[fshift]);
00139             fshift += DU::FRAME_SIZE;
00140             if(fshift > max_size)
00141             {
00142                 std::cout << "fshift " << fshift << " exceed " << max_size << " \n";
00143                 return fshift;
00144             }
00145             if(cb[fshift] == DU::END_OF_FRAME) fshift++;
00146         } while(true);
00147     }
00148 }
```

#### 4.7.2.10 getGTC()

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

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

```
00051 { return (cb[idx + DU::GTC_SHIFT] << 24) + (cb[idx + DU::GTC_SHIFT + 1] << 16) + (cb[idx + DU::GTC_SHIFT
+ 2] << 8) + cb[idx + DU::GTC_SHIFT + 3]; }
```

#### 4.7.2.11 getID()

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

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

```
00047 { return cb[idx + DU::ID_SHIFT]; }
```

#### 4.7.2.12 getLines()

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

Definition at line 62 of file [DIFUnpacker.cc](#).

```
00062 { return (cb[idx + DU::LINES_SHIFT] >> 4) & 0x5; }
```

#### 4.7.2.13 getStartOfDIF()

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

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

```
00031 {
00032     std::uint32_t id0{0};
00033     for(std::uint32_t i = start; i < size_buf; i++)
00034     {
00035         if(cbuf[i] != DU::START_OF_DIF && cbuf[i] != DU::START_OF_DIF_TEMP) continue;
00036         else
00037         {
00038             id0 = i;
00039             break;
00040         }
00041         // if (cbuf[id0+DU::ID_SHIFT]>0xFF) continue;
00042     }
00043     // std::cout << "***** " << id0 << std::endl;
00044     return id0;
00045 }
```

#### 4.7.2.14 getTASU1()

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

Definition at line 66 of file [DIFUnpacker.cc](#).

```
00066 { return (cb[idx + DU::TASU1_SHIFT] << 24) + (cb[idx + DU::TASU1_SHIFT + 1] << 16) + (cb[idx +
    DU::TASU1_SHIFT + 2] << 8) + cb[idx + DU::TASU1_SHIFT + 3]; }
```

#### 4.7.2.15 getTASU2()

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

Definition at line 68 of file [DIFUnpacker.cc](#).

```
00068 { return (cb[idx + DU::TASU2_SHIFT] << 24) + (cb[idx + DU::TASU2_SHIFT + 1] << 16) + (cb[idx +
    DU::TASU2_SHIFT + 2] << 8) + cb[idx + DU::TASU2_SHIFT + 3]; }
```

#### 4.7.2.16 getTDIF()

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

Definition at line 70 of file [DIFUnpacker.cc](#).

```
00070 { return (cb[idx + DU::TDIF_SHIFT]); }
```

#### 4.7.2.17 GrayToBin()

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

Definition at line 15 of file [DIFUnpacker.cc](#).

```
00016 {
00017     std::uint64_t ish{1};
00018     std::uint64_t anss{n};
00019     std::uint64_t idiv{0};
00020     std::uint64_t ishmax{sizeof(std::uint64_t) * 8};
00021     while(true)
00022     {
00023         idiv = anss >> ish;
00024         anss ^= idiv;
00025         if(idiv <= 1 || ish == ishmax) return anss;
00026         ish <<= 1;
00027     }
00028 }
```

**4.7.2.18 hasAnalogReadout()**

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

Definition at line 74 of file [DIFUnpacker.cc](#).

```
00074 { return (DIFUnpacker::getLines(cb, idx) != 0); }
```

**4.7.2.19 hasLine()**

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

Definition at line 64 of file [DIFUnpacker.cc](#).

```
00064 { return ((cb[idx + DU::LINES_SHIFT] >> line) & 0x1); }
```

**4.7.2.20 hasTemperature()**

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

Definition at line 72 of file [DIFUnpacker.cc](#).

```
00072 { return (cb[idx] == DU::START_OF_DIF_TEMP); }
```

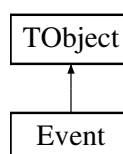
The documentation for this class was generated from the following files:

- [libs/core/include/DIFUnpacker.h](#)
- [libs/core/src/DIFUnpacker.cc](#)

**4.8 Event Class Reference**

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

Inheritance diagram for Event:



## Public Member Functions

- void [clear](#) ()
- void [addDIF](#) (const [DIF](#) &dif)

### 4.8.1 Detailed Description

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

### 4.8.2 Member Function Documentation

#### 4.8.2.1 addDIF()

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

Definition at line [10](#) of file [Event.cc](#).  
00010 { DIFs[dif.getID()] = dif; }

#### 4.8.2.2 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](#)

## 4.9 Hit Class Reference

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

Inheritance diagram for Hit:





## Public Member Functions

- 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](#) ()
- std::uint8\_t [getASICid](#) ()
- std::uint8\_t [getChannelId](#) ()
- std::uint8\_t [getThreshold](#) ()
- std::uint32\_t [getDTC](#) ()
- std::uint32\_t [getGTC](#) ()
- std::uint32\_t [getDIFBCID](#) ()
- std::uint32\_t [getFrameBCID](#) ()
- std::uint32\_t [getTimestamp](#) ()
- std::uint64\_t [getAbsoluteBCID](#) ()

### 4.9.1 Detailed Description

Definition at line 10 of file [Hit.h](#).

### 4.9.2 Member Function Documentation

#### 4.9.2.1 [getAbsoluteBCID\(\)](#)

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

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

```
00048 { return m_AbsoluteBCID; }
```

#### 4.9.2.2 [getASICid\(\)](#)

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

Definition at line 32 of file [Hit.cc](#).

```
00032 { return m_ASIC; }
```

#### 4.9.2.3 getChannelId()

```
std::uint8_t Hit::getChannelId ( )
```

Definition at line 34 of file [Hit.cc](#).

```
00034 { return m_Channel; }
```

#### 4.9.2.4 getDIFBCID()

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

Definition at line 42 of file [Hit.cc](#).

```
00042 { return m_DIFBCID; }
```

#### 4.9.2.5 getDIFid()

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

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

```
00030 { return m_DIF; }
```

#### 4.9.2.6 getDTC()

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

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

```
00038 { return m_DTC; }
```

#### 4.9.2.7 getFrameBCID()

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

Definition at line 44 of file [Hit.cc](#).

```
00044 { return m_FrameBCID; }
```

#### 4.9.2.8 getGTC()

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

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

```
00040 { return m_GTC; }
```

#### 4.9.2.9 getThreshold()

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

Definition at line 36 of file [Hit.cc](#).

```
00036 { return m_Threshold; }
```

#### 4.9.2.10 getTimestamp()

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

Definition at line 46 of file [Hit.cc](#).

```
00046 { return m_Timestamp; }
```

#### 4.9.2.11 setAbsoluteBCID()

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

Definition at line 28 of file [Hit.cc](#).

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

#### 4.9.2.12 setASIC()

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

Definition at line 12 of file [Hit.cc](#).

```
00012 { m_ASIC = asic; }
```

#### 4.9.2.13 setChannel()

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

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

```
00014 { m_Channel = channel; }
```

#### 4.9.2.14 setDIF()

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

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

```
00010 { m_DIF = dif; }
```

#### 4.9.2.15 setDIFBCID()

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

Definition at line 22 of file [Hit.cc](#).

```
00022 { m_DIFBCID = difbcid; }
```

#### 4.9.2.16 setDTC()

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

Definition at line 18 of file [Hit.cc](#).

```
00018 { m_DTC = dtc; }
```

#### 4.9.2.17 setFrameBCID()

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

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

```
00024 { m_FrameBCID = framebcid; }
```

#### 4.9.2.18 setGTC()

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

Definition at line 20 of file [Hit.cc](#).

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

**4.9.2.19 setThreshold()**

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

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

```
00016 { m_Threshold = threshold; }
```

**4.9.2.20 setTimestamp()**

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

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

```
00026 { 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**

template class should implement void SOURCE::start(); bool SOURCE::next(); void SOURCE::end(); const [Buffer&](#) SOURCE::getSDHCALBuffer();

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

Inheritance diagram for Interface:

**Public Member Functions**

- [Interface](#) ()
- virtual [~Interface](#) ()
- 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)

### 4.10.1 Detailed Description

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

void DESTINATION::begin(); void DESTINATION::processDIF(const DIFPtr&); void DESTINATION::processFrame(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();

Definition at line 26 of file [Interface.h](#).

### 4.10.2 Constructor & Destructor Documentation

#### 4.10.2.1 Interface()

```
Interface::Interface ( ) [inline]
```

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

```
00029 {}
```

#### 4.10.2.2 ~Interface()

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

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

```
00030 {}
```

### 4.10.3 Member Function Documentation

#### 4.10.3.1 endDIF()

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

Reimplemented in [ROOTWriter](#).

Definition at line 34 of file [Interface.h](#).

```
00034 {}
```

#### 4.10.3.2 endEvent()

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

Reimplemented in [ROOTWriter](#).

Definition at line 32 of file [Interface.h](#).

```
00032 {}
```

#### 4.10.3.3 endFrame()

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

Reimplemented in [ROOTWriter](#).

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

```
00036 {}
```

#### 4.10.3.4 endPad()

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

Reimplemented in [ROOTWriter](#).

Definition at line 38 of file [Interface.h](#).

```
00038 {}
```

#### 4.10.3.5 log()

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

Definition at line 39 of file [Interface.h](#).

```
00039 { return m_Logger; }
```

#### 4.10.3.6 setLogger()

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

Definition at line 40 of file [Interface.h](#).

```
00040 { m_Logger = logger; }
```

#### 4.10.3.7 startDIF()

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

Reimplemented in [ROOTWriter](#).

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

```
00033 {}
```

#### 4.10.3.8 startEvent()

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

Reimplemented in [ROOTWriter](#).

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

```
00031 {}
```

#### 4.10.3.9 startFrame()

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

Reimplemented in [ROOTWriter](#).

Definition at line 35 of file [Interface.h](#).

```
00035 {}
```

#### 4.10.3.10 startPad()

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

Reimplemented in [ROOTWriter](#).

Definition at line 37 of file [Interface.h](#).

```
00037 {}
```

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

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

## 4.11 RawBufferNavigator Class Reference

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



## Public Member Functions

- [RawBufferNavigator](#) ()=default
- [~RawBufferNavigator](#) ()=default
- [RawBufferNavigator](#) (const [Buffer](#) &b, const int &start=-1)
- void [setBuffer](#) (const [Buffer](#) &b, const int &start=-1)
- std::uint8\_t [getDetectorID](#) ()
- bool [validBuffer](#) ()
- std::uint32\_t [getStartOfDIF](#) ()
- unsigned char \* [getDIFBufferStart](#) ()
- std::uint32\_t [getDIFBufferSize](#) ()
- [Buffer](#) [getDIFBuffer](#) ()
- [DIFPtr](#) & [getDIFPtr](#) ()
- std::uint32\_t [getEndOfDIFData](#) ()
- std::uint32\_t [getSizeAfterDIFPtr](#) ()
- std::uint32\_t [getDIF\\_CRC](#) ()
- bool [hasSlowControlData](#) ()
- [Buffer](#) [getSCBuffer](#) ()
- bool [badSCData](#) ()
- [Buffer](#) [getEndOfAllData](#) ()

## Static Public Member Functions

- static void [StartAt](#) (const int &start)

### 4.11.1 Detailed Description

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

### 4.11.2 Constructor & Destructor Documentation

#### 4.11.2.1 RawBufferNavigator() [1/2]

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

#### 4.11.2.2 ~RawBufferNavigator()

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

#### 4.11.2.3 RawBufferNavigator() [2/2]

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

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

```
00016 : m_Buffer(b) { setBuffer(b, start); }
```

### 4.11.3 Member Function Documentation

#### 4.11.3.1 badSCData()

```
bool RawBufferNavigator::badSCData ( )
```

Definition at line 57 of file [RawBufferNavigator.cc](#).

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

#### 4.11.3.2 getDetectorID()

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

Definition at line 18 of file [RawBufferNavigator.cc](#).

```
00018 { return m_Buffer[0]; }
```

#### 4.11.3.3 getDIF\_CRC()

```
std::uint32_t RawBufferNavigator::getDIF_CRC ( )
```

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

```
00041 {
00042     uint32_t i{getEndOfDIFData()};
00043     uint32_t ret{0};
00044     ret |= (m_Buffer.begin()[i - 2]) << 8);
00045     ret |= m_Buffer.begin()[i - 1];
00046     return ret;
00047 }
```

#### 4.11.3.4 getDIFBuffer()

```
Buffer RawBufferNavigator::getDIFBuffer ( )
```

Definition at line 28 of file [RawBufferNavigator.cc](#).

```
00028 { return Buffer(getDIFBufferStart(), getDIFBufferSize()); }
```

#### 4.11.3.5 getDIFBufferSize()

```
std::uint32_t RawBufferNavigator::getDIFBufferSize ( )
```

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

```
00026 { return m_Buffer.size() - m_DIFstartIndex; }
```

#### 4.11.3.6 getDIFBufferStart()

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

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

```
00024 { return &(m_Buffer.begin())[m_DIFstartIndex]; }
```

#### 4.11.3.7 getDIFPtr()

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

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

```
00031 {
00032     m_TheDIFPtr.setBuffer(getDIFBufferStart(), getDIFBufferSize());
00033     return m_TheDIFPtr;
00034 }
```

#### 4.11.3.8 getEndOfAllData()

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

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

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

#### 4.11.3.9 getEndOfDIFData()

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

Definition at line 36 of file [RawBufferNavigator.cc](#).

```
00036 { return getDIFPtr().getGetFramePtrReturn() + 3; }
```

#### 4.11.3.10 getSCBuffer()

`Buffer RawBufferNavigator::getSCBuffer ( )`

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

```
00052 {
00053     setSCBuffer();
00054     return m_SCbuffer;
00055 }
```

#### 4.11.3.11 getSizeAfterDIFPtr()

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

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

```
00038 { return getDIFBufferSize() - getDIFPtr().getGetFramePtrReturn(); }
```

#### 4.11.3.12 getStartOfDIF()

`std::uint32_t RawBufferNavigator::getStartOfDIF ( )`

Definition at line 22 of file [RawBufferNavigator.cc](#).

```
00022 { return m_DIFstartIndex; }
```

#### 4.11.3.13 hasSlowControlData()

`bool RawBufferNavigator::hasSlowControlData ( )`

Definition at line 49 of file [RawBufferNavigator.cc](#).

```
00049 { return getDIFBufferStart()[getEndOfDIFData()] == 0xb1; }
```

#### 4.11.3.14 setBuffer()

`void RawBufferNavigator::setBuffer (`  
     `const Buffer & b,`  
     `const int & start = -1 ) [inline]`

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

```
00019 {
00020     m_BadSCdata = false;
00021     m_Buffer     = b;
00022     StartAt(start);
00023     m_DIFstartIndex = DIFUnpacker::getStartOfDIF(m_Buffer.begin(), m_Buffer.size(), m_Start);
00024 }
```

## 4.11.3.15 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 }
```

## 4.11.3.16 validBuffer()

```
bool RawBufferNavigator::validBuffer ( )
```

Definition at line 20 of file [RawBufferNavigator.cc](#).

```
00020 { return m_DIFstartIndex != 0; }
```

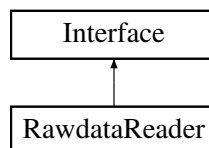
The documentation for this class was generated from the following files:

- [libs/core/include/RawBufferNavigator.h](#)
- [libs/core/src/RawBufferNavigator.cc](#)

## 4.12 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](#) & [getSDHCALBuffer](#) ()
- virtual [~RawdataReader](#) ()

## Static Public Member Functions

- static void [setDefaultBufferSize](#) (const std::size\_t &size)

### 4.12.1 Detailed Description

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

### 4.12.2 Constructor & Destructor Documentation

#### 4.12.2.1 RawdataReader()

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

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

```
00017 {
00018     m_buf.reserve(m_BufferSize);
00019     m_Filename = fileName;
00020 }
```

#### 4.12.2.2 ~RawdataReader()

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

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

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

### 4.12.3 Member Function Documentation

#### 4.12.3.1 closeFile()

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

Definition at line 42 of file [RawdataReader.cc](#).

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

#### 4.12.3.2 end()

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

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

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

#### 4.12.3.3 getFileSize()

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

Definition at line 126 of file [RawdataReader.cc](#).

```
00126 { return m_FileSize; }
```

#### 4.12.3.4 getSDHCALBuffer()

```
const Buffer & RawdataReader::getSDHCALBuffer ( )
```

Definition at line 118 of file [RawdataReader.cc](#).

```
00119 {  
00120     uncompress();  
00121     return m_Buffer;  
00122 }
```

#### 4.12.3.5 nextDIFbuffer()

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

Definition at line 91 of file [RawdataReader.cc](#).

```
00092 {  
00093     try  
00094     {  
00095         static int DIF_processed{0};  
00096         if(DIF_processed >= m_NumberOfDIF)  
00097         {  
00098             DIF_processed = 0;  
00099             return false;  
00100         }  
00101         else  
00102         {  
00103             DIF_processed++;  
00104             std::uint32_t bsize{0};  
00105             m_FileStream.read(reinterpret_cast<char*>(&bsize), sizeof(std::uint32_t));  
00106             m_FileStream.read(reinterpret_cast<char*>(&m_buf[0]), bsize);  
00107             m_Buffer = Buffer(m_buf);  
00108         }  
00109     }  
00110     catch(const std::ios_base::failure& e)  
00111     {  
00112         log()->error("Caught an ios_base::failure in openFile : {}", e.what());  
00113         return false;  
00114     }  
00115     return true;  
00116 }
```

#### 4.12.3.6 nextEvent()

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

Definition at line 76 of file [RawdataReader.cc](#).

```
00077 {
00078     try
00079     {
00080         m_FileStream.read(reinterpret_cast<char*>(&m_EventNumber), sizeof(std::uint32_t));
00081         m_FileStream.read(reinterpret_cast<char*>(&m_NumberOfDIF), sizeof(std::uint32_t));
00082     }
00083     catch(const std::ios_base::failure& e)
00084     {
00085         log()->error("Caught an ios_base::failure in openFile : {}", e.what());
00086         return false;
00087     }
00088     return true;
00089 }
```

#### 4.12.3.7 openFile()

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

Definition at line 55 of file [RawdataReader.cc](#).

```
00056 {
00057     try
00058     {
00059         m_FileStream.rdbuf()->pubsetbuf(0, 0);
00060         m_FileStream.exceptions(std::ifstream::failbit | std::ifstream::badbit);
00061         m_FileStream.open(fileName.c_str(), std::ios::in | std::ios::binary | std::ios::ate); // Start at
the end to directly calculate the size of the file then come back to beginning
00062         m_FileStream.rdbuf()->pubsetbuf(0, 0);
00063         if(m_FileStream.is_open())
00064         {
00065             setFileSize(m_FileStream.tellg());
00066             m_FileStream.seekg(0, std::ios::beg);
00067         }
00068     }
00069     catch(const std::ios_base::failure& e)
00070     {
00071         log()->error("Caught an ios_base::failure in openFile : {}", e.what());
00072         throw;
00073     }
00074 }
```

#### 4.12.3.8 setDefaultBufferSize()

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

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

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



#### 4.12.3.9 start()

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

Definition at line 22 of file [RawdataReader.cc](#).

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

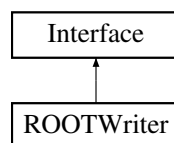
The documentation for this class was generated from the following files:

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

## 4.13 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](#) ()

#### 4.13.1 Detailed Description

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

## 4.13.2 Constructor & Destructor Documentation

### 4.13.2.1 ROOTWriter()

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

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

```
00010 {}
```

## 4.13.3 Member Function Documentation

### 4.13.3.1 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 }
```

### 4.13.3.2 endDIF()

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

Reimplemented from [Interface](#).

Definition at line 67 of file [ROOTWriter.cc](#).

```
00068 {  
00069     m_Event->addDIF(*m_DIF);  
00070     delete m_DIF;  
00071 }
```

### 4.13.3.3 endEvent()

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

Reimplemented from [Interface](#).

Definition at line 59 of file [ROOTWriter.cc](#).

```
00060 {  
00061     m_Tree->Fill();  
00062     if(m_Event) delete m_Event;  
00063 }
```

#### 4.13.3.4 endFrame()

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

Reimplemented from [Interface](#).

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

```
00076 {
00077     if(m_Hit->getThreshold() != 0) { m_DIF->addHit(*m_Hit); }
00078     delete m_Hit;
00079 }
```

#### 4.13.3.5 endPad()

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

Reimplemented from [Interface](#).

Definition at line 83 of file [ROOTWriter.cc](#).

```
00083 {}
```

#### 4.13.3.6 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 }
```

#### 4.13.3.7 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 }
```

#### 4.13.3.8 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 (static_cast<std::uint8_t>(channelIndex));
00054     m_Hit->setThreshold (static_cast<std::uint8_t>(d.getThresholdStatus (frameIndex, channelIndex)));
00055 }
```

#### 4.13.3.9 processSlowControl()

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

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

```
00029 { ; }
```

#### 4.13.3.10 setFilename()

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

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

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

#### 4.13.3.11 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::kLZMA, 9));
00015     m_Tree = new TTree("RawData", "Raw SDHCAL data tree");
00016     m_Tree->Branch("Events", &m_Event, 10, 0);
00017 }
```

#### 4.13.3.12 startDIF()

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

Reimplemented from [Interface](#).

Definition at line 65 of file [ROOTWriter.cc](#).

```
00065 { m_DIF = new DIF(); }
```

#### 4.13.3.13 startEvent()

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

Reimplemented from [Interface](#).

Definition at line 57 of file [ROOTWriter.cc](#).

```
00057 { m_Event = new Event(); }
```

#### 4.13.3.14 startFrame()

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

Reimplemented from [Interface](#).

Definition at line 73 of file [ROOTWriter.cc](#).

```
00073 { m_Hit = new Hit(); }
```

#### 4.13.3.15 startPad()

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

Reimplemented from [Interface](#).

Definition at line 81 of file [ROOTWriter.cc](#).

```
00081 {}
```

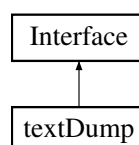
The documentation for this class was generated from the following files:

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

## 4.14 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)

### 4.14.1 Detailed Description

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

### 4.14.2 Constructor & Destructor Documentation

#### 4.14.2.1 textDump()

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

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

```
00018 {
00019     m_InternalLogger = std::make_shared<spdlog::logger>("textDump",
        std::make_shared<spdlog::sinks::stdout_color_sink_mt>());
00020     m_InternalLogger->set_level(spdlog::level::trace);
00021 }
```

### 4.14.3 Member Function Documentation

#### 4.14.3.1 end()

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

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

```
00025 { print()->info("textDump end of report"); }
```

#### 4.14.3.2 print()

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

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

```
00028 { return m_InternalLogger; }
```

#### 4.14.3.3 processDIF()

```
void textDump::processDIF (
    const DIFPtr & d )
```

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

```
00011 { print()->info("DIF_ID : {}, DTC : {}, GTC : {}, DIF BCID {}, Absolute BCID : {}, Nbr frames {}",
    d.getDIFid(), d.getDTC(), d.getGTC(), d.getBCID(), d.getAbsoluteBCID(), d.getNumberOfFrames()); }
```

#### 4.14.3.4 processFrame()

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

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

```
00014 {
00015     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));
00016 }
```

#### 4.14.3.5 processPadInFrame()

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

Definition at line 18 of file [textDump.cc](#).

```
00019 {
00020     if(d.getThresholdStatus(frameIndex, channelIndex) > 0) { print()->info("\t\tChannel {}, Threshold
    {}", channelIndex, d.getThresholdStatus(frameIndex, channelIndex)); }
00021 }
```

#### 4.14.3.6 processSlowControl()

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

Definition at line 23 of file [textDump.cc](#).

```
00023 { print()->error("textDump::processSlowControl not implemented yet."); }
```

#### 4.14.3.7 setLevel()

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

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

```
00029 { m_InternalLogger->set_level(level); }
```

#### 4.14.3.8 start()

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

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

```
00009 { 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](#)

## 4.15 Timer Class Reference

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

### Public Member Functions

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

#### 4.15.1 Detailed Description

Definition at line 10 of file [Timer.h](#).

#### 4.15.2 Member Function Documentation

##### 4.15.2.1 getElapsedTime()

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

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

```
00015 { return std::chrono::duration_cast<std::chrono::microseconds>(m_StopTime - m_StartTime).count(); }
```



#### 4.15.2.2 start()

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

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

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

#### 4.15.2.3 stop()

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

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

```
00014 { m_StopTime = std::chrono::high_resolution_clock::now(); }
```

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

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



## Chapter 5

# File Documentation

### 5.1 libs/core/include/Bits.h File Reference

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

#### Typedefs

- using [bit8\\_t](#) = std::uint8\_t
- using [bit16\\_t](#) = std::uint16\_t
- using [bit32\\_t](#) = std::uint32\_t
- using [bit64\\_t](#) = std::uint64\_t

#### Functions

- std::ostream & [operator<<](#) (std::ostream &os, const [bit8\\_t](#) &c)  
*Stream operator to print bit8\_t aka std::uint8\_t and not char or unsigned char.*

#### 5.1.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

#### 5.1.2 Typedef Documentation

#### 5.1.2.1 bit16\_t

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

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

#### 5.1.2.2 bit32\_t

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

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

#### 5.1.2.3 bit64\_t

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

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

#### 5.1.2.4 bit8\_t

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

Definition at line 10 of file [Bits.h](#).

### 5.1.3 Function Documentation

#### 5.1.3.1 operator<<()

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

Stream operator to print bit8\_t aka std::uint8\_t and not char or unsigned char.

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

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

## 5.2 Bits.h

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

```
00001
00005 #pragma once
00006
00007 #include <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 <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 <vector>
00012
00013 class Buffer
00014 {
00015 public:
00016     Buffer() : m_Buffer(nullptr), m_Size(0), m_Capacity(0) {}
00017     virtual ~Buffer() {}
00018     Buffer(const bit8_t b[], const std::size_t& i) : m_Buffer(const_cast<bit8_t*>(&b[0])), m_Size(i),
m_Capacity(i) {}
```

```

00019   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)) {}
00020   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)) {}
00021   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)) {}
00022
00023   std::size_t size()const { return m_Size; }
00024   std::size_t capacity()const { return m_Capacity; }
00025
00026   void set(unsigned char* b) { m_Buffer = b; }
00027   bit8_t* begin()const { return m_Buffer; }
00028   bit8_t* end()const { return m_Buffer + m_Size; }
00029   bit8_t& operator[](const std::size_t& pos) { return m_Buffer[pos]; }
00030   bit8_t& operator[](const std::size_t& pos)const { return m_Buffer[pos]; }
00031
00032   void setSize(const std::size_t& size) { m_Size = size; }
00033
00034 private:
00035   bit8_t* m_Buffer{nullptr};
00036   std::size_t m_Size{0};
00037   std::size_t m_Capacity{0};
00038 };

```

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

```

#include "Buffer.h"
#include "BufferLooperCounter.h"
#include "DetectorId.h"
#include "Formatters.h"
#include "RawBufferNavigator.h"
#include "Timer.h"
#include "Words.h"
#include <algorithm>
#include <cassert>
#include <memory>
#include <spdlog/sinks/null_sink.h>
#include <spdlog/spdlog.h>
#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 "Buffer.h"
00008 #include "BufferLooperCounter.h"
00009 #include "DetectorId.h"
00010 #include "Formatters.h"
00011 #include "RawBufferNavigator.h"
00012 #include "Timer.h"
00013 #include "Words.h"
00014
00015 #include <algorithm>
00016 #include <cassert>
00017 #include <memory>
00018 #include <spdlog/sinks/null_sink.h>
00019 #include <spdlog/spdlog.h>
00020 #include <vector>
00021 // function to loop on buffers
00022
00023 template<typename SOURCE, typename DESTINATION> class BufferLooper
00024 {
00025 public:
00026     BufferLooper(SOURCE& source, DESTINATION& dest, bool debug = false) : m_Source(source),
00027     m_Destination(dest), m_Debug(debug)
00028     {
00029         m_Logger = spdlog::create<spdlog::sinks::null_sink_mt>("streamout");
00030         if(!spdlog::get("streamout")) { spdlog::register_logger(m_Logger); }
00031         m_Source.setLogger(m_Logger);
00032         m_Destination.setLogger(m_Logger);
00033     }
00034     void addSink(const spdlog::sink_ptr& sink, const spdlog::level::level_enum& level =
00035     spdlog::get_level())
00036     {
00037         sink->set_level(level);
00038         m_Sinks.push_back(sink);
00039         m_Logger = std::make_shared<spdlog::logger>("streamout", begin(m_Sinks), end(m_Sinks));
00040         m_Source.setLogger(m_Logger);
00041         m_Destination.setLogger(m_Logger);
00042     }
00043     void loop(const std::uint32_t& m_NbrEventsToProcess = 0)
00044     {
00045         Timer timer;
00046         timer.start();
00047         m_Source.start();
00048         m_Destination.start();
00049         RawBufferNavigator bufferNavigator;
00050         while(m_Source.nextEvent() && m_NbrEventsToProcess >= m_NbrEvents)
00051         {
00052             m_Source.startEvent();
00053             m_Destination.startEvent();
00054
00055             m_Logger->warn("====* Event number {} *====", m_NbrEvents);
00056             while(m_Source.nextDIFbuffer())
00057             {
00058                 const Buffer& buffer = m_Source.getSDHCALBuffer();
00059                 bufferNavigator.setBuffer(buffer);
00060
00061                 bit8_t* debug_variable_1 = buffer.end();
00062                 bit8_t* debug_variable_2 = bufferNavigator.getDIFBuffer().end();
00063                 if(debug_variable_1 != debug_variable_2) m_Logger->info("DIF BUFFER END {} {}",
00064                 fmt::ptr(debug_variable_1), fmt::ptr(debug_variable_2));
00065                 if(m_Debug) assert(debug_variable_1 == debug_variable_2);
00066
00067                 if(std::find(m_DetectorIDs.begin(), m_DetectorIDs.end(),
00068                 static_cast<DetectorID>(bufferNavigator.getDetectorID())) == m_DetectorIDs.end())
00069                 {
00070                     m_Logger->trace("{} ", bufferNavigator.getDetectorID());
00071                     continue;
00072                 }
00073
00074                 m_Source.startDIF();
00075                 m_Destination.startDIF();
00076
00077                 uint32_t idstart = bufferNavigator.getStartOfDIF();
00078                 if(m_Debug && idstart == 0) m_Logger->info(to_hex(buffer));
00079                 c.DIFStarter[idstart]++;
00080                 if(!bufferNavigator.validBuffer())
00081                 {
00082                     m_Logger->error("!bufferNavigator.validBuffer()");
00083                     continue;
00084                 }
00085             }
00086         }
00087     }

```

```

00086     }
00087     DIFPtr& d = bufferNavigator.getDIFPtr();
00088     c.DIFPtrValueAtReturnedPos[bufferNavigator.getDIFBufferStart()[d.getGetFramePtrReturn()]]++;
00089     if(m_Debug) assert(bufferNavigator.getDIFBufferStart()[d.getGetFramePtrReturn()] == 0xa0);
00090     c.SizeAfterDIFPtr[bufferNavigator.getSizeAfterDIFPtr()]++;
00091     m_Destination.processDIF(d);
00092     for(std::size_t i = 0; i < d.getNumberOfFrames(); ++i)
00093     {
00094         m_Source.startFrame();
00095         m_Destination.startFrame();
00096         m_Destination.processFrame(d, i);
00097         for(std::size_t j = 0; j < DU::NUMBER_PAD; ++j)
00098         {
00099             m_Source.startPad();
00100             m_Destination.startPad();
00101             m_Destination.processPadInFrame(d, i, j);
00102             m_Source.endPad();
00103             m_Destination.endPad();
00104         }
00105         m_Source.endFrame();
00106         m_Destination.endFrame();
00107     }
00108
00109     bool processSC = false;
00110     if(bufferNavigator.hasSlowControlData())
00111     {
00112         c.hasSlowControl++;
00113         processSC = true;
00114     }
00115     if(bufferNavigator.badSCData())
00116     {
00117         c.hasBadSlowControl++;
00118         processSC = false;
00119     }
00120     if(processSC) { m_Destination.processSlowControl(bufferNavigator.getSCBuffer()); }
00121
00122     Buffer eod = bufferNavigator.getEndOfAllData();
00123     c.SizeAfterAllData[eod.size()]++;
00124     bit8_t* debug_variable_3 = eod.end();
00125     if(debug_variable_1 != debug_variable_3) m_Logger->info("END DATA BUFFER END {} {}",
00126         fmt::ptr(debug_variable_1), fmt::ptr(debug_variable_3));
00127     if(m_Debug) assert(debug_variable_1 == debug_variable_3);
00128     if(eod.size() != 0) m_Logger->info("End of Data remaining stuff : {}", to_hex(eod));
00129
00130     int nonzeroCount = 0;
00131     for(bit8_t it = eod.begin(); it != eod.end(); it++)
00132         if(static_cast<int>(*it) != 0) nonzeroCount++;
00133     c.NonZeroValueAtEndOfData[nonzeroCount]++;
00134     m_Source.endDIF();
00135     m_Destination.endDIF();
00136     } // end of DIF while loop
00137     m_Logger->warn("***** Event number {} *****", m_NbrEvents);
00138     m_NbrEvents++;
00139     m_Source.endEvent();
00140     m_Destination.endEvent();
00141     } // end of event while loop
00142     m_Destination.end();
00143     m_Source.end();
00144     timer.stop();
00145     fmt::print("=== elapsed time {}ms ({}ms/event) ===\n", timer.getElapsedTime() / 1000,
00146         timer.getElapsedTime() / (1000 * m_NbrEvents));
00147 }
00148 void printAllCounters() { c.printAllCounters(); }
00149 std::shared_ptr<spdlog::logger> log() { return m_Logger; }
00150
00151 void setDetectorIDs(const std::vector<DetectorID>& detectorIDs) { m_DetectorIDs = detectorIDs; }
00152
00153 private:
00154     std::vector<DetectorID> m_DetectorIDs;
00155     std::shared_ptr<spdlog::logger> m_Logger{nullptr};
00156     std::vector<spdlog::sink_ptr> m_Sinks;
00157     BufferLooperCounter c;
00158     SOURCE& m_Source{nullptr};
00159     DESTINATION& m_Destination{nullptr};
00160     bool m_Debug{false};
00161     std::uint32_t m_NbrEvents{1};
00162 };

```

## 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 <stdint>
```

## Enumerations

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

### 5.9.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

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

## 5.9.2 Enumeration Type Documentation

### 5.9.2.1 DetectorID

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

#### Enumerator

HARDROC	
HARDROC_NEW	
RUNHEADER	

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

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

## 5.10 DetectorId.h

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

```
00001
00005 #pragma once
00006
00007 #include <stdint>
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 "DIFUnpacker.h"
#include <stdint>
#include <spdlog/spdlog.h>
#include <string>
#include <vector>
```

### Classes

- class [DIFPtr](#)

### 5.11.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

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

## 5.12 DIFPtr.h

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

```

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

```

```

00069     }
00070 }
00071
00072 inline unsigned char*      DIFPtr::getPtr()const { return theDIF_; }
00073 inline std::uint32_t      DIFPtr::getGetFramePtrReturn()const { return
    theGetFramePtrReturn_; }
00074 inline std::vector<unsigned char*>& DIFPtr::getFramesVector() { return theFrames_; }
00075 inline std::vector<unsigned char*>& DIFPtr::getLinesVector() { return theLines_; }
00076 inline std::uint32_t      DIFPtr::getID()const { return DIFUnpacker::getID(theDIF_); }
00077 inline std::uint32_t      DIFPtr::getDTC()const { return DIFUnpacker::getDTC(theDIF_); }
00078 inline std::uint32_t      DIFPtr::getGTC()const { return DIFUnpacker::getGTC(theDIF_); }
00079 inline std::uint64_t      DIFPtr::getAbsoluteBCID()const { return
    DIFUnpacker::getAbsoluteBCID(theDIF_); }
00080 inline std::uint32_t      DIFPtr::getBCID()const { return DIFUnpacker::getBCID(theDIF_); }
00081 inline std::uint32_t      DIFPtr::getLines()const { return DIFUnpacker::getLines(theDIF_); }
00082 inline bool               DIFPtr::hasLine(const std::uint32_t& line)const { return
    DIFUnpacker::hasLine(line, theDIF_); }
00083 inline std::uint32_t      DIFPtr::getTASU1()const { return DIFUnpacker::getTASU1(theDIF_); }
00084 inline std::uint32_t      DIFPtr::getTASU2()const { return DIFUnpacker::getTASU2(theDIF_); }
00085 inline std::uint32_t      DIFPtr::getTDIF()const { return DIFUnpacker::getTDIF(theDIF_); }
00086 inline float              DIFPtr::getTemperatureDIF()const { return 0.508 * getTDIF() -
    9.659; }
00087 inline float              DIFPtr::getTemperatureASU1()const { return (getTASU1() >> 3) *
    0.0625; }
00088 inline float              DIFPtr::getTemperatureASU2()const { return (getTASU2() >> 3) *
    0.0625; }
00089 inline bool               DIFPtr::hasTemperature()const { return
    DIFUnpacker::hasTemperature(theDIF_); }
00090 inline bool               DIFPtr::hasAnalogReadout()const { return
    DIFUnpacker::hasAnalogReadout(theDIF_); }
00091 inline std::uint32_t      DIFPtr::getNumberOfFrames()const { return theFrames_.size(); }
00092 inline unsigned char*     DIFPtr::getFramePtr(const std::uint32_t& i)const { return
    theFrames_[i]; }
00093 inline std::uint32_t      DIFPtr::getFrameAsicHeader(const std::uint32_t& i)const { return
    DIFUnpacker::getFrameAsicHeader(theFrames_[i]); }
00094 inline std::uint32_t      DIFPtr::getFrameBCID(const std::uint32_t& i)const { return
    DIFUnpacker::getFrameBCID(theFrames_[i]); }
00095 inline std::uint32_t      DIFPtr::getFrameTimeToTrigger(const std::uint32_t& i)const {
    return getBCID() - getFrameBCID(i); }
00096 inline bool               DIFPtr::getFrameLevel(const std::uint32_t& i, const std::uint32_t&
    ipad, const std::uint32_t& ilevel)const { return DIFUnpacker::getFrameLevel(theFrames_[i], ipad,
    ilevel); }
00097 // Addition by GG
00098 inline uint32_t           DIFPtr::getDIFid()const { return getID() & 0xFF; }
00099 inline uint32_t           DIFPtr::getASICid(const std::uint32_t& i)const { return
    getFrameAsicHeader(i) & 0xFF; }
00100 inline uint32_t           DIFPtr::getThresholdStatus(const std::uint32_t& i, const
    std::uint32_t& ipad)const { return (((uint32_t)getFrameLevel(i, ipad, 1)) << 1) |
    ((uint32_t)getFrameLevel(i, ipad, 0)); }

```

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

```

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

```

### Classes

- class [DIFSlowControl](#)  
Handler of *DIF* Slow Control info.

### 5.13.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

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

## 5.14 DIFSlowControl.h

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

```

00001
00005 #pragma once
00006
00007 #include <bitset>
00008 #include <cstdint>
00009 #include <map>
00010 #include <string>
00019 class DIFSlowControl
00020 {
00021 public:
00023
00028     DIFSlowControl(const std::uint8_t& version, const std::uint8_t& DIFid, unsigned char* buf);
00029
00031     inline std::uint8_t getDIFid();
00032
00034
00037     inline std::map<int, std::map<std::string, int>> getChipsMap();
00038
00040
00044     inline std::map<std::string, int> getChipSlowControl(const int& asicid);
00045
00047
00051     inline int getChipSlowControl(const std::int8_t& asicid, const std::string& param);
00052
00054     void Dump();
00055
00056 private:
00058     DIFSlowControl() = delete;
00060     void FillHR1(const int& header_shift, unsigned char* cbuf);
00062     void FillHR2(const int& header_shift, unsigned char* cbuf);
00064     void FillAsicHR1(const std::bitset<72 * 8>& bs);
00066     void FillAsicHR2(const std::bitset<109 * 8>& bs);
00067
00068     unsigned int                m_DIFid{0};
00069     unsigned int                m_Version{0};
00070     unsigned int                m_AsicType{0}; // asicType_
00071     unsigned int                m_NbrAsic{0};
00072     std::map<int, std::map<std::string, int>> m_MapSC;
00073 };

```

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

```

#include <cstdint>
#include <vector>

```

### Classes

- class [DIFUnpacker](#)

### 5.15.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

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

## 5.16 DIFUnpacker.h

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

```

00001
00005 #pragma once
00006
00007 #include <cstdint>
00008 #include <vector>
00009
00010 class DIFUnpacker
00011 {
00012 public:
00013     static std::uint64_t GrayToBin(const std::uint64_t& n);
00014     static std::uint32_t getStartOfDIF(const unsigned char* cbuf, const std::uint32_t& size_buf, const
std::uint32_t& start = 92);
00015     static std::uint32_t getID(const unsigned char* cb, const std::uint32_t& idx = 0);
00016     static std::uint32_t getDTC(const unsigned char* cb, const std::uint32_t& idx = 0);
00017     static std::uint32_t getGTC(const unsigned char* cb, const std::uint32_t& idx = 0);
00018     static std::uint64_t getAbsoluteBCID(const unsigned char* cb, const std::uint32_t& idx = 0);
00019     static std::uint32_t getBCID(const unsigned char* cb, const std::uint32_t& idx = 0);
00020     static std::uint32_t getLines(const unsigned char* cb, const std::uint32_t& idx = 0);
00021     static bool
std::uint32_t& idx = 0);
00022     static std::uint32_t getTASU1(const unsigned char* cb, const std::uint32_t& idx = 0);
00023     static std::uint32_t getTASU2(const unsigned char* cb, const std::uint32_t& idx = 0);
00024     static std::uint32_t getTDIF(const unsigned char* cb, const std::uint32_t& idx = 0);
00025     static bool
hasTemperature(const unsigned char* cb, const std::uint32_t& idx = 0);
00026     static bool
hasAnalogReadout(const unsigned char* cb, const std::uint32_t& idx = 0);
00027
00028     static std::uint32_t getFrameAsicHeader(const unsigned char* framePtr);
00029     static std::uint32_t getFrameBCID(const unsigned char* framePtr);
00030
00031     static bool getFramePAD(const unsigned char* framePtr, const std::uint32_t& ip);
00032     static bool getFrameLevel(const unsigned char* framePtr, const std::uint32_t& ip, const
std::uint32_t& level);
00033
00034     static std::uint32_t getAnalogPtr(std::vector<unsigned char*>& vLines, unsigned char* cb, const
std::uint32_t& idx = 0);
00035     static std::uint32_t getFramePtr(std::vector<unsigned char*>& vFrame, std::vector<unsigned char*>&
vLines, const std::uint32_t& max_size, unsigned char* cb, const std::uint32_t& idx = 0);
00036 };

```

## 5.17 libs/core/include/Formatters.h File Reference

```

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

```

### Functions

- std::string [to\\_dec](#) (const [Buffer](#) &b, const std::size\_t &begin=0, const std::size\_t &end=-1)
- std::string [to\\_dec](#) (const [bit8\\_t](#) &)
- std::string [to\\_dec](#) (const [bit16\\_t](#) &)
- std::string [to\\_dec](#) (const [bit32\\_t](#) &)
- std::string [to\\_dec](#) (const [bit64\\_t](#) &)
- std::string [to\\_hex](#) (const [Buffer](#) &b, const std::size\_t &begin=0, const std::size\_t &end=-1)
- std::string [to\\_hex](#) (const [bit8\\_t](#) &)
- std::string [to\\_hex](#) (const [bit16\\_t](#) &)
- std::string [to\\_hex](#) (const [bit32\\_t](#) &)
- std::string [to\\_hex](#) (const [bit64\\_t](#) &)
- std::string [to\\_bin](#) (const [Buffer](#) &b, const std::size\_t &begin=0, const std::size\_t &end=-1)
- std::string [to\\_bin](#) (const [bit8\\_t](#) &)
- std::string [to\\_bin](#) (const [bit16\\_t](#) &)
- std::string [to\\_bin](#) (const [bit32\\_t](#) &)

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

### 5.17.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

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

### 5.17.2 Function Documentation

#### 5.17.2.1 to\_bin() [1/5]

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

Definition at line 71 of file [Formatters.cc](#).

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

#### 5.17.2.2 to\_bin() [2/5]

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

Definition at line 73 of file [Formatters.cc](#).

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

#### 5.17.2.3 to\_bin() [3/5]

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

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

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

#### 5.17.2.4 to\_bin() [4/5]

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

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

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

#### 5.17.2.5 to\_bin() [5/5]

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

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

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

#### 5.17.2.6 to\_dec() [1/5]

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

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

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

#### 5.17.2.7 to\_dec() [2/5]

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

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

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



### 5.17.2.8 to\_dec() [3/5]

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

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

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

### 5.17.2.9 to\_dec() [4/5]

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

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

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

### 5.17.2.10 to\_dec() [5/5]

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

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

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

### 5.17.2.11 to\_hex() [1/5]

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

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

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

#### 5.17.2.12 to\_hex() [2/5]

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

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

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

#### 5.17.2.13 to\_hex() [3/5]

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

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

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

#### 5.17.2.14 to\_hex() [4/5]

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

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

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

#### 5.17.2.15 to\_hex() [5/5]

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

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

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

#### 5.17.2.16 to\_oct() [1/5]

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

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

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

#### 5.17.2.17 to\_oct() [2/5]

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

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

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

#### 5.17.2.18 to\_oct() [3/5]

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

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

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

#### 5.17.2.19 to\_oct() [4/5]

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

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

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

#### 5.17.2.20 to\_oct() [5/5]

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

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

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

## 5.18 Formatters.h

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

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

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

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

### Classes

- class [Interface](#)

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

### 5.19.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

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

## 5.20 Interface.h

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

```

00001
00004 #pragma once
00005
00006 #include "Buffer.h"
00007
00008 #include <memory>
00009 #include <spdlog/logger.h>
00010
00026 class Interface
00027 {
00028 public:
00029     Interface() {}
00030     virtual ~Interface() {}
00031     virtual void                startEvent() {}
00032     virtual void                endEvent() {}
00033     virtual void                startDIF() {}
00034     virtual void                endDIF() {}
00035     virtual void                startFrame() {}
00036     virtual void                endFrame() {}
00037     virtual void                startPad() {}
00038     virtual void                endPad() {}
00039     std::shared_ptr<spdlog::logger>& log() { return m_Logger; }
00040     void                setLogger(const std::shared_ptr<spdlog::logger>& logger) { m_Logger
= logger; }
00041
00042 private:
00043     std::shared_ptr<spdlog::logger> m_Logger{nullptr};
00044 };

```

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

```

#include "Buffer.h"
#include "DIFPtr.h"
#include "DIFUnpacker.h"

```

### Classes

- class [RawBufferNavigator](#)

### 5.21.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

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

## 5.22 RawBufferNavigator.h

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

```

00001
00005 #pragma once
00006
00007 #include "Buffer.h"
00008 #include "DIFPtr.h"
00009 #include "DIFUnpacker.h"
00010
00011 // class to navigate in the raw data buffer
00012 class RawBufferNavigator
00013 {
00014 public:
00015     RawBufferNavigator() = default;
00016     ~RawBufferNavigator() = default;
00017     explicit RawBufferNavigator(const Buffer& b, const int& start = -1);
00018     void setBuffer(const Buffer& b, const int& start = -1)
00019     {
00020         m_BadSCdata = false;
00021         m_Buffer = b;
00022         StartAt(start);
00023         m_DIFstartIndex = DIFUnpacker::getStartOfDIF(m_Buffer.begin(), m_Buffer.size(), m_Start);
00024     }
00025     std::uint8_t    getDetectorID();
00026     bool            validBuffer();
00027     std::uint32_t    getStartOfDIF();
00028     unsigned char*   getDIFBufferStart();
00029     std::uint32_t    getDIFBufferSize();
00030     Buffer           getDIFBuffer();
00031     DIFPtr&          getDIFPtr();
00032     std::uint32_t    getEndOfDIFData();
00033     std::uint32_t    getSizeAfterDIFPtr();
00034     std::uint32_t    getDIF_CRC();
00035     bool            hasSlowControlData();
00036     Buffer           getSCBuffer();
00037     bool            badSCData();
00038     Buffer           getEndOfAllData();
00039     static void      StartAt(const int& start);
00040
00041 private:
00042     void            setSCBuffer();
00043     Buffer           m_Buffer;
00044     Buffer           m_SCbuffer;
00045     std::uint32_t    m_DIFstartIndex{0};
00046     DIFPtr          m_TheDIFPtr;
00047     bool            m_BadSCdata{false};
00048     static int       m_Start;
00049 };

```

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

```
#include <chrono>
```

### Classes

- class [Timer](#)

### 5.23.1 Detailed Description

#### Copyright

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

#### See also

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

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

## 5.24 Timer.h

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

```
00001
00006 #pragma once
00007
00008 #include <chrono>
00009
00010 class Timer
00011 {
00012 public:
00013     void start() { m_StartTime = std::chrono::high_resolution_clock::now(); }
00014     void stop() { m_StopTime = std::chrono::high_resolution_clock::now(); }
00015     float getElapsedTime() { return std::chrono::duration_cast<std::chrono::microseconds>(m_StopTime -
m_StartTime).count(); }
00016
00017 private:
00018     std::chrono::time_point<std::chrono::high_resolution_clock> m_StartTime;
00019     std::chrono::time_point<std::chrono::high_resolution_clock> m_StopTime;
00020 };
```

## 5.25 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 }

### 5.25.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

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

### 5.25.2 Enumeration Type Documentation

#### 5.25.2.1 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 };

```

## 5.26 Words.h

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

```

00001
00005 #pragma once
00006
00007 #include <stdint>
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 };

```

## 5.27 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.*

#### 5.27.1 Detailed Description

##### Copyright

2022 G.Grenier F.Lagarde

Definition in file [Bits.cc](#).

#### 5.27.2 Function Documentation

##### 5.27.2.1 `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; }
```

## 5.28 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; }
```

## 5.29 libs/core/src/Buffer.cc File Reference

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

## 5.30 Buffer.cc

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

```
00001
00006 #include "Buffer.h"
```

## 5.31 libs/core/src/BufferLooperCounter.cc File Reference

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

## 5.32 BufferLooperCounter.cc

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

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

## 5.33 libs/core/src/DIFSlowControl.cc File Reference

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

### 5.33.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

Definition in file [DIFSlowControl.cc](#).

## 5.34 DIFSlowControl.cc

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

```
00001
00005 #include "DIFSlowControl.h"
00006
00007 #include <stdint>
00008 #include <iostream>
00009
00010 DIFSlowControl::DIFSlowControl(const std::uint8_t& version, const std::uint8_t& DIFId, unsigned char*
    cbuf) : m_Version(version), m_DIFId(DIFId), m_AsicType(2)
00011 {
00012     if(cbuf[0] != 0xb1) return;
00013     int header_shift{6};
00014     if(m_Version < 8) m_NbrAsic = cbuf[5];
00015     else
00016     {
00017         m_DIFId = cbuf[1];
00018         m_NbrAsic = cbuf[2];
00019         header_shift = 3;
00020     }
00021     int size_hardroc1 = m_NbrAsic * 72 + header_shift + 1;
00022     if(cbuf[size_hardroc1 - 1] != 0xal) size_hardroc1 = 0;
00023
00024     int size_hardroc2 = m_NbrAsic * 109 + header_shift + 1;
00025     if(cbuf[size_hardroc2 - 1] != 0xal) size_hardroc2 = 0;
00026     if(size_hardroc1 != 0)
00027     {
00028         FillHR1(header_shift, cbuf);
00029         m_AsicType = 1;
00030     }
00031     else if(size_hardroc2 != 0)
00032         FillHR2(header_shift, cbuf);
00033     else
00034         return;
00035 }
00036
00037 inline std::uint8_t DIFSlowControl::getDIFId() { return m_DIFId; }
00038
00039 inline std::map<int, std::map<std::string, int> DIFSlowControl::getChipsMap() { return m_MapSC; }
00040
00041 inline std::map<std::string, int> DIFSlowControl::getChipSlowControl(const int& asicid) { return
    m_MapSC[asicid]; }
00042
00043 inline int DIFSlowControl::getChipSlowControl(const std::int8_t& asicid, const std::string& param) {
    return getChipSlowControl(asicid)[param]; }
00044
00045 void DIFSlowControl::Dump()
00046 {
00047     for(std::map<int, std::map<std::string, int>::iterator it = m_MapSC.begin(); it != m_MapSC.end();
        it++)
00048     {
00049         std::cout << "ASIC " << it->first << std::endl;
00050         for(std::map<std::string, int>::iterator jt = (it->second).begin(); jt != (it->second).end();
            jt++) std::cout << jt->first << " : " << jt->second << std::endl;
```

```

00051     }
00052 }
00053
00054 void DIFSlowControl::FillHR1(const int& header_shift, unsigned char* cbuf)
00055 {
00056     int nasic{cbuf[header_shift - 1]};
00057     int idx{header_shift};
00058     for(int k = 0; k < nasic; k++)
00059     {
00060         std::bitset<72 * 8> bs;
00061         // printf("%x %x \n",cbuf[idx+k*72+69],cbuf[idx+k*72+70]);
00062         for(int l = 71; l >= 0; l--)
00063         {
00064             // printf("%d %x : %d -->",l,cbuf[idx+k*72+l], (71-l)*8);
00065             for(int m = 0; m < 8; m++)
00066             {
00067                 if(((1 < m) & cbuf[idx + k * 72 + l]) != 0) bs.set((71 - l) * 8 + m, 1);
00068                 else
00069                     bs.set((71 - l) * 8 + m, 0);
00070                 // printf("%d", (int) bs[(71-l)*8+m]);
00071             }
00072             // printf("\n");
00073         }
00074         FillAsicHR1(bs);
00075     }
00076 }
00077
00078 void DIFSlowControl::FillHR2(const int& header_shift, unsigned char* cbuf)
00079 {
00080     // int scsizer=cbuf[header_shift-1]*109+(header_shift-1)+2;
00081     int nasic{cbuf[header_shift - 1]};
00082     int idx{header_shift};
00083     // std::cout<<" DIFSlowControl::FillHR nasic "<nasic<<std::endl;
00084     for(int k = 0; k < nasic; k++)
00085     {
00086         std::bitset<109 * 8> bs;
00087         // printf("%x %x \n",cbuf[idx+k*109+69],cbuf[idx+k*109+70]);
00088         for(int l = 108; l >= 0; l--)
00089         {
00090             // printf("%d %x : %d -->",l,cbuf[idx+k*109+l], (71-l)*8);
00091             for(int m = 0; m < 8; m++)
00092             {
00093                 if(((1 < m) & cbuf[idx + k * 109 + l]) != 0) bs.set((108 - l) * 8 + m, 1);
00094                 else
00095                     bs.set((108 - l) * 8 + m, 0);
00096                 // printf("%d", (int) bs[(71-l)*8+m]);
00097             }
00098             // printf("\n");
00099         }
00100         FillAsicHR2(bs);
00101     }
00102 }
00103
00104 void DIFSlowControl::FillAsicHR1(const std::bitset<72 * 8>& bs)
00105 {
00106     // Asic Id
00107     int asicid{0};
00108     for(int j = 0; j < 8; j++)
00109         if(bs[j + 9] != 0) asicid += (1 < (7 - j));
00110     std::map<std::string, int> mAsic;
00111     // Slow Control
00112     mAsic["SSC0"] = static_cast<int>(bs[575]);
00113     mAsic["SSC1"] = static_cast<int>(bs[574]);
00114     mAsic["SSC2"] = static_cast<int>(bs[573]);
00115     mAsic["Choix_caisson"] = static_cast<int>(bs[572]);
00116     mAsic["SW_50k"] = static_cast<int>(bs[571]);
00117     mAsic["SW_100k"] = static_cast<int>(bs[570]);
00118     mAsic["SW_100f"] = static_cast<int>(bs[569]);
00119     mAsic["SW_50f"] = static_cast<int>(bs[568]);
00120
00121     mAsic["Valid_DC"] = static_cast<int>(bs[567]);
00122     mAsic["ON_Discri"] = static_cast<int>(bs[566]);
00123     mAsic["ON_Fsb"] = static_cast<int>(bs[565]);
00124     mAsic["ON_Otaq"] = static_cast<int>(bs[564]);
00125     mAsic["ON_W"] = static_cast<int>(bs[563]);
00126     mAsic["ON_Ss"] = static_cast<int>(bs[562]);
00127     mAsic["ON_Buf"] = static_cast<int>(bs[561]);
00128     mAsic["ON_Paf"] = static_cast<int>(bs[560]);
00129     // Gain
00130     for(int i = 0; i < 64; i++)
00131     {
00132         int gain{0};
00133         for(int j = 0; j < 6; j++)
00134             if(bs[176 + i * 6 + j] != 0) gain += (1 < j);
00135         mAsic["Channel_" + std::to_string(i) + "_" + "Gain"] = gain;
00136         mAsic["Channel_" + std::to_string(i) + "_" + "cTest"] = bs[112 + i];
00137         mAsic["Channel_" + std::to_string(i) + "_" + "Valid_trig"] = static_cast<int>(bs[25 + i]);

```

```

00138     }
00139
00140     mAsic["ON_Otabg"] = static_cast<int>(bs[111]);
00141     mAsic["ON_Dac"] = static_cast<int>(bs[110]);
00142     mAsic["ON_Otadac"] = static_cast<int>(bs[109]);
00143     // DAC
00144     int dac1{0};
00145     for(int j = 0; j < 10; j++)
00146         if(bs[j + 99] != 0) dac1 += (1 << j);
00147     mAsic["DAC1"] = dac1;
00148     int dac0{0};
00149     for(int j = 0; j < 10; j++)
00150         if(bs[j + 89] != 0) dac0 += (1 << j);
00151     mAsic["DAC0"] = dac0;
00152     mAsic["EN_Raz_Ext"] = static_cast<int>(bs[23]);
00153     mAsic["EN_Raz_Int"] = static_cast<int>(bs[22]);
00154     mAsic["EN_Out_Raz_Int"] = static_cast<int>(bs[21]);
00155     mAsic["EN_Trig_Ext"] = static_cast<int>(bs[20]);
00156     mAsic["EN_Trig_Int"] = static_cast<int>(bs[19]);
00157     mAsic["EN_Out_Trig_Int"] = static_cast<int>(bs[18]);
00158     mAsic["Bypass_Chip"] = static_cast<int>(bs[17]);
00159     mAsic["HardrocHeader"] = static_cast<int>(asicid);
00160     mAsic["EN_Out_Discr"] = static_cast<int>(bs[8]);
00161     mAsic["EN_Transmit_On"] = static_cast<int>(bs[7]);
00162     mAsic["EN_Dout"] = static_cast<int>(bs[6]);
00163     mAsic["EN_RamFull"] = static_cast<int>(bs[5]);
00164     m_MapSC[asicid] = mAsic;
00165 }
00166
00167 void DIFSlowControl::FillAsicHR2(const std::bitset<109 * 8>& bs)
00168 {
00169     int asicid{0};
00170     for(int j = 0; j < 8; j++)
00171         if(bs[j + (108 - 7) * 8 + 2] != 0) asicid += (1 << (7 - j));
00172     std::map<std::string, int> mAsic;
00173     for(int i = 0; i < 64; i++)
00174     {
00175         int gain{0};
00176         int mask{0};
00177         mAsic["Channel_" + std::to_string(i) + "_" + "cTest"] = bs[i];
00178         for(int j = 0; j < 8; j++)
00179             if(bs[64 + i * 8 + j] != 0) gain += (1 << j);
00180         mAsic["Channel_" + std::to_string(i) + "_" + "Gain"] = gain;
00181         for(int j = 0; j < 3; j++)
00182             if(bs[8 * 77 + 2 + i * 3 + j] != 0) mask += (1 << j);
00183         mAsic["Channel_" + std::to_string(i) + "_" + "Mask"] = mask;
00184     }
00185     mAsic["PwrOnPA"] = static_cast<int>(bs[8 * 72]);
00186     mAsic["Cmdb3SS"] = static_cast<int>(bs[8 * 72 + 1]);
00187     mAsic["Cmdb2SS"] = static_cast<int>(bs[8 * 72 + 2]);
00188     mAsic["Cmdb1SS"] = static_cast<int>(bs[8 * 72 + 3]);
00189     mAsic["Cmdb0SS"] = static_cast<int>(bs[8 * 72 + 4]);
00190     mAsic["SwSsc0"] = static_cast<int>(bs[8 * 72 + 5]);
00191     mAsic["SwSsc1"] = static_cast<int>(bs[8 * 72 + 6]);
00192     mAsic["SwSsc2"] = static_cast<int>(bs[8 * 72 + 7]);
00193
00194     mAsic["PwrOnBuff"] = static_cast<int>(bs[8 * 73]);
00195     mAsic["PwrOnSS"] = static_cast<int>(bs[8 * 73 + 1]);
00196     mAsic["PwrOnW"] = static_cast<int>(bs[8 * 73 + 2]);
00197     mAsic["Cmdb3Fsb2"] = static_cast<int>(bs[8 * 73 + 3]);
00198     mAsic["Cmdb2Fsb2"] = static_cast<int>(bs[8 * 73 + 4]);
00199     mAsic["Cmdb1Fsb2"] = static_cast<int>(bs[8 * 73 + 5]);
00200     mAsic["Cmdb0Fsb2"] = static_cast<int>(bs[8 * 73 + 6]);
00201     mAsic["Sw50k2"] = static_cast<int>(bs[8 * 73 + 7]);
00202
00203     mAsic["Sw100k2"] = static_cast<int>(bs[8 * 74]);
00204     mAsic["Sw100f2"] = static_cast<int>(bs[8 * 74 + 1]);
00205     mAsic["Sw50f2"] = static_cast<int>(bs[8 * 74 + 2]);
00206     mAsic["Cmdb3Fsb1"] = static_cast<int>(bs[8 * 74 + 3]);
00207     mAsic["Cmdb2Fsb1"] = static_cast<int>(bs[8 * 74 + 4]);
00208     mAsic["Cmdb1Fsb1"] = static_cast<int>(bs[8 * 74 + 5]);
00209     mAsic["Cmdb0Fsb1"] = static_cast<int>(bs[8 * 74 + 6]);
00210     mAsic["Sw50k1"] = static_cast<int>(bs[8 * 74 + 7]);
00211
00212     mAsic["Sw100k1"] = static_cast<int>(bs[8 * 75]);
00213     mAsic["Sw100f1"] = static_cast<int>(bs[8 * 75 + 1]);
00214     mAsic["Sw50f1"] = static_cast<int>(bs[8 * 75 + 2]);
00215     mAsic["Sel0"] = static_cast<int>(bs[8 * 75 + 3]);
00216     mAsic["Sel1"] = static_cast<int>(bs[8 * 75 + 4]);
00217     mAsic["PwrOnFsb"] = static_cast<int>(bs[8 * 75 + 5]);
00218     mAsic["PwrOnFsb1"] = static_cast<int>(bs[8 * 75 + 6]);
00219     mAsic["PwrOnFsb2"] = static_cast<int>(bs[8 * 75 + 7]);
00220
00221     mAsic["Sw50k0"] = static_cast<int>(bs[8 * 76]);
00222     mAsic["Sw100k0"] = static_cast<int>(bs[8 * 76 + 1]);
00223     mAsic["Sw100f0"] = static_cast<int>(bs[8 * 76 + 2]);
00224     mAsic["Sw50f0"] = static_cast<int>(bs[8 * 76 + 3]);

```

```

00225     mAsic["EnOtaQ"]           = static_cast<int>(bs[8 * 76 + 4]);
00226     mAsic["OtaQ_PwrADC"]      = static_cast<int>(bs[8 * 76 + 5]);
00227     mAsic["Discr1_PwrA"]     = static_cast<int>(bs[8 * 76 + 6]);
00228     mAsic["Discr12"]         = static_cast<int>(bs[8 * 76 + 7]);
00229
00230     mAsic["Discr11"]          = static_cast<int>(bs[8 * 77]);
00231     mAsic["RS_or_Discr1"]     = static_cast<int>(bs[8 * 77 + 1]);
00232
00233     mAsic["Header"] = asicid;
00234     for(int i = 0; i < 3; i++)
00235     {
00236         int B = 0;
00237         for(int j = 0; j < 10; j++)
00238             if(bs[8 * 102 + 2 + i * 10 + j] != 0) B += (1 << j);
00239         mAsic["B" + std::to_string(i)] = B;
00240     }
00241
00242     mAsic["Smallldac"]        = static_cast<int>(bs[8 * 106]);
00243     mAsic["DacSw"]            = static_cast<int>(bs[8 * 106 + 1]);
00244     mAsic["OtagBgSw"]         = static_cast<int>(bs[8 * 106 + 2]);
00245     mAsic["Trig2b"]           = static_cast<int>(bs[8 * 106 + 3]);
00246     mAsic["Trigl1b"]          = static_cast<int>(bs[8 * 106 + 4]);
00247     mAsic["Trig0b"]           = static_cast<int>(bs[8 * 106 + 5]);
00248     mAsic["EnTrigOut"]        = static_cast<int>(bs[8 * 106 + 6]);
00249     mAsic["DiscrOrOr"]        = static_cast<int>(bs[8 * 106 + 7]);
00250
00251     mAsic["TrigExtVal"]        = static_cast<int>(bs[8 * 107]);
00252     mAsic["RazChnIntVal"]      = static_cast<int>(bs[8 * 107 + 1]);
00253     mAsic["RazChnExtVal"]      = static_cast<int>(bs[8 * 107 + 2]);
00254     mAsic["ScOn"]              = static_cast<int>(bs[8 * 107 + 3]);
00255     mAsic["CLKMux"]            = static_cast<int>(bs[8 * 107 + 4]);
00256
00257     // EnOCDout1b   EnOCDout2b   EnOCTransmitOn1b   EnOCTransmitOn2b   EnOCChipsatb   SelStartReadout
SelEndReadout
00258     mAsic["SelEndReadout"]      = static_cast<int>(bs[8 * 108 + 1]);
00259     mAsic["SelStartReadout"]    = static_cast<int>(bs[8 * 108 + 2]);
00260     mAsic["EnOCChipsatb"]      = static_cast<int>(bs[8 * 108 + 3]);
00261     mAsic["EnOCTransmitOn2b"]  = static_cast<int>(bs[8 * 108 + 4]);
00262     mAsic["EnOCTransmitOn1b"]  = static_cast<int>(bs[8 * 108 + 5]);
00263     mAsic["EnOCDout2b"]        = static_cast<int>(bs[8 * 108 + 6]);
00264     mAsic["EnOCDout1b"]        = static_cast<int>(bs[8 * 108 + 7]);
00265     m_MapSC[asicid]            = mAsic;
00266 }

```

## 5.35 libs/core/src/DIFUnpacker.cc File Reference

```

#include "DIFUnpacker.h"
#include "Formatters.h"
#include "Words.h"
#include <bitset>
#include <cstdint>
#include <iostream>
#include <spdlog/spdlog.h>

```

### 5.35.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

Definition in file [DIFUnpacker.cc](#).

## 5.36 DIFUnpacker.cc

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

```

00001
00005 #include "DIFUnpacker.h"
00006
00007 #include "Formatters.h"
00008 #include "Words.h"
00009
00010 #include <bitset>
00011 #include <cstdint>
00012 #include <iostream>
00013 #include <spdlog/spdlog.h>
00014
00015 std::uint64_t DIFUnpacker::GrayToBin(const std::uint64_t& n)
00016 {
00017     std::uint64_t ish{1};
00018     std::uint64_t anss{n};
00019     std::uint64_t idiv{0};
00020     std::uint64_t ishmax{sizeof(std::uint64_t) * 8};
00021     while(true)
00022     {
00023         idiv = anss >> ish;
00024         anss ^= idiv;
00025         if(idiv <= 1 || ish == ishmax) return anss;
00026         ish <= 1;
00027     }
00028 }
00029
00030 std::uint32_t DIFUnpacker::getStartOfDIF(const unsigned char* cbuf, const std::uint32_t& size_buf,
const std::uint32_t& start)
00031 {
00032     std::uint32_t id0{0};
00033     for(std::uint32_t i = start; i < size_buf; i++)
00034     {
00035         if(cbuf[i] != DU::START_OF_DIF && cbuf[i] != DU::START_OF_DIF_TEMP) continue;
00036         else
00037         {
00038             id0 = i;
00039             break;
00040         }
00041         // if (cbuf[id0+DU::ID_SHIFT]>0xFF) continue;
00042     }
00043     // std::cout << "***** " << id0 << std::endl;
00044     return id0;
00045 }
00046
00047 std::uint32_t DIFUnpacker::getID(const unsigned char* cb, const std::uint32_t& idx) { return cb[idx +
DU::ID_SHIFT]; }
00048
00049 std::uint32_t DIFUnpacker::getDTC(const unsigned char* cb, const std::uint32_t& idx) { return (cb[idx
+ DU::DTC_SHIFT] << 24) + (cb[idx + DU::DTC_SHIFT + 1] << 16) + (cb[idx + DU::DTC_SHIFT + 2] << 8) +
cb[idx + DU::DTC_SHIFT + 3]; }
00050
00051 std::uint32_t DIFUnpacker::getGTC(const unsigned char* cb, const std::uint32_t& idx) { return (cb[idx
+ DU::GTC_SHIFT] << 24) + (cb[idx + DU::GTC_SHIFT + 1] << 16) + (cb[idx + DU::GTC_SHIFT + 2] << 8) +
cb[idx + DU::GTC_SHIFT + 3]; }
00052
00053 std::uint64_t DIFUnpacker::getAbsoluteBCID(const unsigned char* cb, const std::uint32_t& idx)
00054 {
00055     std::uint64_t Shift{16777216ULL}; // to shift the value from the 24 first bits
00056     std::uint64_t pos{idx + DU::ABCID_SHIFT};
00057     std::uint64_t LBC = ((cb[pos] << 16) | (cb[pos + 1] << 8) | (cb[pos + 2])) * Shift + ((cb[pos + 3] <
16) | (cb[pos + 4] << 8) | (cb[pos + 5]));
00058     return LBC;
00059 }
00060
00061 std::uint32_t DIFUnpacker::getBCID(const unsigned char* cb, const std::uint32_t& idx) { return (cb[idx
+ DU::BCID_SHIFT] << 16) + (cb[idx + DU::BCID_SHIFT + 1] << 8) + cb[idx + DU::BCID_SHIFT + 2]; }
00062 std::uint32_t DIFUnpacker::getLines(const unsigned char* cb, const std::uint32_t& idx) { return
(cb[idx + DU::LINES_SHIFT] >> 4) & 0x5; }
00063
00064 bool DIFUnpacker::hasLine(const std::uint32_t& line, const unsigned char* cb, const std::uint32_t&
idx) { return ((cb[idx + DU::LINES_SHIFT] >> line) & 0x1); }
00065
00066 std::uint32_t DIFUnpacker::getTASU1(const unsigned char* cb, const std::uint32_t& idx) { return
(cb[idx + DU::TASU1_SHIFT] << 24) + (cb[idx + DU::TASU1_SHIFT + 1] << 16) + (cb[idx + DU::TASU1_SHIFT +
2] << 8) + cb[idx + DU::TASU1_SHIFT + 3]; }
00067
00068 std::uint32_t DIFUnpacker::getTASU2(const unsigned char* cb, const std::uint32_t& idx) { return
(cb[idx + DU::TASU2_SHIFT] << 24) + (cb[idx + DU::TASU2_SHIFT + 1] << 16) + (cb[idx + DU::TASU2_SHIFT +
2] << 8) + cb[idx + DU::TASU2_SHIFT + 3]; }
00069
00070 std::uint32_t DIFUnpacker::getTDIF(const unsigned char* cb, const std::uint32_t& idx) { return (cb[idx
+ DU::TDIF_SHIFT]); }

```

```

00071
00072 bool DIFUnpacker::hasTemperature(const unsigned char* cb, const std::uint32_t& idx) { return (cb[idx]
== DU::START_OF_DIF_TEMP); }
00073
00074 bool DIFUnpacker::hasAnalogReadout(const unsigned char* cb, const std::uint32_t& idx) { return
(DIFUnpacker::getLines(cb, idx) != 0); }
00075
00076 std::uint32_t DIFUnpacker::getFrameAsicHeader(const unsigned char* framePtr) { return
(framePtr[DU::FRAME_ASIC_HEADER_SHIFT]); }
00077
00078 std::uint32_t DIFUnpacker::getFrameBCID(const unsigned char* framePtr)
00079 {
00080     std::uint32_t igray = (framePtr[DU::FRAME_BCID_SHIFT] << 16) + (framePtr[DU::FRAME_BCID_SHIFT + 1] <<
8) + framePtr[DU::FRAME_BCID_SHIFT + 2];
00081     return DIFUnpacker::GrayToBin(igray);
00082 }
00083
00084 bool DIFUnpacker::getFramePAD(const unsigned char* framePtr, const std::uint32_t& ip)
00085 {
00086     std::uint32_t* iframe{(std::uint32_t*)&framePtr[DU::FRAME_DATA_SHIFT]};
00087     return ((iframe[3 - ip / 32] >> (ip % 32)) & 0x1);
00088 }
00089
00090 bool DIFUnpacker::getFrameLevel(const unsigned char* framePtr, const std::uint32_t& ip, const
std::uint32_t& level) { return ((framePtr[DU::FRAME_DATA_SHIFT + ((3 - ip / 16) * 4 + (ip % 16) / 4)]
>> (7 - (((ip % 16) % 4) * 2 + level))) & 0x1); }
00091
00092 std::uint32_t DIFUnpacker::getAnalogPtr(std::vector<unsigned char*>& vLines, unsigned char* cb, const
std::uint32_t& idx)
00093 {
00094     std::uint32_t fshift{idx};
00095     if(cb[fshift] != DU::START_OF_LINES) return fshift;
00096     fshift++;
00097     while(cb[fshift] != DU::END_OF_LINES)
00098     {
00099         vLines.push_back(&cb[fshift]);
00100         std::uint32_t nchip{cb[fshift]};
00101         fshift += 1 + nchip * 64 * 2;
00102     }
00103     return fshift++;
00104 }
00105
00106 std::uint32_t DIFUnpacker::getFramePtr(std::vector<unsigned char*>& vFrame, std::vector<unsigned
char*>& vLines, const std::uint32_t& max_size, unsigned char* cb, const std::uint32_t& idx)
00107 {
00108     std::uint32_t fshift{0};
00109     if(DATA_FORMAT_VERSION >= 13)
00110     {
00111         fshift = idx + DU::LINES_SHIFT + 1;
00112         if(DIFUnpacker::hasTemperature(cb, idx)) fshift = idx + DU::TDIF_SHIFT + 1;
00113         // jenlev 1
00114         if(DIFUnpacker::hasAnalogReadout(cb, idx)) fshift = DIFUnpacker::getAnalogPtr(vLines, cb, fshift);
00115         // to be implemented
00116     }
00117     else
00118     {
00119         fshift = idx + DU::BCID_SHIFT + 3;
00120         if(cb[fshift] != DU::START_OF_FRAME)
00121         {
00122             std::cout << "This is not a start of frame " << to_hex(cb[fshift]) << " \n";
00123             return fshift;
00124         }
00125         do {
00126             // printf("fshift %d and %d \n", fshift, max_size);
00127             if(cb[fshift] == DU::END_OF_DIF) return fshift;
00128             if(cb[fshift] == DU::START_OF_FRAME) fshift++;
00129             if(cb[fshift] == DU::END_OF_FRAME)
00130             {
00131                 fshift++;
00132                 continue;
00133             }
00134             std::uint32_t header = DIFUnpacker::getFrameAsicHeader(&cb[fshift]);
00135             if(header == DU::END_OF_FRAME) return (fshift + 2);
00136             // std::cout<<header<<" "<<fshift<<std::endl;
00137             if(header < 1 || header > 48) { throw header + " Header problem " + fshift; }
00138             vFrame.push_back(&cb[fshift]);
00139             fshift += DU::FRAME_SIZE;
00140             if(fshift > max_size)
00141             {
00142                 std::cout << "fshift " << fshift << " exceed " << max_size << " \n";
00143                 return fshift;
00144             }
00145             if(cb[fshift] == DU::END_OF_FRAME) fshift++;
00146         } while(true);
00147     }
00148 }

```



## 5.37 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)

### 5.37.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

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

### 5.37.2 Function Documentation

#### 5.37.2.1 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); }
```

### 5.37.2.2 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); }
```

### 5.37.2.3 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); }
```

### 5.37.2.4 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); }
```

### 5.37.2.5 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 }
```

**5.37.2.6 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); }
```

**5.37.2.7 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); }
```

**5.37.2.8 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); }
```

**5.37.2.9 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); }
```

**5.37.2.10 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 }
```

**5.37.2.11 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); }
```

**5.37.2.12 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); }
```

**5.37.2.13 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); }
```

**5.37.2.14 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); }
```

**5.37.2.15 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 }
```

### 5.37.2.16 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); }
```

### 5.37.2.17 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); }
```

### 5.37.2.18 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); }
```

### 5.37.2.19 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); }
```

### 5.37.2.20 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 }
```

## 5.38 Formatters.cc

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

```

00001
00006 #include "Formatters.h"
00007
00008 #include "Bits.h"
00009 #include "Buffer.h"
00010 #include "Words.h"
00011
00012 #include <fmt/format.h>
00013
00014 std::string to_dec(const Buffer& b, const std::size_t& begin, const std::size_t& end)
00015 {
00016     std::size_t iend = end;
00017     if(iend == -1) iend = b.size();
00018     std::string ret;
00019     for(std::size_t k = begin; k < iend; k++)
00020     {
00021         ret += to_dec(b[k]);
00022         ret += " - ";
00023     }
00024     return ret;
00025 }
00026
00027 std::string to_dec(const bit8_t& b) { return fmt::format("{:d}", b); }
00028
00029 std::string to_dec(const bit16_t& b) { return fmt::format("{:d}", b); }
00030
00031 std::string to_dec(const bit32_t& b) { return fmt::format("{:d}", b); }
00032
00033 std::string to_dec(const bit64_t& b) { return fmt::format("{:d}", b); }
00034
00035 std::string to_hex(const Buffer& b, const std::size_t& begin, const std::size_t& end)
00036 {
00037     std::size_t iend = end;
00038     if(iend == -1) iend = b.size();
00039     std::string ret;
00040     for(std::size_t k = begin; k < iend; k++)
00041     {
00042         ret += to_hex(b[k]);
00043         ret += " - ";
00044     }
00045     return ret;
00046 }
00047
00048 std::string to_hex(const bit8_t& b) { return fmt::format("{:02x}", b); }
00049
00050 std::string to_hex(const bit16_t& b) { return fmt::format("{:04x}", b); }
00051
00052 std::string to_hex(const bit32_t& b) { return fmt::format("{:08x}", b); }
00053
00054 std::string to_hex(const bit64_t& b) { return fmt::format("{:016x}", b); }
00055
00056 std::string to_bin(const Buffer& b, const std::size_t& begin, const std::size_t& end)
00057 {
00058     std::size_t iend = end;
00059     if(iend == -1) iend = b.size();
00060     std::string ret;
00061     for(std::size_t k = begin; k < iend; k++)
00062     {
00063         ret += to_bin(b[k]);
00064         ret += " - ";
00065     }
00066     return ret;
00067 }
00068
00069 std::string to_bin(const bit8_t& b) { return fmt::format("{:08b}", b); }
00070
00071 std::string to_bin(const bit16_t& b) { return fmt::format("{:016b}", b); }
00072
00073 std::string to_bin(const bit32_t& b) { return fmt::format("{:032b}", b); }
00074
00075 std::string to_bin(const bit64_t& b) { return fmt::format("{:064b}", b); }
00076
00077 std::string to_oct(const Buffer& b, const std::size_t& begin, const std::size_t& end)
00078 {
00079     std::size_t iend = end;
00080     if(iend == -1) iend = b.size();
00081     std::string ret;
00082     for(std::size_t k = begin; k < iend; k++)
00083     {
00084         ret += to_oct(b[k]);
00085         ret += " - ";
00086     }

```

```

00087     return ret;
00088 }
00089
00090 std::string to_oct(const bit8_t& b) { return fmt::format("{:#04o}", b); }
00091
00092 std::string to_oct(const bit16_t& b) { return fmt::format("{:#08o}", b); }
00093
00094 std::string to_oct(const bit32_t& b) { return fmt::format("{:#016o}", b); }
00095
00096 std::string to_oct(const bit64_t& b) { return fmt::format("{:#032o}", b); }

```

## 5.39 libs/core/src/RawBufferNavigator.cc File Reference

```

#include "RawBufferNavigator.h"
#include <iostream>

```

### 5.39.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

Definition in file [RawBufferNavigator.cc](#).

## 5.40 RawBufferNavigator.cc

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

```

00001
00005 #include "RawBufferNavigator.h"
00006
00007 #include <iostream>
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(const Buffer& b, const int& start) : m_Buffer(b) {
00017     setBuffer(b, start); }
00018
00018 std::uint8_t RawBufferNavigator::getDetectorID() { return m_Buffer[0]; }
00019
00020 bool RawBufferNavigator::validBuffer() { return m_DIFstartIndex != 0; }
00021
00022 std::uint32_t RawBufferNavigator::getStartOfDIF() { return m_DIFstartIndex; }
00023
00024 unsigned char* RawBufferNavigator::getDIFBufferStart() { return &(m_Buffer.begin())[m_DIFstartIndex]; }
00025
00026 std::uint32_t RawBufferNavigator::getDIFBufferSize() { return m_Buffer.size() - m_DIFstartIndex; }
00027
00028 Buffer RawBufferNavigator::getDIFBuffer() { return Buffer(getDIFBufferStart(), getDIFBufferSize()); }
00029
00030 DIFPtr& RawBufferNavigator::getDIFPtr()
00031 {
00032     m_TheDIFPtr.setBuffer(getDIFBufferStart(), getDIFBufferSize());
00033     return m_TheDIFPtr;
00034 }
00035
00036 std::uint32_t RawBufferNavigator::getEndOfDIFData() { return getDIFPtr().getGetFramePtrReturn() + 3; }
00037
00038 std::uint32_t RawBufferNavigator::getSizeAfterDIFPtr() { return getDIFBufferSize() -
00039     getDIFPtr().getGetFramePtrReturn(); }

```

```

00040 std::uint32_t RawBufferNavigator::getDIF_CRC()
00041 {
00042     uint32_t i{getEndOfDIFData()};
00043     uint32_t ret{0};
00044     ret |= ((m_Buffer.begin()[i - 2]) << 8);
00045     ret |= m_Buffer.begin()[i - 1];
00046     return ret;
00047 }
00048
00049 bool RawBufferNavigator::hasSlowControlData() { return getDIFBufferStart()[getEndOfDIFData()] == 0xb1;
00050 }
00051 Buffer RawBufferNavigator::getSCBuffer()
00052 {
00053     setSCBuffer();
00054     return m_SCbuffer;
00055 }
00056
00057 bool RawBufferNavigator::badSCData()
00058 {
00059     setSCBuffer();
00060     return m_BadSCdata;
00061 }
00062
00063 void RawBufferNavigator::setSCBuffer()
00064 {
00065     if(!hasSlowControlData()) return;
00066     if(m_SCbuffer.size() != 0) return; // deja fait
00067     if(m_BadSCdata) return;
00068     m_SCbuffer.set(&(getDIFBufferStart()[getEndOfDIFData()]));
00069     // compute Slow Control size
00070     std::size_t maxsize{m_Buffer.size() - m_DIFstartIndex - getEndOfDIFData() + 1}; // should I +1 here
00071     ?
00072     uint32_t k{1}; // SC Header
00073     uint32_t dif_ID{m_SCbuffer[1]};
00074     uint32_t chipSize{m_SCbuffer[3]};
00075     while((dif_ID != 0x01 && m_SCbuffer[k] != 0x01 && k < maxsize) || (dif_ID == 0x01 && m_SCbuffer[k +
00076 2] == chipSize && k < maxsize))
00077     {
00078         k += 2; // DIF ID + ASIC Header
00079         uint32_t scsize = m_SCbuffer[k];
00080         if(scsize != 74 && scsize != 109)
00081         {
00082             std::cout << "PROBLEM WITH SC SIZE " << scsize << std::endl;
00083             k = 0;
00084             m_BadSCdata = true;
00085             break;
00086         }
00087         k++; // skip size bit
00088         k += scsize; // skip the data
00089     }
00090     if(m_SCbuffer[k] == 0x01 && !m_BadSCdata) m_SCbuffer.setSize(k + 1); // add the trailer
00091     else
00092     {
00093         m_BadSCdata = true;
00094         std::cout << "PROBLEM SC TRAILER NOT FOUND " << std::endl;
00095     }
00096 }
00097
00098 Buffer RawBufferNavigator::getEndOfAllData()
00099 {
00100     setSCBuffer();
00101     if(hasSlowControlData() && !m_BadSCdata) { return Buffer(&(m_SCbuffer.begin()[m_SCbuffer.size()]),
00102         getSizeAfterDIFPtr() - 3 - m_SCbuffer.size()); }
00103     else
00104     {
00105         return Buffer(&(getDIFBufferStart()[getEndOfDIFData()]), getSizeAfterDIFPtr() - 3); // remove the
00106         2 bytes for CRC and the DIF trailer
00107     }
00108 }

```

## 5.41 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](#)

### 5.41.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

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

## 5.42 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 Interface
00015 {
00016 public:
00017     textDump()
00018     {
00019         m_InternalLogger = std::make_shared<spdlog::logger>("textDump",
00020             std::make_shared<spdlog::sinks::stdout_color_sink_mt>());
00021         m_InternalLogger->set_level(spdlog::level::trace);
00022     }
00023     void start();
00024     void processDIF(const DIFPtr&);
00025     void processFrame(const DIFPtr&, uint32_t frameIndex);
00026     void processPadInFrame(const DIFPtr&, uint32_t frameIndex, uint32_t
00027         channelIndex);
00028     void processSlowControl(Buffer);
00029     void end();
00030     std::shared_ptr<spdlog::logger>& print() { return m_InternalLogger; }
00031     void setLevel(const spdlog::level::level_enum& level) {
00032         m_InternalLogger->set_level(level); }
00033 private:
00034     // This class is a dumb class to print on terminal so we need the logger + the standard one given by
00035     the interface.
00036     std::shared_ptr<spdlog::logger> m_InternalLogger{nullptr};
00037 };
```

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

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

### 5.43.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

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

## 5.44 textDump.cc

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

```
00001
00005 #include "textDump.h"
00006
00007 #include "DIFPtr.h"
00008
00009 void textDump::start() { print()->info("Will dump bunch of DIF data"); }
00010
00011 void textDump::processDIF(const DIFPtr& d) { print()->info("DIF_ID : {}, DTC : {}, GTC : {}, DIF BCID
    {}, Absolute BCID : {}, Nbr frames {}", d.getDIFid(), d.getDTC(), d.getGTC(), d.getBCID(),
    d.getAbsoluteBCID(), d.getNumberOfFrames()); }
00012
00013 void textDump::processFrame(const DIFPtr& d, uint32_t frameIndex)
00014 {
00015     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));
00016 }
00017
00018 void textDump::processPadInFrame(const DIFPtr& d, uint32_t frameIndex, uint32_t channelIndex)
00019 {
00020     if(d.getThresholdStatus(frameIndex, channelIndex) > 0) { print()->info("\t\tChannel {}, Threshold
    {}", channelIndex, d.getThresholdStatus(frameIndex, channelIndex)); }
00021 }
00022
00023 void textDump::processSlowControl(Buffer) { print()->error("textDump::processSlowControl not
    implemented yet."); }
00024
00025 void textDump::end() { print()->info("textDump end of report"); }
```

## 5.45 libs/interface/LCIO/include/LCIOWriter.h File Reference

### 5.45.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

## 5.46 LCIOWriter.h

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

```
00001
00005 #pragma once
```

## 5.47 libs/interface/LCIO/src/LCIOWriter.cc File Reference

### 5.47.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

Definition in file [LCIOWriter.cc](#).

## 5.48 LCIOWriter.cc

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

00001

## 5.49 libs/interface/RawDataReader/include/RawdataReader.h File Reference

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

### Classes

- class [RawdataReader](#)

### 5.49.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

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

## 5.50 RawdataReader.h

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

```
00001
00005 #pragma once
00006
00007 #include "Interface.h"
00008
00009 #include <array>
00010 #include <cstdint>
00011 #include <fstream>
00012 #include <string>
00013 #include <vector>
00014
00015 class Buffer;
00016
00017 class RawdataReader : public Interface
00018 {
00019 public:
00020     explicit RawdataReader(const char* fileName);
00021     void start();
00022     void end();
00023     float getFileSize();
00024     void openFile(const std::string& fileName);
00025     void closeFile();
00026     bool nextEvent();
00027     bool nextDIFbuffer();
00028     const Buffer& getSDHCALBuffer();
00029     virtual ~RawdataReader() { closeFile(); }
```

```

00030     static void setDefaultBufferSize(const std::size_t& size);
00031
00032 private:
00033     void                uncompress();
00034     std::ifstream       m_FileStream;
00035     void                setFileSize(const std::size_t& size);
00036     static std::size_t  m_BufferSize;
00037     std::size_t         m_FileSize{0};
00038     std::uint32_t       m_NumberOfDIF{0};
00039     std::uint32_t       m_EventNumber{0};
00040     std::vector<bit8_t> m_buf;
00041     Buffer              m_Buffer;
00042     std::string         m_Filename;
00043 };

```

## 5.51 libs/interface/RawDataReader/src/RawdataReader.cc File Reference

```

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

```

### 5.51.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

Definition in file [RawdataReader.cc](#).

## 5.52 RawdataReader.cc

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

```

00001
00004 #include "RawdataReader.h"
00005
00006 #include <stdint>
00007 #include <cstring>
00008 #include <stdexcept>
00009 #include <zlib.h>
00010
00012 std::size_t RawdataReader::m_BufferSize = 0x100000;
00013
00014 void RawdataReader::setDefaultBufferSize(const std::size_t& size) { m_BufferSize = size; }
00015
00016 RawdataReader::RawdataReader(const char* fileName)
00017 {
00018     m_buf.reserve(m_BufferSize);
00019     m_Filename = fileName;
00020 }
00021
00022 void RawdataReader::start() { openFile(m_Filename); }
00023
00024 void RawdataReader::end() { closeFile(); }
00025
00026 void RawdataReader::uncompress()
00027 {
00028     static const std::size_t size_buffer{0x20000};
00029     std::size_t              shift{3 * sizeof(std::uint32_t) + sizeof(std::uint64_t)};
00030     static bit8_t            obuf[size_buffer];
00031     unsigned long            size_buffer_end{0x20000}; // NOLINT(runtime/int)
00032     std::int8_t              rc = ::uncompress(obuf, &size_buffer_end, &m_Buffer[shift], m_Buffer.size()
- shift);

```

```

00033     switch(rc)
00034     {
00035         case Z_OK: break;
00036         default: throw "decompress error"; break;
00037     }
00038     memcpy(&m_Buffer[shift], obuf, size_buffer_end);
00039     m_Buffer.setSize(size_buffer_end + shift);
00040 }
00041
00042 void RawdataReader::closeFile()
00043 {
00044     try
00045     {
00046         if(m_FileStream.is_open()) m_FileStream.close();
00047     }
00048     catch(const std::ios_base::failure& e)
00049     {
00050         log()->error("Caught an ios_base::failure in closeFile : {} {}", e.what(), e.code().value());
00051         throw;
00052     }
00053 }
00054
00055 void RawdataReader::openFile(const std::string& fileName)
00056 {
00057     try
00058     {
00059         m_FileStream.rdbuf()->pubsetbuf(0, 0);
00060         m_FileStream.exceptions(std::ifstream::failbit | std::ifstream::badbit);
00061         m_FileStream.open(fileName.c_str(), std::ios::in | std::ios::binary | std::ios::ate); // Start at
the end to directly calculate the size of the file then come back to beginning
00062         m_FileStream.rdbuf()->pubsetbuf(0, 0);
00063         if(m_FileStream.is_open())
00064         {
00065             setFileSize(m_FileStream.tellg());
00066             m_FileStream.seekg(0, std::ios::beg);
00067         }
00068     }
00069     catch(const std::ios_base::failure& e)
00070     {
00071         log()->error("Caught an ios_base::failure in openFile : {}", e.what());
00072         throw;
00073     }
00074 }
00075
00076 bool RawdataReader::nextEvent()
00077 {
00078     try
00079     {
00080         m_FileStream.read(reinterpret_cast<char*>(&m_EventNumber), sizeof(std::uint32_t));
00081         m_FileStream.read(reinterpret_cast<char*>(&m_NumberOfDIF), sizeof(std::uint32_t));
00082     }
00083     catch(const std::ios_base::failure& e)
00084     {
00085         log()->error("Caught an ios_base::failure in openFile : {}", e.what());
00086         return false;
00087     }
00088     return true;
00089 }
00090
00091 bool RawdataReader::nextDIFbuffer()
00092 {
00093     try
00094     {
00095         static int DIF_processed{0};
00096         if(DIF_processed >= m_NumberOfDIF)
00097         {
00098             DIF_processed = 0;
00099             return false;
00100         }
00101         else
00102         {
00103             DIF_processed++;
00104             std::uint32_t bsize{0};
00105             m_FileStream.read(reinterpret_cast<char*>(&bsize), sizeof(std::uint32_t));
00106             m_FileStream.read(reinterpret_cast<char*>(&m_buf[0]), bsize);
00107             m_Buffer = Buffer(m_buf);
00108         }
00109     }
00110     catch(const std::ios_base::failure& e)
00111     {
00112         log()->error("Caught an ios_base::failure in openFile : {}", e.what());
00113         return false;
00114     }
00115     return true;
00116 }
00117
00118 const Buffer& RawdataReader::getSDHCALBuffer()

```

```

00119 {
00120     uncompress();
00121     return m_Buffer;
00122 }
00123
00124 void RawdataReader::setFileSize(const std::size_t& size) { m_FileSize = size; }
00125
00126 float RawdataReader::getFileSize() { return m_FileSize; }

```

## 5.53 libs/interface/ROOT/include/DIF.h File Reference

```

#include "Hit.h"
#include <TObject.h>
#include <stdint>
#include <vector>

```

### Classes

- class [DIF](#)

### 5.53.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

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

## 5.54 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 <stdint>
00011 #include <vector>
00012
00013 class DIF : public TObject
00014 {
00015 public:
00016     void addHit(const Hit&);
00017     void setID(const std::uint8_t&);
00018     std::uint8_t getID() const;
00019     void setDTC(const std::uint32_t&);
00020     std::uint32_t getDTC() const;
00021     void setGTC(const std::uint32_t&);
00022     std::uint32_t getGTC() const;
00023     void setDIFBCID(const std::uint32_t&);
00024     std::uint32_t getDIFBCID() const;
00025     void setAbsoluteBCID(const std::uint64_t&);
00026     std::uint64_t getAbsoluteBCID() const;
00027
00028 private:
00029     std::uint8_t m_ID{0};
00030     std::uint32_t m_DTC{0};
00031     std::uint32_t m_GTC{0};
00032     std::uint32_t m_DIFBCID{0};
00033     std::uint64_t m_AbsoluteBCID{0};
00034     std::vector<Hit> m_Hits;
00035     ClassDef(DIF, 1);
00036 };

```

## 5.55 libs/interface/ROOT/include/DIFLinkDef.h File Reference

```
#include <vector>
```

### 5.55.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

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

## 5.56 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
```

## 5.57 libs/interface/ROOT/include/Event.h File Reference

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

### Classes

- class [Event](#)

### 5.57.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

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

## 5.58 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 class Event : public TObject
00014 {
00015 public:
00016     void clear();
00017     void addDIF(const DIF& dif);
00018
00019 private:
00020     std::map<std::uint8_t, DIF> DIFs;
00021     ClassDef(Event, 1);
00022 };
```

## 5.59 libs/interface/ROOT/include/EventLinkDef.h File Reference

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

### 5.59.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

## 5.60 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
```

## 5.61 libs/interface/ROOT/include/Hit.h File Reference

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



## Classes

- class [Hit](#)

### 5.61.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

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

## 5.62 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          setDIF(const std::uint8_t&);
00014     void          setASIC(const std::uint8_t&);
00015     void          setChannel(const std::uint8_t&);
00016     void          setThreshold(const std::uint8_t&);
00017     void          setDTC(const std::uint32_t&);
00018     void          setGTC(const std::uint32_t&);
00019     void          setDIFBCID(const std::uint32_t&);
00020     void          setFrameBCID(const std::uint32_t&);
00021     void          setTimestamp(const std::uint32_t&);
00022     void          setAbsoluteBCID(const std::uint64_t&);
00023     std::uint8_t  getDIFid();
00024     std::uint8_t  getASICid();
00025     std::uint8_t  getChannelId();
00026     std::uint8_t  getThreshold();
00027     std::uint32_t  getDTC();
00028     std::uint32_t  getGTC();
00029     std::uint32_t  getDIFBCID();
00030     std::uint32_t  setFrameBCID();
00031     std::uint32_t  getTimestamp();
00032     std::uint64_t  getAbsoluteBCID();
00033
00034 private:
00035     std::uint8_t  m_DIF{0};
00036     std::uint8_t  m_ASIC{0};
00037     std::uint8_t  m_Channel{0};
00038     std::uint8_t  m_Threshold{0};
00039     std::uint32_t  m_DTC{0};
00040     std::uint32_t  m_GTC{0};
00041     std::uint32_t  m_DIFBCID{0};
00042     std::uint32_t  m_FrameBCID{0};
00043     std::uint32_t  m_Timestamp{0};
00044     std::uint64_t  m_AbsoluteBCID{0};
00045     ClassDef(Hit, 1);
00046 };
```

## 5.63 libs/interface/ROOT/include/HitLinkDef.h File Reference

### 5.63.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

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

## 5.64 HitLinkDef.h

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

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

## 5.65 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](#)

## 5.66 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 Interface
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 };

```

## 5.67 libs/interface/ROOT/src/DIF.cc File Reference

```

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

```

### 5.67.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

Definition in file [DIF.cc](#).

## 5.68 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; }

```

## 5.69 libs/interface/ROOT/src/Event.cc File Reference

```

#include "Event.h"

```

### 5.69.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

Definition in file [Event.cc](#).

## 5.70 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; }
```

## 5.71 libs/interface/ROOT/src/Hit.cc File Reference

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

### 5.71.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

Definition in file [Hit.cc](#).

## 5.72 Hit.cc

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

```
00001
00006 #include "Hit.h"
00007
00008 #include <cstdint>
00009
00010 void Hit::setDIF(const std::uint8_t& dif) { m_DIF = dif; }
00011
00012 void Hit::setASIC(const std::uint8_t& asic) { m_ASIC = asic; }
00013
00014 void Hit::setChannel(const std::uint8_t& channel) { m_Channel = channel; }
00015
00016 void Hit::setThreshold(const std::uint8_t& threshold) { m_Threshold = threshold; }
00017
00018 void Hit::setDTC(const std::uint32_t& dtc) { m_DTC = dtc; }
00019
00020 void Hit::setGTC(const std::uint32_t& gtc) { m_GTC = gtc; }
00021
00022 void Hit::setDIFBCID(const std::uint32_t& difbcid) { m_DIFBCID = difbcid; }
00023
00024 void Hit::setFrameBCID(const std::uint32_t& framebcid) { m_FrameBCID = framebcid; }
00025
00026 void Hit::setTimestamp(const std::uint32_t& timestamp) { m_Timestamp = timestamp; }
```

```

00027
00028 void Hit::setAbsoluteBCID(const std::uint64_t& absolutebcid) { m_AbsoluteBCID = absolutebcid; }
00029
00030 std::uint8_t Hit::getDIFid() { return m_DIF; }
00031
00032 std::uint8_t Hit::getASICid() { return m_ASIC; }
00033
00034 std::uint8_t Hit::getChannelId() { return m_Channel; }
00035
00036 std::uint8_t Hit::getThreshold() { return m_Threshold; }
00037
00038 std::uint32_t Hit::getDTC() { return m_DTC; }
00039
00040 std::uint32_t Hit::getGTC() { return m_GTC; }
00041
00042 std::uint32_t Hit::getDIFBCID() { return m_DIFBCID; }
00043
00044 std::uint32_t Hit::getFrameBCID() { return m_FrameBCID; }
00045
00046 std::uint32_t Hit::getTimestamp() { return m_Timestamp; }
00047
00048 std::uint64_t Hit::getAbsoluteBCID() { return m_AbsoluteBCID; }

```

## 5.73 libs/interface/ROOT/src/ROOTWriter.cc File Reference

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

### 5.73.1 Detailed Description

#### Copyright

2022 G.Grenier F.Lagarde

Definition in file [ROOTWriter.cc](#).

## 5.74 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() {}
00011
00012 void ROOTWriter::start()
00013 {
00014     m_File = TFile::Open(m_Filename.c_str(), "RECREATE", m_Filename.c_str(),
00015         ROOT::CompressionSettings(ROOT::kLZMA, 9));
00016     m_Tree = new TTree("RawData", "Raw SDHCAL data tree");
00017     m_Tree->Branch("Events", &m_Event, 10, 0);
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(static_cast<std::uint8_t>(channelIndex));
00054     m_Hit->setThreshold(static_cast<std::uint8_t>(d.getThresholdStatus(frameIndex, channelIndex)));
00055 }
00056
00057 void ROOTWriter::startEvent() { m_Event = new Event(); }
00058
00059 void ROOTWriter::endEvent()
00060 {
00061     m_Tree->Fill();
00062     if(m_Event) delete m_Event;
00063 }
00064
00065 void ROOTWriter::startDIF() { m_DIF = new DIF(); }
00066
00067 void ROOTWriter::endDIF()
00068 {
00069     m_Event->addDIF(*m_DIF);
00070     delete m_DIF;
00071 }
00072
00073 void ROOTWriter::startFrame() { m_Hit = new Hit(); }
00074
00075 void ROOTWriter::endFrame()
00076 {
00077     if(m_Hit->getThreshold() != 0) { m_DIF->addHit(*m_Hit); }
00078     delete m_Hit;
00079 }
00080
00081 void ROOTWriter::startPad() {}
00082
00083 void ROOTWriter::endPad() {}

```