# streamout

Generated by Doxygen 1.9.2

**1 Hierarchical Index** 1

1.1 Class Hierarchy . . . 1

**2 Class Index** 3

2.1 Class List . . . 3

**3 File Index** 5

3.1 File List . . . 5

**4 Class Documentation** 7

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() [2/5] . . . 8

4.1.2.3 Buffer() [3/5] . . . 8

4.1.2.4 Buffer() [4/5] . . . 8

4.1.2.5 Buffer() [5/5] . . . 8

4.1.2.6 ~Buffer() . . . 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 ROOTtreeDest::DATA Struct Reference . . . 10

4.2.1 Detailed Description . . . 11

4.2.2 Member Data Documentation . . . 11

4.2.2.1 AbsoluteBCID . . . 11

4.2.2.2 ASICid . . . 11

4.2.2.3 CHANNELid . . . 11

4.2.2.4 DIF_BCID . . . 12

4.2.2.5 DIFid . . . 12

4.2.2.6 DTC . . . 12

4.2.2.7 frame_BCID . . . 12

4.2.2.8 GTC . . . 12

4.2.2.9 Thresh . . . 12

4.2.2.10 timeStamp . . . 13

4.3 DIFPtr Class Reference . . . 13

4.3.1 Detailed Description . . . 13

4.3.2 Member Function Documentation . . . 14
4.3.2.1 getAbsoluteBCID() . . . 14
4.3.2.2 getASICid() . . . 14
4.3.2.3 getBCID() . . . 14
4.3.2.4 getDIFid() . . . 14
4.3.2.5 getDTC() . . . 14
4.3.2.6 getFrameAsicHeader() . . . 15
4.3.2.7 getFrameBCID() . . . 15
4.3.2.8 getFrameLevel() . . . 15
4.3.2.9 getFramePtr() . . . 15
4.3.2.10 getFramesVector() . . . 15
4.3.2.11 getFrameTimeToTrigger() . . . 16
4.3.2.12 getGetFramePtrReturn() . . . 16
4.3.2.13 getGTC() . . . 16
4.3.2.14 getID() . . . 16
4.3.2.15 getLines() . . . 16
4.3.2.16 getLinesVector() . . . 16
4.3.2.17 getNumberOfFrames() . . . 17
4.3.2.18 getPtr() . . . 17
4.3.2.19 getTASU1() . . . 17
4.3.2.20 getTASU2() . . . 17
4.3.2.21 getTDIF() . . . 17
4.3.2.22 getTemperatureASU1() . . . 17
4.3.2.23 getTemperatureASU2() . . . 18
4.3.2.24 getTemperatureDIF() . . . 18
4.3.2.25 getThresholdStatus() . . . 18
4.3.2.26 hasAnalogReadout() . . . 18
4.3.2.27 hasLine() . . . 18
4.3.2.28 hasTemperature() . . . 19
4.3.2.29 setBuffer() . . . 19
4.4 DIFSlowControl Class Reference . . . 19
4.4.1 Detailed Description . . . 20
4.4.2 Constructor & Destructor Documentation . . . 20
4.4.2.1 DIFSlowControl() . . . 20
4.4.3 Member Function Documentation . . . 21
4.4.3.1 Dump() . . . 21
4.4.3.2 getChipSlowControl() [1/2] . . . 21
4.4.3.3 getChipSlowControl() [2/2] . . . 22
4.4.3.4 getChipsMap() . . . 22
4.4.3.5 getDIFId() . . . 22
4.5 DIFUnpacker Class Reference . . . 22
4.5.1 Detailed Description . . . 23

4.5.2 Member Function Documentation . . . 23
4.5.2.1 dumpFrameOld() . . . 23
4.5.2.2 getAbsoluteBCID() . . . 24
4.5.2.3 getAnalogPtr() . . . 24
4.5.2.4 getBCID() . . . 25
4.5.2.5 getDTC() . . . 25
4.5.2.6 getFrameAsicHeader() . . . 25
4.5.2.7 getFrameBCID() . . . 25
4.5.2.8 getFrameLevel() . . . 25
4.5.2.9 getFramePAD() . . . 26
4.5.2.10 getFramePtr() . . . 26
4.5.2.11 getGTC() . . . 27
4.5.2.12 getID() . . . 27
4.5.2.13 getLines() . . . 27
4.5.2.14 getStartOfDIF() . . . 27
4.5.2.15 getTASU1() . . . 28
4.5.2.16 getTASU2() . . . 28
4.5.2.17 getTDIF() . . . 28
4.5.2.18 GrayToBin() . . . 28
4.5.2.19 hasAnalogReadout() . . . 29
4.5.2.20 hasLine() . . . 29
4.5.2.21 hasTemperature() . . . 29
4.5.2.22 swap_bytes() . . . 29
4.6 Interface Class Reference . . . 30
4.6.1 Detailed Description . . . 30
4.6.2 Constructor & Destructor Documentation . . . 30
4.6.2.1 Interface() . . . 30
4.6.2.2 ~Interface() . . . 31
4.6.3 Member Function Documentation . . . 31
4.6.3.1 log() . . . 31
4.6.3.2 setLogger() . . . 31
4.7 RawBufferNavigator Class Reference . . . 31
4.7.1 Detailed Description . . . 32
4.7.2 Constructor & Destructor Documentation . . . 32
4.7.2.1 RawBufferNavigator() [1/2] . . . 32
4.7.2.2 ~RawBufferNavigator() . . . 32
4.7.2.3 RawBufferNavigator() [2/2] . . . 32
4.7.3 Member Function Documentation . . . 32
4.7.3.1 badSCData() . . . 32
4.7.3.2 getDIF_CRC() . . . 33
4.7.3.3 getDIFBuffer() . . . 33
4.7.3.4 getDIFBufferSize() . . . 33

4.7.3.5 getDIFBufferStart() . . . 33
4.7.3.6 getDIFPtr() . . . 33
4.7.3.7 getEndOfAllData() . . . 34
4.7.3.8 getEndOfDIFData() . . . 34
4.7.3.9 getSCBuffer() . . . 34
4.7.3.10 getSizeAfterDIFPtr() . . . 34
4.7.3.11 getStartOfDIF() . . . 34
4.7.3.12 hasSlowControlData() . . . 35
4.7.3.13 setBuffer() . . . 35
4.7.3.14 StartAt() . . . 35
4.7.3.15 validBuffer() . . . 35
4.8 RawdataReader Class Reference . . . 36
4.8.1 Detailed Description . . . 36
4.8.2 Constructor & Destructor Documentation . . . 36
4.8.2.1 RawdataReader() . . . 36
4.8.2.2 ∼RawdataReader() . . . 37
4.8.3 Member Function Documentation . . . 37
4.8.3.1 closeFile() . . . 37
4.8.3.2 end() . . . 37
4.8.3.3 getFileSize() . . . 37
4.8.3.4 getSDHCALBuffer() . . . 37
4.8.3.5 nextDIFbuffer() . . . 38
4.8.3.6 nextEvent() . . . 38
4.8.3.7 openFile() . . . 38
4.8.3.8 setDefaultBufferSize() . . . 39
4.8.3.9 start() . . . 39
4.9 ROOTtreeDest Class Reference . . . 39
4.9.1 Detailed Description . . . 40
4.9.2 Constructor & Destructor Documentation . . . 40
4.9.2.1 ROOTtreeDest() . . . 40
4.9.3 Member Function Documentation . . . 40
4.9.3.1 end() . . . 40
4.9.3.2 processDIF() . . . 40
4.9.3.3 processFrame() . . . 41
4.9.3.4 processPadInFrame() . . . 41
4.9.3.5 processSlowControl() . . . 41
4.9.3.6 start() . . . 41
4.10 SDHCAL_buffer_loop< SOURCE, DESTINATION > Class Template Reference . . . 42
4.10.1 Detailed Description . . . 42
4.10.2 Constructor & Destructor Documentation . . . 42
4.10.2.1 SDHCAL_buffer_loop() . . . 42
4.10.3 Member Function Documentation . . . 42

4.10.3.1 addSink() . . . 43
4.10.3.2 log() . . . 43
4.10.3.3 loop() . . . 43
4.10.3.4 printAllCounters() . . . 44
4.11 SDHCAL_buffer_LoopCounter Struct Reference . . . 44
4.11.1 Detailed Description . . . 45
4.11.2 Member Function Documentation . . . 45
4.11.2.1 printAllCounters() . . . 45
4.11.2.2 printCounter() . . . 45
4.11.3 Member Data Documentation . . . 45
4.11.3.1 DIFPtrValueAtReturnedPos . . . 46
4.11.3.2 DIFStarter . . . 46
4.11.3.3 hasBadSlowControl . . . 46
4.11.3.4 hasSlowControl . . . 46
4.11.3.5 NonZeroValusAtEndOfData . . . 46
4.11.3.6 SizeAfterAllData . . . 46
4.11.3.7 SizeAfterDIFPtr . . . 47
4.12 textDump Class Reference . . . 47
4.12.1 Detailed Description . . . 47
4.12.2 Constructor & Destructor Documentation . . . 47
4.12.2.1 textDump() . . . 48
4.12.3 Member Function Documentation . . . 48
4.12.3.1 end() . . . 48
4.12.3.2 print() . . . 48
4.12.3.3 processDIF() . . . 48
4.12.3.4 processFrame() . . . 49
4.12.3.5 processPadInFrame() . . . 49
4.12.3.6 processSlowControl() . . . 49
4.12.3.7 setLevel() . . . 49
4.12.3.8 start() . . . 50
4.13 Timer Class Reference . . . 50
4.13.1 Detailed Description . . . 50
4.13.2 Member Function Documentation . . . 50
4.13.2.1 getElapsedTime() . . . 50
4.13.2.2 start() . . . 50
4.13.2.3 stop() . . . 50

**5 File Documentation** **51**

5.1 /home/runner/work/streamout/streamout/libs/core/include/Bits.h File Reference . . . 51
5.1.1 Detailed Description . . . 51
5.1.2 Typedef Documentation . . . 51
5.1.2.1 bit16_t . . . 52

5.1.2.2 bit32_t . . . 52
5.1.2.3 bit64_t . . . 52
5.1.2.4 bit8_t . . . 52
5.1.3 Function Documentation . . . 52
5.1.3.1 operator<<() . . . 52
5.2 Bits.h . . . 53
5.3 /home/runner/work/streamout/streamout/libs/core/include/Buffer.h File Reference . . . 53
5.3.1 Detailed Description . . . 53
5.4 Buffer.h . . . 53
5.5 /home/runner/work/streamout/streamout/libs/core/include/DIFPtr.h File Reference . . . 54
5.5.1 Detailed Description . . . 54
5.6 DIFPtr.h . . . 54
5.7 /home/runner/work/streamout/streamout/libs/core/include/DIFSlowControl.h File Reference . . . 56
5.7.1 Detailed Description . . . 56
5.8 DIFSlowControl.h . . . 57
5.9 /home/runner/work/streamout/streamout/libs/core/include/DIFUnpacker.h File Reference . . . 57
5.9.1 Detailed Description . . . 57
5.10 DIFUnpacker.h . . . 58
5.11 /home/runner/work/streamout/streamout/libs/core/include/Formatters.h File Reference . . . 58
5.11.1 Detailed Description . . . 59
5.11.2 Function Documentation . . . 59
5.11.2.1 to_bin() [1/5] . . . 59
5.11.2.2 to_bin() [2/5] . . . 59
5.11.2.3 to_bin() [3/5] . . . 59
5.11.2.4 to_bin() [4/5] . . . 60
5.11.2.5 to_bin() [5/5] . . . 60
5.11.2.6 to_dec() [1/5] . . . 60
5.11.2.7 to_dec() [2/5] . . . 60
5.11.2.8 to_dec() [3/5] . . . 61
5.11.2.9 to_dec() [4/5] . . . 61
5.11.2.10 to_dec() [5/5] . . . 61
5.11.2.11 to_hex() [1/5] . . . 61
5.11.2.12 to_hex() [2/5] . . . 62
5.11.2.13 to_hex() [3/5] . . . 62
5.11.2.14 to_hex() [4/5] . . . 62
5.11.2.15 to_hex() [5/5] . . . 62
5.11.2.16 to_oct() [1/5] . . . 63
5.11.2.17 to_oct() [2/5] . . . 63
5.11.2.18 to_oct() [3/5] . . . 63
5.11.2.19 to_oct() [4/5] . . . 63
5.11.2.20 to_oct() [5/5] . . . 63
5.12 Formatters.h . . . 64

5.13 /home/runner/work/streamout/streamout/libs/core/include/Interface.h File Reference . . . . . 64
5.13.1 Detailed Description . . . . . 64
5.14 Interface.h . . . . . 65
5.15 /home/runner/work/streamout/streamout/libs/core/include/RawBufferNavigator.h File Reference . . . 65
5.15.1 Detailed Description . . . . . 65
5.16 RawBufferNavigator.h . . . . . 65
5.17 /home/runner/work/streamout/streamout/libs/core/include/SDHCAL_buffer_loop.h File Reference . . 66
5.17.1 Detailed Description . . . . . 66
5.18 SDHCAL_buffer_loop.h . . . . . 67
5.19 /home/runner/work/streamout/streamout/libs/core/include/SDHCAL_buffer_LoopCounter.h File Reference . . . . . 68
5.19.1 Detailed Description . . . . . 68
5.20 SDHCAL_buffer_LoopCounter.h . . . . . 69
5.21 /home/runner/work/streamout/streamout/libs/core/include/Timer.h File Reference . . . . . 69
5.21.1 Detailed Description . . . . . 69
5.22 Timer.h . . . . . 69
5.23 /home/runner/work/streamout/streamout/libs/core/include/Words.h File Reference . . . . . 70
5.23.1 Detailed Description . . . . . 70
5.23.2 Enumeration Type Documentation . . . . . 70
5.23.2.1 DU . . . . . 70
5.24 Words.h . . . . . 71
5.25 /home/runner/work/streamout/streamout/libs/core/src/Bits.cc File Reference . . . . . 72
5.25.1 Detailed Description . . . . . 72
5.25.2 Function Documentation . . . . . 72
5.25.2.1 operator<<() . . . . . 72
5.26 Bits.cc . . . . . 72
5.27 /home/runner/work/streamout/streamout/libs/core/src/Buffer.cc File Reference . . . . . 72
5.28 Buffer.cc . . . . . 73
5.29 /home/runner/work/streamout/streamout/libs/core/src/DIFSlowControl.cc File Reference . . . . . 73
5.29.1 Detailed Description . . . . . 73
5.30 DIFSlowControl.cc . . . . . 73
5.31 /home/runner/work/streamout/streamout/libs/core/src/DIFUnpacker.cc File Reference . . . . . 76
5.31.1 Detailed Description . . . . . 76
5.32 DIFUnpacker.cc . . . . . 77
5.33 /home/runner/work/streamout/streamout/libs/core/src/Formatters.cc File Reference . . . . . 79
5.33.1 Detailed Description . . . . . 80
5.33.2 Function Documentation . . . . . 80
5.33.2.1 to_bin() [1/5] . . . . . 80
5.33.2.2 to_bin() [2/5] . . . . . 80
5.33.2.3 to_bin() [3/5] . . . . . 81
5.33.2.4 to_bin() [4/5] . . . . . 81
5.33.2.5 to_bin() [5/5] . . . . . 81

5.33.2.6 to_dec() [1/5] . . . 81
5.33.2.7 to_dec() [2/5] . . . 82
5.33.2.8 to_dec() [3/5] . . . 82
5.33.2.9 to_dec() [4/5] . . . 82
5.33.2.10 to_dec() [5/5] . . . 82
5.33.2.11 to_hex() [1/5] . . . 83
5.33.2.12 to_hex() [2/5] . . . 83
5.33.2.13 to_hex() [3/5] . . . 83
5.33.2.14 to_hex() [4/5] . . . 83
5.33.2.15 to_hex() [5/5] . . . 83
5.33.2.16 to_oct() [1/5] . . . 84
5.33.2.17 to_oct() [2/5] . . . 84
5.33.2.18 to_oct() [3/5] . . . 84
5.33.2.19 to_oct() [4/5] . . . 84
5.33.2.20 to_oct() [5/5] . . . 84
5.34 Formatters.cc . . . 85
5.35 /home/runner/work/streamout/streamout/libs/core/src/RawBufferNavigator.cc File Reference . . . 86
5.35.1 Detailed Description . . . 86
5.36 RawBufferNavigator.cc . . . 86
5.37 /home/runner/work/streamout/streamout/libs/core/src/SDHCAL_buffer_LoopCounter.cc File Reference 87
5.37.1 Detailed Description . . . 88
5.38 SDHCAL_buffer_LoopCounter.cc . . . 88
5.39 /home/runner/work/streamout/streamout/libs/interface/Dump/include/textDump.h File Reference . . 88
5.39.1 Detailed Description . . . 89
5.40 textDump.h . . . 89
5.41 /home/runner/work/streamout/streamout/libs/interface/Dump/src/textDump.cc File Reference . . . 89
5.41.1 Detailed Description . . . 89
5.42 textDump.cc . . . 90
5.43 /home/runner/work/streamout/streamout/libs/interface/RawDataReader/include/RawdataReader.h File Reference . . . 90
5.43.1 Detailed Description . . . 90
5.44 RawdataReader.h . . . 91
5.45 /home/runner/work/streamout/streamout/libs/interface/RawDataReader/src/RawdataReader.cc File Reference . . . 91
5.45.1 Detailed Description . . . 91
5.46 RawdataReader.cc . . . 92
5.47 /home/runner/work/streamout/streamout/libs/interface/ROOT/include/ROOTtreeDest.h File Reference 93
5.47.1 Detailed Description . . . 93
5.48 ROOTtreeDest.h . . . 94
5.49 /home/runner/work/streamout/streamout/libs/interface/ROOT/src/ROOTtreeDest.cc File Reference . 94
5.49.1 Detailed Description . . . 94
5.50 ROOTtreeDest.cc . . . 94

# Chapter 1

# Hierarchical Index

## 1.1 Class Hierarchy

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


Buffer . . . 7
ROOTtreeDest::DATA . . . 10
DIFPtr . . . 13
DIFSlowControl . . . 19
DIFUnpacker . . . 22
Interface . . . 30
  ROOTtreeDest . . . 39
  RawdataReader . . . 36
  textDump . . . 47
RawBufferNavigator . . . 31
SDHCAL_buffer_loop< SOURCE, DESTINATION > . . . 42
SDHCAL_buffer_LoopCounter . . . 44
Timer . . . 50

# Chapter 2

# Class Index

## 2.1 Class List

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


Buffer . . . . . 7
ROOTtreeDest::DATA . . . . . 10
DIFPtr . . . . . 13
DIFSlowControl
    Handler of DIF Slow Control info . . . . . 19
DIFUnpacker . . . . . 22
Interface
    Template class should implement void SOURCE::start(); bool SOURCE::next(); void SOURCE↩
    ::end(); const Buffer& SOURCE::getSDHCALBuffer(); . . . . . 30
RawBufferNavigator . . . . . 31
RawdataReader . . . . . 36
ROOTtreeDest . . . . . 39
SDHCAL_buffer_loop< SOURCE, DESTINATION > . . . . . 42
SDHCAL_buffer_LoopCounter . . . . . 44
textDump . . . . . 47
Timer . . . . . 50

# Chapter 3

# File Index

## 3.1 File List

Here is a list of all files with brief descriptions:


/home/runner/work/streamout/streamout/libs/core/include/Bits.h . . . . . 51
/home/runner/work/streamout/streamout/libs/core/include/Buffer.h . . . . . 53
/home/runner/work/streamout/streamout/libs/core/include/DIFPtr.h . . . . . 54
/home/runner/work/streamout/streamout/libs/core/include/DIFSlowControl.h . . . . . 56
/home/runner/work/streamout/streamout/libs/core/include/DIFUnpacker.h . . . . . 57
/home/runner/work/streamout/streamout/libs/core/include/Formatters.h . . . . . 58
/home/runner/work/streamout/streamout/libs/core/include/Interface.h . . . . . 64
/home/runner/work/streamout/streamout/libs/core/include/RawBufferNavigator.h . . . . . 65
/home/runner/work/streamout/streamout/libs/core/include/SDHCAL_buffer_loop.h . . . . . 66
/home/runner/work/streamout/streamout/libs/core/include/SDHCAL_buffer_LoopCounter.h . . . . . 68
/home/runner/work/streamout/streamout/libs/core/include/Timer.h . . . . . 69
/home/runner/work/streamout/streamout/libs/core/include/Words.h . . . . . 70
/home/runner/work/streamout/streamout/libs/core/src/Bits.cc . . . . . 72
/home/runner/work/streamout/streamout/libs/core/src/Buffer.cc . . . . . 72
/home/runner/work/streamout/streamout/libs/core/src/DIFSlowControl.cc . . . . . 73
/home/runner/work/streamout/streamout/libs/core/src/DIFUnpacker.cc . . . . . 76
/home/runner/work/streamout/streamout/libs/core/src/Formatters.cc . . . . . 79
/home/runner/work/streamout/streamout/libs/core/src/RawBufferNavigator.cc . . . . . 86
/home/runner/work/streamout/streamout/libs/core/src/SDHCAL_buffer_LoopCounter.cc . . . . . 87
/home/runner/work/streamout/streamout/libs/interface/Dump/include/textDump.h . . . . . 88
/home/runner/work/streamout/streamout/libs/interface/Dump/src/textDump.cc . . . . . 89
/home/runner/work/streamout/streamout/libs/interface/RawDataReader/include/RawdataReader.h . . . . . 90
/home/runner/work/streamout/streamout/libs/interface/RawDataReader/src/RawdataReader.cc . . . . . 91
/home/runner/work/streamout/streamout/libs/interface/ROOT/include/ROOTtreeDest.h . . . . . 93
/home/runner/work/streamout/streamout/libs/interface/ROOT/src/ROOTtreeDest.cc . . . . . 94

# Chapter 4

# Class Documentation

## 4.1 Buffer Class Reference

`#include <Buffer.h>`

### Public Member Functions

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

### 4.1.1 Detailed Description

Definition at line 13 of file Buffer.h.

### 4.1.2 Constructor & Destructor Documentation

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

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

Definition at line 16 of file Buffer.h.

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

#### 4.1.2.2 Buffer() [2/5]

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

Definition at line 17 of file Buffer.h.

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

#### 4.1.2.3 Buffer() [3/5]

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

Definition at line 18 of file Buffer.h.

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

#### 4.1.2.4 Buffer() [4/5]

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

Definition at line 19 of file Buffer.h.

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

#### 4.1.2.5 Buffer() [5/5]

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

Definition at line 20 of file Buffer.h.

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

#### 4.1.2.6 ~Buffer()

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

Definition at line 8 of file Buffer.cc.

```
00008 {}
```

### 4.1.3 Member Function Documentation

#### 4.1.3.1 begin()

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

Definition at line 26 of file Buffer.h.

```
00026 { return m_Buffer; }
```

#### 4.1.3.2 capacity()

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

Definition at line 23 of file Buffer.h.

```
00023 { return m_Capacity; }
```

#### 4.1.3.3 end()

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

Definition at line 27 of file Buffer.h.

```
00027 { return m_Buffer + m_Size; }
```

#### 4.1.3.4 operator[]() [1/2]

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

Definition at line 28 of file Buffer.h.

```
00028 { return m_Buffer[pos]; }
```

#### 4.1.3.5 operator[]() [2/2]

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

Definition at line 29 of file Buffer.h.

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

#### 4.1.3.6 set()

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

Definition at line 25 of file Buffer.h.

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

#### 4.1.3.7 setSize()

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

Definition at line 31 of file Buffer.h.

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

#### 4.1.3.8 size()

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

Definition at line 22 of file Buffer.h.

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

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

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

## 4.2 ROOTtreeDest::DATA Struct Reference

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

## Public Attributes

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

### 4.2.1 Detailed Description

Definition at line 16 of file ROOTtreeDest.h.

### 4.2.2 Member Data Documentation

#### 4.2.2.1 AbsoluteBCID

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

Definition at line 21 of file ROOTtreeDest.h.

#### 4.2.2.2 ASICid

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

Definition at line 18 of file ROOTtreeDest.h.

#### 4.2.2.3 CHANNELid

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

Definition at line 18 of file ROOTtreeDest.h.

#### 4.2.2.4 DIF_BCID

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

Definition at line 20 of file ROOTtreeDest.h.

#### 4.2.2.5 DIFid

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

Definition at line 18 of file ROOTtreeDest.h.

#### 4.2.2.6 DTC

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

Definition at line 20 of file ROOTtreeDest.h.

#### 4.2.2.7 frame_BCID

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

Definition at line 20 of file ROOTtreeDest.h.

#### 4.2.2.8 GTC

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

Definition at line 20 of file ROOTtreeDest.h.

#### 4.2.2.9 Thresh

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

Definition at line 19 of file ROOTtreeDest.h.

#### 4.2.2.10 timeStamp

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

Definition at line 20 of file ROOTtreeDest.h.

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

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

## 4.3 DIFPtr Class Reference

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

### Public Member Functions

- void setBuffer (unsigned char *p, const std::uint32_t &max_size)
- 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 (uint32_t line) 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 (uint32_t i) const
- std::uint32_t getFrameAsicHeader (uint32_t i) const
- std::uint32_t getFrameBCID (uint32_t i) const
- std::uint32_t getFrameTimeToTrigger (uint32_t i) const
- bool getFrameLevel (uint32_t i, uint32_t ipad, uint32_t ilevel) const
- uint32_t getDIFid () const
- uint32_t getASICid (uint32_t i) const
- uint32_t getThresholdStatus (uint32_t i, uint32_t ipad) const

### 4.3.1 Detailed Description

Definition at line 14 of file DIFPtr.h.

### 4.3.2 Member Function Documentation

#### 4.3.2.1 getAbsoluteBCID()

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

Definition at line 79 of file DIFPtr.h.

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

#### 4.3.2.2 getASICid()

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

Definition at line 106 of file DIFPtr.h.

```
00106 { return getFrameAsicHeader(i) & 0xFF; }
```

#### 4.3.2.3 getBCID()

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

Definition at line 80 of file DIFPtr.h.

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

#### 4.3.2.4 getDIFid()

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

Definition at line 105 of file DIFPtr.h.

```
00105 { return getID() & 0xFF; }
```

#### 4.3.2.5 getDTC()

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

Definition at line 77 of file DIFPtr.h.

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

#### 4.3.2.6 getFrameAsicHeader()

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

Definition at line 93 of file DIFPtr.h.

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

#### 4.3.2.7 getFrameBCID()

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

Definition at line 94 of file DIFPtr.h.

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

#### 4.3.2.8 getFrameLevel()

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

Definition at line 96 of file DIFPtr.h.

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

#### 4.3.2.9 getFramePtr()

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

Definition at line 92 of file DIFPtr.h.

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

#### 4.3.2.10 getFramesVector()

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

Definition at line 74 of file DIFPtr.h.

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

#### 4.3.2.11 getFrameTimeToTrigger()

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

Definition at line 95 of file DIFPtr.h.

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

#### 4.3.2.12 getGetFramePtrReturn()

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

Definition at line 73 of file DIFPtr.h.

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

#### 4.3.2.13 getGTC()

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

Definition at line 78 of file DIFPtr.h.

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

#### 4.3.2.14 getID()

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

Definition at line 76 of file DIFPtr.h.

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

#### 4.3.2.15 getLines()

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

Definition at line 81 of file DIFPtr.h.

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

#### 4.3.2.16 getLinesVector()

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

Definition at line 75 of file DIFPtr.h.

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

#### 4.3.2.17 getNumberOfFrames()

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

Definition at line 91 of file DIFPtr.h.

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

#### 4.3.2.18 getPtr()

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

Definition at line 72 of file DIFPtr.h.

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

#### 4.3.2.19 getTASU1()

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

Definition at line 83 of file DIFPtr.h.

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

#### 4.3.2.20 getTASU2()

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

Definition at line 84 of file DIFPtr.h.

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

#### 4.3.2.21 getTDIF()

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

Definition at line 85 of file DIFPtr.h.

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

#### 4.3.2.22 getTemperatureASU1()

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

Definition at line 87 of file DIFPtr.h.

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

#### 4.3.2.23 getTemperatureASU2()

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

Definition at line 88 of file DIFPtr.h.

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

#### 4.3.2.24 getTemperatureDIF()

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

Definition at line 86 of file DIFPtr.h.

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

#### 4.3.2.25 getThresholdStatus()

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

Definition at line 107 of file DIFPtr.h.

```
00107 { return (((uint32_t)getFrameLevel(i, ipad, 1)) << 1) | ((uint32_t)getFrameLevel(i, ipad, 0)); }
```

#### 4.3.2.26 hasAnalogReadout()

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

Definition at line 90 of file DIFPtr.h.

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

#### 4.3.2.27 hasLine()

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

Definition at line 82 of file DIFPtr.h.

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

#### 4.3.2.28 hasTemperature()

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

Definition at line 89 of file DIFPtr.h.

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

#### 4.3.2.29 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:

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

## 4.4 DIFSlowControl Class Reference

Handler of DIF Slow Control info.

`#include <DIFSlowControl.h>`

### Public Member Functions

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

### 4.4.1 Detailed Description

Handler of DIF Slow Control info.

**Author**

L.Mirabito

**Date**

March 2010

**Version**

1.0

Definition at line 19 of file DIFSlowControl.h.

### 4.4.2 Constructor & Destructor Documentation

#### 4.4.2.1 DIFSlowControl()

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

Constructor.

**Parameters**

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

Definition at line 10 of file DIFSlowControl.cc.

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

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

### 4.4.3 Member Function Documentation

#### 4.4.3.1 Dump()

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

print out full map

Definition at line 45 of file DIFSlowControl.cc.

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

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

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

Get one chip map.

**Parameters**

| | |
|---|---|
| *asicid* | ASIC ID |

**Returns**

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

Definition at line 41 of file DIFSlowControl.cc.

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

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

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

Get one Chip value.

**Parameters**

| | |
|---|---|
| *asicid* | ASic ID |
| *param* | Parameter name |

Definition at line 43 of file DIFSlowControl.cc.

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

#### 4.4.3.4 getChipsMap()

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

Get chips map.

**Returns**

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

Definition at line 39 of file DIFSlowControl.cc.

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

#### 4.4.3.5 getDIFId()

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

get DIF id

Definition at line 37 of file DIFSlowControl.cc.

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

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

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

## 4.5 DIFUnpacker Class Reference

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

## Static Public Member Functions

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

### 4.5.1 Detailed Description

Definition at line 10 of file DIFUnpacker.h.

### 4.5.2 Member Function Documentation

#### 4.5.2.1 dumpFrameOld()

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

Definition at line 146 of file DIFUnpacker.cc.

```
00147 {
00148   bool         PAD[128];
00149   bool         l0[64];
00150   bool         l1[64];
00151   std::uint8_t un{1};
00152   for(std::size_t ip = 0; ip < 128; ip++) { PAD[ip] = false; }  // init PADs
00153   std::uint32_t idx1{4};
00154   for(int ik = 0; ik < 4; ik++)
00155   {
00156     std::uint32_t PadEtat{swap_bytes(&buf[idx1])};
00157     idx1 += 4;
00158     for(int e = 0; e < 32; e++)
00159     {
00160       PAD[((3 - ik) * 32) + (31 - e)] = PadEtat & un;  // binary operation
```

```
00161       PadEtat                         = PadEtat » 1;  // décalage des bit de 1
00162     }
00163   }
00164   // fill bool arrays
00165   for(int p = 0; p < 64; p++)
00166   {
00167     l0[p] = static_cast<bool>(PAD[(2 * p)]);      // _Lev0 (PAD paire)
00168     l1[p] = static_cast<bool>(PAD[(2 * p) + 1]);  // _Lev1 (PAD impaires)
00169   }
00170   std::bitset<64> bs0(0);
00171   std::bitset<64> bs1(0);
00172   for(std::uint32_t ip = 0; ip < 64; ip++)
00173   {
00174     bs0.set(ip, l0[ip]);
00175     bs1.set(ip, l1[ip]);
00176   }
00177   std::cout « "\t \t" « bs0 « std::endl;
00178   std::cout « "\t \t" « bs1 « std::endl;
00179 }
```

#### 4.5.2.2 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::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 }
```

#### 4.5.2.3 getAnalogPtr()

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

Definition at line 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 }
```

#### 4.5.2.4 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]; }
```

#### 4.5.2.5 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]; }
```

#### 4.5.2.6 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]); }
```

#### 4.5.2.7 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 }
```

#### 4.5.2.8 getFrameLevel()

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

Definition at line 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); }
```

#### 4.5.2.9 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 }
```

#### 4.5.2.10 getFramePtr()

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

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

#### 4.5.2.11 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]; }
```

#### 4.5.2.12 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]; }
```

#### 4.5.2.13 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; }
```

#### 4.5.2.14 getStartOfDIF()

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

Definition at line 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 }
```

#### 4.5.2.15 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]; }
```

#### 4.5.2.16 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]; }
```

#### 4.5.2.17 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]); }
```

#### 4.5.2.18 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 }
```

#### 4.5.2.19 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); }
```

#### 4.5.2.20 hasLine()

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

Definition at line 64 of file DIFUnpacker.cc.

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

#### 4.5.2.21 hasTemperature()

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

Definition at line 72 of file DIFUnpacker.cc.

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

#### 4.5.2.22 swap_bytes()

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

Definition at line 181 of file DIFUnpacker.cc.

```
00182 {
00183   unsigned char Swapped[4];
00184   for(std::size_t i = 0; i < 4; i++) Swapped[i] = buf[4 - 1 - i];
00185   return *reinterpret_cast<std::uint32_t*>(&Swapped[0]);
00186 }
```

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

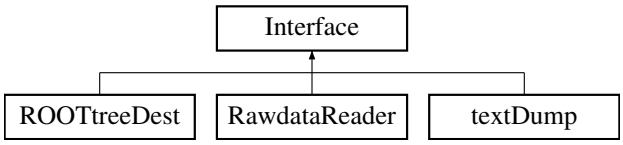
- /home/runner/work/streamout/streamout/libs/core/include/DIFUnpacker.h
- /home/runner/work/streamout/streamout/libs/core/src/DIFUnpacker.cc

## 4.6 Interface Class Reference

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

`#include <Interface.h>`

Inheritance diagram for Interface:




### Public Member Functions

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

### 4.6.1 Detailed Description

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

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

Definition at line 26 of file Interface.h.

### 4.6.2 Constructor & Destructor Documentation

#### 4.6.2.1 Interface()

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

Definition at line 29 of file Interface.h.

```
00029 {}
```

#### 4.6.2.2 ~Interface()

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

Definition at line 30 of file Interface.h.

```
00030 {}
```

### 4.6.3 Member Function Documentation

#### 4.6.3.1 log()

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

Definition at line 31 of file Interface.h.

```
00031 { return m_Logger; }
```

#### 4.6.3.2 setLogger()

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

Definition at line 32 of file Interface.h.

```
00032 { m_Logger = logger; }
```

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

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

## 4.7 RawBufferNavigator Class Reference

```
#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)
- 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)

### 4.7.1 Detailed Description

Definition at line 12 of file RawBufferNavigator.h.

### 4.7.2 Constructor & Destructor Documentation

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

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

#### 4.7.2.2 ~RawBufferNavigator()

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

#### 4.7.2.3 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)
00017 {
00018   StartAt(start);
00019   m_DIFstartIndex = DIFUnpacker::getStartOfDIF(m_Buffer.begin(), m_Buffer.size(), m_Start);
00020 }
```

### 4.7.3 Member Function Documentation

#### 4.7.3.1 badSCData()

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

Definition at line 59 of file RawBufferNavigator.cc.

```
00060 {
00061   setSCBuffer();
00062   return m_BadSCdata;
00063 }
```

#### 4.7.3.2 getDIF_CRC()

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

Definition at line 42 of file RawBufferNavigator.cc.

```
00043 {
00044   uint32_t i{getEndOfDIFData()};
00045   uint32_t ret{0};
00046   ret |= ((m_Buffer.begin()[i - 2]) « 8);
00047   ret |= m_Buffer.begin()[i - 1];
00048   return ret;
00049 }
```

#### 4.7.3.3 getDIFBuffer()

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

Definition at line 30 of file RawBufferNavigator.cc.

```
00030 { return Buffer(getDIFBufferStart(), getDIFBufferSize()); }
```

#### 4.7.3.4 getDIFBufferSize()

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

Definition at line 28 of file RawBufferNavigator.cc.

```
00028 { return m_Buffer.size() - m_DIFstartIndex; }
```

#### 4.7.3.5 getDIFBufferStart()

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

Definition at line 26 of file RawBufferNavigator.cc.

```
00026 { return &(m_Buffer.begin()[m_DIFstartIndex]); }
```

#### 4.7.3.6 getDIFPtr()

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

Definition at line 32 of file RawBufferNavigator.cc.

```
00033 {
00034   m_TheDIFPtr.setBuffer(getDIFBufferStart(), getDIFBufferSize());
00035   return m_TheDIFPtr;
00036 }
```

#### 4.7.3.7 getEndOfAllData()

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

Definition at line 98 of file RawBufferNavigator.cc.

```
00099 {
00100   setSCBuffer();
00101   if(hasSlowControlData() && !m_BadSCdata) { return Buffer(&(m_SCbuffer.begin()[m_SCbuffer.size()]),
      getSizeAfterDIFPtr() - 3 - m_SCbuffer.size()); }
00102   else
00103     return Buffer(&(getDIFBufferStart()[getEndOfDIFData()]), getSizeAfterDIFPtr() - 3);  // remove the
      2 bytes for CRC and the DIF trailer
00104 }
```

#### 4.7.3.8 getEndOfDIFData()

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

Definition at line 38 of file RawBufferNavigator.cc.

```
00038 { return getDIFPtr().getGetFramePtrReturn() + 3; }
```

#### 4.7.3.9 getSCBuffer()

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

Definition at line 53 of file RawBufferNavigator.cc.

```
00054 {
00055   setSCBuffer();
00056   return m_SCbuffer;
00057 }
```

#### 4.7.3.10 getSizeAfterDIFPtr()

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

Definition at line 40 of file RawBufferNavigator.cc.

```
00040 { return getDIFBufferSize() - getDIFPtr().getGetFramePtrReturn(); }
```

#### 4.7.3.11 getStartOfDIF()

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

Definition at line 24 of file RawBufferNavigator.cc.

```
00024 { return m_DIFstartIndex; }
```

#### 4.7.3.12 hasSlowControlData()

```
bool RawBufferNavigator::hasSlowControlData ( )
```

Definition at line 51 of file RawBufferNavigator.cc.

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

#### 4.7.3.13 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   }
```

#### 4.7.3.14 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 }
```

#### 4.7.3.15 validBuffer()

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

Definition at line 22 of file RawBufferNavigator.cc.

```
00022 { return m_DIFstartIndex != 0; }
```

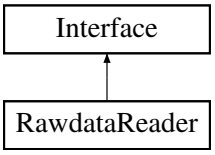
The documentation for this class was generated from the following files:

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

## 4.8 RawdataReader Class Reference

`#include <RawdataReader.h>`

Inheritance diagram for RawdataReader:




### Public Member Functions

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

### Static Public Member Functions

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

### 4.8.1 Detailed Description

Definition at line 17 of file RawdataReader.h.

### 4.8.2 Constructor & Destructor Documentation

#### 4.8.2.1 RawdataReader()

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

Definition at line 16 of file RawdataReader.cc.

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

#### 4.8.2.2 ~RawdataReader()

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

Definition at line 29 of file RawdataReader.h.

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

### 4.8.3 Member Function Documentation

#### 4.8.3.1 closeFile()

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

Definition at line 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 }
```

#### 4.8.3.2 end()

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

Definition at line 24 of file RawdataReader.cc.

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

#### 4.8.3.3 getFileSize()

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

Definition at line 124 of file RawdataReader.cc.

```
00124 { return m_FileSize; }
```

#### 4.8.3.4 getSDHCALBuffer()

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

Definition at line 116 of file RawdataReader.cc.

```
00117 {
00118   uncompress();
00119   return m_Buffer;
00120 }
```

#### 4.8.3.5 nextDIFbuffer()

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

Definition at line 90 of file RawdataReader.cc.

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

#### 4.8.3.6 nextEvent()

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

Definition at line 76 of file RawdataReader.cc.

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

#### 4.8.3.7 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(), e.code().value());
00072     throw;
00073   }
00074 }
```

#### 4.8.3.8 setDefaultBufferSize()

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

Definition at line 14 of file RawdataReader.cc.

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

#### 4.8.3.9 start()

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

Definition at line 22 of file RawdataReader.cc.

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

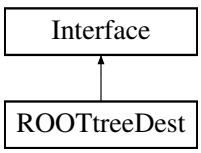
The documentation for this class was generated from the following files:

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

## 4.9 ROOTtreeDest Class Reference

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

Inheritance diagram for ROOTtreeDest:




### Classes

- struct DATA

### Public Member Functions

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

### 4.9.1 Detailed Description

Definition at line 13 of file ROOTtreeDest.h.

### 4.9.2 Constructor & Destructor Documentation

#### 4.9.2.1 ROOTtreeDest()

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

Definition at line 8 of file ROOTtreeDest.cc.

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

### 4.9.3 Member Function Documentation

#### 4.9.3.1 end()

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

Definition at line 31 of file ROOTtreeDest.h.

```
00031 { ; }
```

#### 4.9.3.2 processDIF()

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

Definition at line 25 of file ROOTtreeDest.cc.

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

#### 4.9.3.3 processFrame()

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

Definition at line 34 of file ROOTtreeDest.cc.

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

#### 4.9.3.4 processPadInFrame()

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

Definition at line 41 of file ROOTtreeDest.cc.

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

#### 4.9.3.5 processSlowControl()

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

Definition at line 30 of file ROOTtreeDest.h.

```
00030 { ; }
```

#### 4.9.3.6 start()

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

Definition at line 23 of file ROOTtreeDest.cc.

```
00023 { dataReset(); }
```

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

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

## 4.10 SDHCAL_buffer_loop< SOURCE, DESTINATION > Class Template Reference

`#include <SDHCAL_buffer_loop.h>`

### Public Member Functions

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

### 4.10.1 Detailed Description

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

Definition at line 21 of file SDHCAL_buffer_loop.h.

### 4.10.2 Constructor & Destructor Documentation

#### 4.10.2.1 SDHCAL_buffer_loop()

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

Definition at line 24 of file SDHCAL_buffer_loop.h.

```
00024                                                                    : m_Source(source),
      m_Destination(dest), m_Debug(debug)
00025   {
00026     m_Logger = spdlog::create<spdlog::sinks::null_sink_mt>("streamout");
00027     if(!spdlog::get("streamout")) { spdlog::register_logger(m_Logger); }
00028     m_Source.setLogger(m_Logger);
00029     m_Destination.setLogger(m_Logger);
00030   }
```

### 4.10.3 Member Function Documentation

#### 4.10.3.1 addSink()

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

Definition at line 32 of file SDHCAL_buffer_loop.h.

```
00033   {
00034     sink->set_level(level);
00035     m_Sinks.push_back(sink);
00036     m_Logger = std::make_shared<spdlog::logger>("streamout", begin(m_Sinks), end(m_Sinks));
00037     m_Source.setLogger(m_Logger);
00038     m_Destination.setLogger(m_Logger);
00039   }
```

#### 4.10.3.2 log()

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

Definition at line 112 of file SDHCAL_buffer_loop.h.

```
00112 { return m_Logger; }
```

#### 4.10.3.3 loop()

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

Definition at line 41 of file SDHCAL_buffer_loop.h.

```
00042   {
00043     Timer timer;
00044     timer.start();
00045     m_Source.start();
00046     m_Destination.start();
00047     RawBufferNavigator bufferNavigator;
00048     while(m_Source.nextEvent() && (m_NbrEventsToProcess == 0 || m_NbrEventsToProcess >= m_NbrEvents))
00049     {
00050       m_Logger->warn("===*** Event number {} ***===", m_NbrEvents);
00051       while(m_Source.nextDIFbuffer())
00052       {
00053         const Buffer& buffer           = m_Source.getSDHCALBuffer();
00054         bit8_t*       debug_variable_1 = buffer.end();
00055         bufferNavigator.setBuffer(buffer);
00056         bit8_t* debug_variable_2 = bufferNavigator.getDIFBuffer().end();
00057         m_Logger->info("DIF BUFFER END {} {}", fmt::ptr(debug_variable_1),
      fmt::ptr(debug_variable_2));
00058         if(m_Debug) assert(debug_variable_1 == debug_variable_2);
00059         uint32_t idstart = bufferNavigator.getStartOfDIF();
00060         if(m_Debug && idstart == 0) m_Logger->info(to_hex(buffer));
00061         c.DIFStarter[idstart]++;
00062         if(!bufferNavigator.validBuffer())
00063         {
00064           m_Logger->error("!bufferNavigator.validBuffer()");
00065           continue;
00066         }
00067         DIFPtr& d = bufferNavigator.getDIFPtr();
00068         c.DIFPtrValueAtReturnedPos[bufferNavigator.getDIFBufferStart()[d.getGetFramePtrReturn()]]++;
00069         if(m_Debug) assert(bufferNavigator.getDIFBufferStart()[d.getGetFramePtrReturn()] == 0xa0);
00070         c.SizeAfterDIFPtr[bufferNavigator.getSizeAfterDIFPtr()]++;
00071         m_Destination.processDIF(d);
00072         for(uint32_t i = 0; i < d.getNumberOfFrames(); i++)
00073         {
```

```
00074           m_Destination.processFrame(d, i);
00075           for(uint32_t j = 0; j < 64; j++) m_Destination.processPadInFrame(d, i, j);
00076         }
00077
00078         bool processSC = false;
00079         if(bufferNavigator.hasSlowControlData())
00080         {
00081           c.hasSlowControl++;
00082           processSC = true;
00083         }
00084         if(bufferNavigator.badSCData())
00085         {
00086           c.hasBadSlowControl++;
00087           processSC = false;
00088         }
00089         if(processSC) { m_Destination.processSlowControl(bufferNavigator.getSCBuffer()); }
00090
00091         Buffer eod = bufferNavigator.getEndOfAllData();
00092         c.SizeAfterAllData[eod.size()]++;
00093         bit8_t* debug_variable_3 = eod.end();
00094         m_Logger->info("END DATA BUFFER END {} {}", fmt::ptr(debug_variable_1),
      fmt::ptr(debug_variable_3));
00095         if(m_Debug) assert(debug_variable_1 == debug_variable_3);
00096         if(eod.size() != 0) m_Logger->info("End of Data remaining stuff : {}", to_hex(eod));
00097
00098         int nonzeroCount = 0;
00099         for(unsigned char* it = eod.begin(); it != eod.end(); it++)
00100           if(static_cast<int>(*it) != 0) nonzeroCount++;
00101         c.NonZeroValusAtEndOfData[nonzeroCount]++;
00102       }  // end of DIF while loop
00103       m_Logger->warn("***=== Event number {} ===***", m_NbrEvents);
00104       m_NbrEvents++;
00105     }  // end of event while loop
00106     m_Destination.end();
00107     m_Source.end();
00108     timer.stop();
00109     fmt::print("=== elapsed time {}ms ({}ms/event) ===\n", timer.getElapsedTime() / 1000,
      timer.getElapsedTime() / (1000 * m_NbrEvents));
00110   }
```

#### 4.10.3.4 printAllCounters()

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

Definition at line 111 of file SDHCAL_buffer_loop.h.

```
00111 { c.printAllCounters(); }
```

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

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

## 4.11 SDHCAL_buffer_LoopCounter Struct Reference

```
#include <SDHCAL_buffer_LoopCounter.h>
```

### Public Member Functions

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

### 4.11.1 Detailed Description

Definition at line 11 of file SDHCAL_buffer_LoopCounter.h.

### 4.11.2 Member Function Documentation

#### 4.11.2.1 printAllCounters()

```
void SDHCAL_buffer_LoopCounter::printAllCounters ( )
```

Definition at line 9 of file SDHCAL_buffer_LoopCounter.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 }
```

#### 4.11.2.2 printCounter()

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

Definition at line 20 of file SDHCAL_buffer_LoopCounter.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 }
```

### 4.11.3 Member Data Documentation

#### 4.11.3.1 DIFPtrValueAtReturnedPos

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

Definition at line 17 of file SDHCAL_buffer_LoopCounter.h.

#### 4.11.3.2 DIFStarter

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

Definition at line 16 of file SDHCAL_buffer_LoopCounter.h.

#### 4.11.3.3 hasBadSlowControl

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

Definition at line 15 of file SDHCAL_buffer_LoopCounter.h.

#### 4.11.3.4 hasSlowControl

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

Definition at line 14 of file SDHCAL_buffer_LoopCounter.h.

#### 4.11.3.5 NonZeroValusAtEndOfData

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

Definition at line 20 of file SDHCAL_buffer_LoopCounter.h.

#### 4.11.3.6 SizeAfterAllData

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

Definition at line 19 of file SDHCAL_buffer_LoopCounter.h.

#### 4.11.3.7 SizeAfterDIFPtr

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

Definition at line 18 of file SDHCAL_buffer_LoopCounter.h.

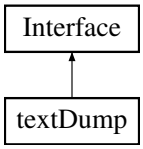
The documentation for this struct was generated from the following files:

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

## 4.12 textDump Class Reference

```
#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)

### 4.12.1 Detailed Description

Definition at line 15 of file textDump.h.

### 4.12.2 Constructor & Destructor Documentation

#### 4.12.2.1 textDump()

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

Definition at line 18 of file textDump.h.

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

### 4.12.3 Member Function Documentation

#### 4.12.3.1 end()

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

Definition at line 40 of file textDump.cc.

```
00040 { print()->info("textDump end of report"); }
```

#### 4.12.3.2 print()

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

Definition at line 29 of file textDump.h.

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

#### 4.12.3.3 processDIF()

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

Definition at line 11 of file textDump.cc.

```
00012 {
00013   print()->info("DIF number is {}", d.getDIFid());
00014   print()->info("DTC value is {}", d.getDTC());
00015   print()->info("GTC value is {}", d.getGTC());
00016   print()->info("DIF BCID is {}", d.getBCID());
00017   print()->info("Absolute BCID is {}", d.getAbsoluteBCID());
00018   print()->info("The number of frame is {}", d.getNumberOfFrames());
00019 }
```

#### 4.12.3.4 processFrame()

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

Definition at line 21 of file textDump.cc.

```
00022 {
00023   print()->info("Displaying frame number {}", frameIndex);
00024   print()->info("ASIC ID is {}", d.getASICid(frameIndex));
00025   print()->info("Frame BCID is {}", d.getFrameBCID(frameIndex));
00026   print()->info("Frame Time To Trigger (a.k.a timestamp) is {}", d.getFrameTimeToTrigger(frameIndex));
00027 }
```

#### 4.12.3.5 processPadInFrame()

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

Definition at line 29 of file textDump.cc.

```
00030 {
00031   if(d.getThresholdStatus(frameIndex, channelIndex) > 0)
00032   {
00033     print()->info("Displaying channel number {}", channelIndex);
00034     print()->info("Threshold status is {}", d.getThresholdStatus(frameIndex, channelIndex));
00035   }
00036 }
```

#### 4.12.3.6 processSlowControl()

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

Definition at line 38 of file textDump.cc.

```
00038 { print()->error("textDump::processSlowControl not implemented yet."); }
```

#### 4.12.3.7 setLevel()

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

Definition at line 30 of file textDump.h.

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

#### 4.12.3.8 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:

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

## 4.13 Timer Class Reference

```
#include <Timer.h>
```

### Public Member Functions

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

### 4.13.1 Detailed Description

Definition at line 10 of file Timer.h.

### 4.13.2 Member Function Documentation

#### 4.13.2.1 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(); }
```

#### 4.13.2.2 start()

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

Definition at line 13 of file Timer.h.

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

#### 4.13.2.3 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:

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

# Chapter 5

# File Documentation

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

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

### Typedefs

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

### Functions

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

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

### 5.1.1 Detailed Description

**Copyright**

2022 G.Grenier F.Lagarde

Definition in file Bits.h.

### 5.1.2 Typedef Documentation

#### 5.1.2.1 bit16_t

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

Definition at line 11 of file Bits.h.

#### 5.1.2.2 bit32_t

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

Definition at line 12 of file Bits.h.

#### 5.1.2.3 bit64_t

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

Definition at line 13 of file Bits.h.

#### 5.1.2.4 bit8_t

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

Definition at line 10 of file Bits.h.

### 5.1.3 Function Documentation

#### 5.1.3.1 operator<<()

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

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

Definition at line 8 of file Bits.cc.

```
00008 { return os « c + 0; }
```

## 5.2 Bits.h

Go to the documentation of this file.

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

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

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

### Classes

- class Buffer

### 5.3.1 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   Buffer(const bit8_t b[], const std::size_t& i) : m_Buffer(const_cast<bit8_t*>(&b[0])), m_Size(i),
        m_Capacity(i) {}
00018   Buffer(const char b[], const std::size_t& i) : m_Buffer(const_cast<bit8_t*>(reinterpret_cast<const
        bit8_t*>(&b[0]))), m_Size(i), m_Capacity(i) {}
00019   template<typename T> Buffer(const std::vector<T>& rawdata) :
        m_Buffer(const_cast<bit8_t*>(reinterpret_cast<const bit8_t*>(rawdata.data()))), m_Size(rawdata.size()
        * sizeof(T)), m_Capacity(rawdata.capacity() * sizeof(T)) {}
00020   template<typename T, std::size_t N> Buffer(const std::array<T, N>& rawdata) :
        m_Buffer(const_cast<bit8_t*>(reinterpret_cast<const bit8_t*>(rawdata.data()))), m_Size(rawdata.size()
        * sizeof(T)), m_Capacity(rawdata.size() * sizeof(T)) {}
00021
00022   std::size_t size() const { return m_Size; }
00023   std::size_t capacity() const { return m_Capacity; }
00024
00025   void    set(unsigned char* b) { m_Buffer = b; }
00026   bit8_t* begin() const { return m_Buffer; }
00027   bit8_t* end() const { return m_Buffer + m_Size; }
00028   bit8_t& operator[](const std::size_t& pos) { return m_Buffer[pos]; }
00029   bit8_t& operator[](const std::size_t& pos) const { return m_Buffer[pos]; }
00030
00031   void setSize(const std::size_t& size) { m_Size = size; }
00032   virtual ~Buffer();
00033
00034 private:
00035   bit8_t*     m_Buffer{nullptr};
00036   std::size_t m_Size{0};
00037   std::size_t m_Capacity{0};
00038 };
```

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

```
#include "DIFUnpacker.h"
#include <cstdint>
#include <spdlog/spdlog.h>
#include <string>
#include <vector>
```

### Classes

- class DIFPtr

### 5.5.1 Detailed Description

**Copyright**

2022 G.Grenier F.Lagarde

Definition in file DIFPtr.h.

## 5.6 DIFPtr.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
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* p, const std::uint32_t& max_size);
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(uint32_t line) 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(uint32_t i) const;
00039   std::uint32_t                 getFrameAsicHeader(uint32_t i) const;
00040   std::uint32_t                 getFrameBCID(uint32_t i) const;
00041   std::uint32_t                 getFrameTimeToTrigger(uint32_t i) const;
00042   bool                          getFrameLevel(uint32_t i, uint32_t ipad, uint32_t ilevel) const;
00043   // Addition by GG
00044   uint32_t                      getDIFid() const;
00045   uint32_t                      getASICid(uint32_t i) const;
00046   uint32_t                      getThresholdStatus(uint32_t i, uint32_t ipad) 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(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(uint32_t i) const { return theFrames_[i]; }
00093 inline std::uint32_t              DIFPtr::getFrameAsicHeader(uint32_t i) const { return
       DIFUnpacker::getFrameAsicHeader(theFrames_[i]); }
00094 inline std::uint32_t              DIFPtr::getFrameBCID(uint32_t i) const { return
       DIFUnpacker::getFrameBCID(theFrames_[i]); }
00095 inline std::uint32_t              DIFPtr::getFrameTimeToTrigger(uint32_t i) const { return getBCID()
       - getFrameBCID(i); }
00096 inline bool                       DIFPtr::getFrameLevel(uint32_t i, uint32_t ipad, uint32_t ilevel)
       const { return DIFUnpacker::getFrameLevel(theFrames_[i], ipad, ilevel); }
00097 /*void                              dumpDIFInfo()
00098   {
00099     printf("DIF %d DTC %d GTC %d ABCID %lld BCID %d Lines %d Temperature %d \n", getID(), getDTC(),
       getGTC(), getAbsoluteBCID(), getBCID(), getLines(), hasTemperature());
00100
00101     if(hasTemperature()) printf("T: ASU1 %d %f ASU2 %d %f DIF %d  %f \n", getTASU1(),
       getTemperatureASU1(), getTASU2(), getTemperatureASU2(), getTDIF(), getTemperatureDIF());
00102     printf("Found %ld Lines and %ld Frames \n", theLines_.size(), theFrames_.size());
00103   }*/
00104 // Addition by GG
00105 inline uint32_t                   DIFPtr::getDIFid() const { return getID() & 0xFF; }
00106 inline uint32_t                   DIFPtr::getASICid(uint32_t i) const { return getFrameAsicHeader(i)
       & 0xFF; }
00107 inline uint32_t                   DIFPtr::getThresholdStatus(uint32_t i, uint32_t ipad) const {
       return (((uint32_t)getFrameLevel(i, ipad, 1)) << 1) | ((uint32_t)getFrameLevel(i, ipad, 0)); }
```

## 5.7 /home/runner/work/streamout/streamout/libs/core/include/DIFSlow↩ Control.h File Reference

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

### Classes

- class DIFSlowControl

  *Handler of DIF Slow Control info.*

### 5.7.1 Detailed Description

**Copyright**

2022 G.Grenier F.Lagarde

Definition in file DIFSlowControl.h.

## 5.8 DIFSlowControl.h

Go to the documentation of this file.

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

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

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

### Classes

- class DIFUnpacker

### 5.9.1 Detailed Description

**Copyright**

2022 G.Grenier F.Lagarde

Definition in file DIFUnpacker.h.

## 5.10 DIFUnpacker.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include <cstdint>
00008 #include <vector>
00009
00010 class DIFUnpacker
00011 {
00012 public:
00013   static std::uint64_t GrayToBin(const std::uint64_t& n);
00014   static std::uint32_t getStartOfDIF(const unsigned char* cbuf, const std::uint32_t& size_buf, const
       std::uint32_t& start = 92);
00015   static std::uint32_t getID(const unsigned char* cb, const std::uint32_t& idx = 0);
00016   static std::uint32_t getDTC(const unsigned char* cb, const std::uint32_t& idx = 0);
00017   static std::uint32_t getGTC(const unsigned char* cb, const std::uint32_t& idx = 0);
00018   static std::uint64_t getAbsoluteBCID(const unsigned char* cb, const std::uint32_t& idx = 0);
00019   static std::uint32_t getBCID(const unsigned char* cb, const std::uint32_t& idx = 0);
00020   static std::uint32_t getLines(const unsigned char* cb, const std::uint32_t& idx = 0);
00021   static bool          hasLine(const std::uint32_t& line, const unsigned char* cb, const
       std::uint32_t& idx = 0);
00022   static std::uint32_t getTASU1(const unsigned char* cb, const std::uint32_t& idx = 0);
00023   static std::uint32_t getTASU2(const unsigned char* cb, const std::uint32_t& idx = 0);
00024   static std::uint32_t getTDIF(const unsigned char* cb, const std::uint32_t& idx = 0);
00025   static bool          hasTemperature(const unsigned char* cb, const std::uint32_t& idx = 0);
00026   static bool          hasAnalogReadout(const unsigned char* cb, const std::uint32_t& idx = 0);
00027
00028   static std::uint32_t getFrameAsicHeader(const unsigned char* framePtr);
00029   static std::uint32_t getFrameBCID(const unsigned char* framePtr);
00030
00031   static bool getFramePAD(const unsigned char* framePtr, const std::uint32_t& ip);
00032   static bool getFrameLevel(const unsigned char* framePtr, const std::uint32_t& ip, const
       std::uint32_t& level);
00033
00034   static std::uint32_t getAnalogPtr(std::vector<unsigned char*>& vLines, unsigned char* cb, const
       std::uint32_t& idx = 0);
00035   static std::uint32_t getFramePtr(std::vector<unsigned char*>& vFrame, std::vector<unsigned char*>&
       vLines, const std::uint32_t& max_size, unsigned char* cb, const std::uint32_t& idx = 0);
00036   static void          dumpFrameOld(const unsigned char* buf);
00037   static std::uint32_t swap_bytes(const unsigned char* buf);  // Stolen from DCBufferReader
00038 };
```

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

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

### Functions

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

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

## 5.11.1 Detailed Description

**Copyright**

2022 G.Grenier F.Lagarde

Definition in file Formatters.h.

## 5.11.2 Function Documentation

### 5.11.2.1 to_bin() [1/5]

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

Definition at line 71 of file Formatters.cc.

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

### 5.11.2.2 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); }
```

### 5.11.2.3 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); }
```

#### 5.11.2.4 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); }
```

#### 5.11.2.5 to_bin() [5/5]

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

Definition at line 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 }
```

#### 5.11.2.6 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); }
```

#### 5.11.2.7 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); }
```

#### 5.11.2.8 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); }
```

#### 5.11.2.9 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); }
```

#### 5.11.2.10 to_dec() [5/5]

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

Definition at line 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 }
```

#### 5.11.2.11 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); }
```

#### 5.11.2.12 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); }
```

#### 5.11.2.13 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); }
```

#### 5.11.2.14 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); }
```

#### 5.11.2.15 to_hex() [5/5]

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

Definition at line 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 }
```

#### 5.11.2.16 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); }
```

#### 5.11.2.17 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); }
```

#### 5.11.2.18 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); }
```

#### 5.11.2.19 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); }
```

#### 5.11.2.20 to_oct() [5/5]

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

Definition at line 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 }
```

## 5.12 Formatters.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include "Bits.h"
00008
00009 #include <iosfwd>
00010 #include <string>
00011
00012 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&);
```

## 5.13 /home/runner/work/streamout/streamout/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();*

### 5.13.1 Detailed Description

**Copyright**

2022 G.Grenier F.Lagarde

Definition in file Interface.h.

## 5.14 Interface.h

Go to the documentation of this file.

```
00001
00004 #pragma once
00005
00006 #include "Buffer.h"
00007
00008 #include <memory>
00009 #include <spdlog/logger.h>
00010
00026 class Interface
00027 {
00028 public:
00029   Interface() {}
00030   virtual ~Interface() {}
00031   std::shared_ptr<spdlog::logger>& log() { return m_Logger; }
00032   void                             setLogger(const std::shared_ptr<spdlog::logger>& logger) { m_Logger
       = logger; }
00033
00034 private:
00035   std::shared_ptr<spdlog::logger> m_Logger{nullptr};
00036 };
```

## 5.15 /home/runner/work/streamout/streamout/libs/core/include/RawBufferNavigator.h File Reference

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

### Classes

- class RawBufferNavigator

### 5.15.1 Detailed Description

**Copyright**

2022 G.Grenier F.Lagarde

Definition in file RawBufferNavigator.h.

## 5.16 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   bool           validBuffer();
00026   std::uint32_t  getStartOfDIF();
00027   unsigned char* getDIFBufferStart();
00028   std::uint32_t  getDIFBufferSize();
00029   Buffer         getDIFBuffer();
00030   DIFPtr&        getDIFPtr();
00031   std::uint32_t  getEndOfDIFData();
00032   std::uint32_t  getSizeAfterDIFPtr();
00033   std::uint32_t  getDIF_CRC();
00034   bool           hasSlowControlData();
00035   Buffer         getSCBuffer();
00036   bool           badSCData();
00037   Buffer         getEndOfAllData();
00038   static void    StartAt(const int& start);
00039
00040 private:
00041   void          setSCBuffer();
00042   Buffer        m_Buffer;
00043   Buffer        m_SCbuffer;
00044   std::uint32_t m_DIFstartIndex{0};
00045   DIFPtr        m_TheDIFPtr;
00046   bool          m_BadSCdata{false};
00047   static int    m_Start;
00048 };
```

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

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

### Classes

- class SDHCAL_buffer_loop< SOURCE, DESTINATION >

### 5.17.1 Detailed Description

**Copyright**

2022 G.Grenier F.Lagarde

Definition in file SDHCAL_buffer_loop.h.

# 5.18 SDHCAL_buffer_loop.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include "Buffer.h"
00008 #include "Formatters.h"
00009 #include "RawBufferNavigator.h"
00010 #include "SDHCAL_buffer_LoopCounter.h"
00011 #include "Timer.h"
00012
00013 #include <cassert>
00014 #include <memory>
00015 #include <spdlog/sinks/null_sink.h>
00016 #include <spdlog/spdlog.h>
00017 #include <vector>
00018
00019 // function to loop on buffers
00020
00021 template<typename SOURCE, typename DESTINATION> class SDHCAL_buffer_loop
00022 {
00023 public:
00024   SDHCAL_buffer_loop(SOURCE& source, DESTINATION& dest, bool debug = false) : m_Source(source),
      m_Destination(dest), m_Debug(debug)
00025   {
00026     m_Logger = spdlog::create<spdlog::sinks::null_sink_mt>("streamout");
00027     if(!spdlog::get("streamout")) { spdlog::register_logger(m_Logger); }
00028     m_Source.setLogger(m_Logger);
00029     m_Destination.setLogger(m_Logger);
00030   }
00031
00032   void addSink(const spdlog::sink_ptr& sink, const spdlog::level::level_enum& level =
      spdlog::get_level())
00033   {
00034     sink->set_level(level);
00035     m_Sinks.push_back(sink);
00036     m_Logger = std::make_shared<spdlog::logger>("streamout", begin(m_Sinks), end(m_Sinks));
00037     m_Source.setLogger(m_Logger);
00038     m_Destination.setLogger(m_Logger);
00039   }
00040
00041   void loop(const std::int32_t& m_NbrEventsToProcess = 0)
00042   {
00043     Timer timer;
00044     timer.start();
00045     m_Source.start();
00046     m_Destination.start();
00047     RawBufferNavigator bufferNavigator;
00048     while(m_Source.nextEvent() && (m_NbrEventsToProcess == 0 || m_NbrEventsToProcess >= m_NbrEvents))
00049     {
00050       m_Logger->warn("===*** Event number {} ***===", m_NbrEvents);
00051       while(m_Source.nextDIFbuffer())
00052       {
00053         const Buffer& buffer           = m_Source.getSDHCALBuffer();
00054         bit8_t*       debug_variable_1 = buffer.end();
00055         bufferNavigator.setBuffer(buffer);
00056         bit8_t* debug_variable_2 = bufferNavigator.getDIFBuffer().end();
00057         m_Logger->info("DIF BUFFER END {} {}", fmt::ptr(debug_variable_1),
      fmt::ptr(debug_variable_2));
00058         if(m_Debug) assert(debug_variable_1 == debug_variable_2);
00059         uint32_t idstart = bufferNavigator.getStartOfDIF();
00060         if(m_Debug && idstart == 0) m_Logger->info(to_hex(buffer));
00061         c.DIFStarter[idstart]++;
00062         if(!bufferNavigator.validBuffer())
00063         {
00064           m_Logger->error("!bufferNavigator.validBuffer()");
00065           continue;
00066         }
00067         DIFPtr& d = bufferNavigator.getDIFPtr();
00068         c.DIFPtrValueAtReturnedPos[bufferNavigator.getDIFBufferStart()[d.getGetFramePtrReturn()]]++;
00069         if(m_Debug) assert(bufferNavigator.getDIFBufferStart()[d.getGetFramePtrReturn()] == 0xa0);
00070         c.SizeAfterDIFPtr[bufferNavigator.getSizeAfterDIFPtr()]++;
00071         m_Destination.processDIF(d);
00072         for(uint32_t i = 0; i < d.getNumberOfFrames(); i++)
00073         {
00074           m_Destination.processFrame(d, i);
00075           for(uint32_t j = 0; j < 64; j++) m_Destination.processPadInFrame(d, i, j);
00076         }
00077
00078         bool processSC = false;
00079         if(bufferNavigator.hasSlowControlData())
00080         {
00081           c.hasSlowControl++;
00082           processSC = true;
```

```
00083         }
00084         if(bufferNavigator.badSCData())
00085         {
00086           c.hasBadSlowControl++;
00087           processSC = false;
00088         }
00089         if(processSC) { m_Destination.processSlowControl(bufferNavigator.getSCBuffer()); }
00090 
00091         Buffer eod = bufferNavigator.getEndOfAllData();
00092         c.SizeAfterAllData[eod.size()]++;
00093         bit8_t* debug_variable_3 = eod.end();
00094         m_Logger->info("END DATA BUFFER END {} {}", fmt::ptr(debug_variable_1),
      fmt::ptr(debug_variable_3));
00095         if(m_Debug) assert(debug_variable_1 == debug_variable_3);
00096         if(eod.size() != 0) m_Logger->info("End of Data remaining stuff : {}", to_hex(eod));
00097 
00098         int nonzeroCount = 0;
00099         for(unsigned char* it = eod.begin(); it != eod.end(); it++)
00100           if(static_cast<int>(*it) != 0) nonzeroCount++;
00101         c.NonZeroValusAtEndOfData[nonzeroCount]++;
00102       }  // end of DIF while loop
00103       m_Logger->warn("***=== Event number {} ===***", m_NbrEvents);
00104       m_NbrEvents++;
00105     }  // end of event while loop
00106     m_Destination.end();
00107     m_Source.end();
00108     timer.stop();
00109     fmt::print("=== elapsed time {}ms ({}ms/event) ===\n", timer.getElapsedTime() / 1000,
      timer.getElapsedTime() / (1000 * m_NbrEvents));
00110   }
00111   void                              printAllCounters() { c.printAllCounters(); }
00112   std::shared_ptr<spdlog::logger> log() { return m_Logger; }
00113 
00114 private:
00115   std::shared_ptr<spdlog::logger> m_Logger{nullptr};
00116   std::vector<spdlog::sink_ptr>   m_Sinks;
00117   SDHCAL_buffer_LoopCounter       c;
00118   SOURCE&                         m_Source{nullptr};
00119   DESTINATION&                    m_Destination{nullptr};
00120   bool                            m_Debug{false};
00121   std::uint32_t                   m_NbrEvents{1};
00122 };
```

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

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

### Classes

- struct SDHCAL_buffer_LoopCounter

### 5.19.1 Detailed Description

**Copyright**

2022 G.Grenier F.Lagarde

Definition in file SDHCAL_buffer_LoopCounter.h.

## 5.20 SDHCAL_buffer_LoopCounter.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 SDHCAL_buffer_LoopCounter
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 };
```

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

```
#include <chrono>
```

### Classes

- class Timer

### 5.21.1 Detailed Description

**Copyright**

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

**See also**

https://github.com/apingault/Trivent4HEP

Definition in file Timer.h.

## 5.22 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 };
```

## 5.23 /home/runner/work/streamout/streamout/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 }

### 5.23.1 Detailed Description

**Copyright**

2022 G.Grenier F.Lagarde

Definition in file Words.h.

### 5.23.2 Enumeration Type Documentation

#### 5.23.2.1 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 | |

**Enumerator**

| | |
|---|---|
| TASU1_SHIFT | |
| TASU2_SHIFT | |
| TDIF_SHIFT | |
| FRAME_ASIC_HEADER_SHIFT | |
| FRAME_BCID_SHIFT | |
| FRAME_DATA_SHIFT | |
| FRAME_SIZE | |

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 };
```

## 5.24 Words.h

Go to the documentation of this file.

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

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

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

### Functions

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

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

### 5.25.1 Detailed Description

**Copyright**

2022 G.Grenier F.Lagarde

Definition in file Bits.cc.

### 5.25.2 Function Documentation

#### 5.25.2.1 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; }
```

## 5.26 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; }
```

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

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

## 5.28 Buffer.cc

Go to the documentation of this file.

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

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

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

### 5.29.1 Detailed Description

**Copyright**

2022 G.Grenier F.Lagarde

Definition in file DIFSlowControl.cc.

## 5.30 DIFSlowControl.cc

Go to the documentation of this file.

```
00001
00005 #include "DIFSlowControl.h"
00006
00007 #include <cstdint>
00008 #include <iostream>
00009
00010 DIFSlowControl::DIFSlowControl(const std::uint8_t& version, const std::uint8_t& DIfId, unsigned char*
       cbuf) : m_Version(version), m_DIFId(DIfId), m_AsicType(2)
00011 {
00012   if(cbuf[0] != 0xb1) return;
00013   int header_shift{6};
00014   if(m_Version < 8) m_NbrAsic = cbuf[5];
00015   else
00016   {
00017     m_DIFId      = cbuf[1];
00018     m_NbrAsic    = cbuf[2];
00019     header_shift = 3;
00020   }
00021   int size_hardroc1 = m_NbrAsic * 72 + header_shift + 1;
00022   if(cbuf[size_hardroc1 - 1] != 0xa1) size_hardroc1 = 0;
00023
00024   int size_hardroc2 = m_NbrAsic * 109 + header_shift + 1;
00025   if(cbuf[size_hardroc2 - 1] != 0xa1) size_hardroc2 = 0;
00026   if(size_hardroc1 != 0)
00027   {
00028     FillHR1(header_shift, cbuf);
00029     m_AsicType = 1;
00030   }
00031   else if(size_hardroc2 != 0)
00032     FillHR2(header_shift, cbuf);
00033   else
00034     return;
00035 }
00036
00037 inline std::uint8_t DIFSlowControl::getDIFId() { return m_DIFId; }
00038
```

```
00039 inline std::map<int, std::map<std::string, int» DIFSlowControl::getChipsMap() { return m_MapSC; }
00040
00041 inline std::map<std::string, int> DIFSlowControl::getChipSlowControl(const int& asicid) { return
       m_MapSC[asicid]; }
00042
00043 inline int DIFSlowControl::getChipSlowControl(const std::int8_t& asicid, const std::string& param) {
       return getChipSlowControl(asicid)[param]; }
00044
00045 void DIFSlowControl::Dump()
00046 {
00047   for(std::map<int, std::map<std::string, int»::iterator it = m_MapSC.begin(); it != m_MapSC.end();
       it++)
00048   {
00049     std::cout « "ASIC " « it->first « std::endl;
00050     for(std::map<std::string, int>::iterator jt = (it->second).begin(); jt != (it->second).end();
       jt++) std::cout « jt->first « " : " « jt->second « std::endl;
00051   }
00052 }
00053
00054 void DIFSlowControl::FillHR1(const int& header_shift, unsigned char* cbuf)
00055 {
00056   int nasic{cbuf[header_shift - 1]};
00057   int idx{header_shift};
00058   for(int k = 0; k < nasic; k++)
00059   {
00060     std::bitset<72 * 8> bs;
00061     // printf("%x %x \n",cbuf[idx+k*72+69],cbuf[idx+k*72+70]);
00062     for(int l = 71; l >= 0; l--)
00063     {
00064       //  printf("%d %x : %d -->",l,cbuf[idx+k*72+l],(71-l)*8);
00065       for(int m = 0; m < 8; m++)
00066       {
00067         if(((1 « m) & cbuf[idx + k * 72 + l]) != 0) bs.set((71 - l) * 8 + m, 1);
00068         else
00069           bs.set((71 - l) * 8 + m, 0);
00070         // printf("%d",(int) bs[(71-l)*8+m]);
00071       }
00072       // printf("\n");
00073     }
00074     FillAsicHR1(bs);
00075   }
00076 }
00077
00078 void DIFSlowControl::FillHR2(const int& header_shift, unsigned char* cbuf)
00079 {
00080   // int scsize1=cbuf[header_shift-1]*109+(header_shift-1)+2;
00081   int nasic{cbuf[header_shift - 1]};
00082   int idx{header_shift};
00083   // std::cout«" DIFSlowControl::FillHR nasic "«nasic«std::endl;
00084   for(int k = 0; k < nasic; k++)
00085   {
00086     std::bitset<109 * 8> bs;
00087     // printf("%x %x \n",cbuf[idx+k*109+69],cbuf[idx+k*109+70]);
00088     for(int l = 108; l >= 0; l--)
00089     {
00090       //  printf("%d %x : %d -->",l,cbuf[idx+k*109+l],(71-l)*8);
00091       for(int m = 0; m < 8; m++)
00092       {
00093         if(((1 « m) & cbuf[idx + k * 109 + l]) != 0) bs.set((108 - l) * 8 + m, 1);
00094         else
00095           bs.set((108 - l) * 8 + m, 0);
00096         // printf("%d",(int) bs[(71-l)*8+m]);
00097       }
00098       // printf("\n");
00099     }
00100     FillAsicHR2(bs);
00101   }
00102 }
00103
00104 void DIFSlowControl::FillAsicHR1(const std::bitset<72 * 8>& bs)
00105 {
00106   // Asic Id
00107   int asicid{0};
00108   for(int j = 0; j < 8; j++)
00109     if(bs[j + 9] != 0) asicid += (1 « (7 - j));
00110   std::map<std::string, int> mAsic;
00111   // Slow Control
00112   mAsic["SSC0"]          = static_cast<int>(bs[575]);
00113   mAsic["SSC1"]          = static_cast<int>(bs[574]);
00114   mAsic["SSC2"]          = static_cast<int>(bs[573]);
00115   mAsic["Choix_caisson"] = static_cast<int>(bs[572]);
00116   mAsic["SW_50k"]        = static_cast<int>(bs[571]);
00117   mAsic["SW_100k"]       = static_cast<int>(bs[570]);
00118   mAsic["SW_100f"]       = static_cast<int>(bs[569]);
00119   mAsic["SW_50f"]        = static_cast<int>(bs[568]);
00120
00121   mAsic["Valid_DC"]  = static_cast<int>(bs[567]);
```

```
00122   mAsic["ON_Discri"] = static_cast<int>(bs[566]);
00123   mAsic["ON_Fsb"]    = static_cast<int>(bs[565]);
00124   mAsic["ON_Otaq"]   = static_cast<int>(bs[564]);
00125   mAsic["ON_W"]      = static_cast<int>(bs[563]);
00126   mAsic["ON_Ss"]     = static_cast<int>(bs[562]);
00127   mAsic["ON_Buf"]    = static_cast<int>(bs[561]);
00128   mAsic["ON_Paf"]    = static_cast<int>(bs[560]);
00129   // Gain
00130   for(int i = 0; i < 64; i++)
00131   {
00132     int gain{0};
00133     for(int j = 0; j < 6; j++)
00134       if(bs[176 + i * 6 + j] != 0) gain += (1 << j);
00135     mAsic["Channel_" + std::to_string(i) + "_" + "Gain"]       = gain;
00136     mAsic["Channel_" + std::to_string(i) + "_" + "cTest"]      = bs[112 + i];
00137     mAsic["Channel_" + std::to_string(i) + "_" + "Valid_trig"] = static_cast<int>(bs[25 + i]);
00138   }
00139
00140   mAsic["ON_Otabg"]  = static_cast<int>(bs[111]);
00141   mAsic["ON_Dac"]    = static_cast<int>(bs[110]);
00142   mAsic["ON_Otadac"] = static_cast<int>(bs[109]);
00143   // DAC
00144   int dac1{0};
00145   for(int j = 0; j < 10; j++)
00146     if(bs[j + 99] != 0) dac1 += (1 << j);
00147   mAsic["DAC1"] = dac1;
00148   int dac0{0};
00149   for(int j = 0; j < 10; j++)
00150     if(bs[j + 89] != 0) dac0 += (1 << j);
00151   mAsic["DAC0"]             = dac0;
00152   mAsic["EN_Raz_Ext"]       = static_cast<int>(bs[23]);
00153   mAsic["EN_Raz_Int"]       = static_cast<int>(bs[22]);
00154   mAsic["EN_Out_Raz_Int"]   = static_cast<int>(bs[21]);
00155   mAsic["EN_Trig_Ext"]      = static_cast<int>(bs[20]);
00156   mAsic["EN_Trig_Int"]      = static_cast<int>(bs[19]);
00157   mAsic["EN_Out_Trig_Int"]  = static_cast<int>(bs[18]);
00158   mAsic["Bypass_Chip"]      = static_cast<int>(bs[17]);
00159   mAsic["HardrocHeader"]    = static_cast<int>(asicid);
00160   mAsic["EN_Out_Discri"]    = static_cast<int>(bs[8]);
00161   mAsic["EN_Transmit_On"]   = static_cast<int>(bs[7]);
00162   mAsic["EN_Dout"]          = static_cast<int>(bs[6]);
00163   mAsic["EN_RamFull"]       = static_cast<int>(bs[5]);
00164   m_MapSC[asicid]           = mAsic;
00165 }
00166
00167 void DIFSlowControl::FillAsicHR2(const std::bitset<109 * 8>& bs)
00168 {
00169   int asicid{0};
00170   for(int j = 0; j < 8; j++)
00171     if(bs[j + (108 - 7) * 8 + 2] != 0) asicid += (1 << (7 - j));
00172   std::map<std::string, int> mAsic;
00173   for(int i = 0; i < 64; i++)
00174   {
00175     int gain{0};
00176     int mask{0};
00177     mAsic["Channel_" + std::to_string(i) + "_" + "cTest"] = bs[i];
00178     for(int j = 0; j < 8; j++)
00179       if(bs[64 + i * 8 + j] != 0) gain += (1 << j);
00180     mAsic["Channel_" + std::to_string(i) + "_" + "Gain"] = gain;
00181     for(int j = 0; j < 3; j++)
00182       if(bs[8 * 77 + 2 + i * 3 + j] != 0) mask += (1 << j);
00183     mAsic["Channel_" + std::to_string(i) + "_" + "Mask"] = mask;
00184   }
00185   mAsic["PwrOnPA"] = static_cast<int>(bs[8 * 72]);
00186   mAsic["Cmdb3SS"] = static_cast<int>(bs[8 * 72 + 1]);
00187   mAsic["Cmdb2SS"] = static_cast<int>(bs[8 * 72 + 2]);
00188   mAsic["Cmdb1SS"] = static_cast<int>(bs[8 * 72 + 3]);
00189   mAsic["Cmdb0SS"] = static_cast<int>(bs[8 * 72 + 4]);
00190   mAsic["SwSsc0"]  = static_cast<int>(bs[8 * 72 + 5]);
00191   mAsic["SwSsc1"]  = static_cast<int>(bs[8 * 72 + 6]);
00192   mAsic["SwSsc2"]  = static_cast<int>(bs[8 * 72 + 7]);
00193
00194   mAsic["PwrOnBuff"] = static_cast<int>(bs[8 * 73]);
00195   mAsic["PwrOnSS"]   = static_cast<int>(bs[8 * 73 + 1]);
00196   mAsic["PwrOnW"]    = static_cast<int>(bs[8 * 73 + 2]);
00197   mAsic["Cmdb3Fsb2"] = static_cast<int>(bs[8 * 73 + 3]);
00198   mAsic["Cmdb2Fsb2"] = static_cast<int>(bs[8 * 73 + 4]);
00199   mAsic["Cmdb1Fsb2"] = static_cast<int>(bs[8 * 73 + 5]);
00200   mAsic["Cmdb0Fsb2"] = static_cast<int>(bs[8 * 73 + 6]);
00201   mAsic["Sw50k2"]    = static_cast<int>(bs[8 * 73 + 7]);
00202
00203   mAsic["Sw100k2"]   = static_cast<int>(bs[8 * 74]);
00204   mAsic["Sw100f2"]   = static_cast<int>(bs[8 * 74 + 1]);
00205   mAsic["Sw50f2"]    = static_cast<int>(bs[8 * 74 + 2]);
00206   mAsic["Cmdb3Fsb1"] = static_cast<int>(bs[8 * 74 + 3]);
00207   mAsic["Cmdb2Fsb1"] = static_cast<int>(bs[8 * 74 + 4]);
00208   mAsic["Cmdb1Fsb1"] = static_cast<int>(bs[8 * 74 + 5]);
```

```
00209   mAsic["Cmdb0Fsb1"] = static_cast<int>(bs[8 * 74 + 6]);
00210   mAsic["Sw50k1"]    = static_cast<int>(bs[8 * 74 + 7]);
00211
00212   mAsic["Sw100k1"]   = static_cast<int>(bs[8 * 75]);
00213   mAsic["Sw100f1"]   = static_cast<int>(bs[8 * 75 + 1]);
00214   mAsic["Sw50f1"]    = static_cast<int>(bs[8 * 75 + 2]);
00215   mAsic["Sel0"]      = static_cast<int>(bs[8 * 75 + 3]);
00216   mAsic["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["Discri_PwrA"] = static_cast<int>(bs[8 * 76 + 6]);
00228   mAsic["Discri2"]     = static_cast<int>(bs[8 * 76 + 7]);
00229
00230   mAsic["Discri1"]      = static_cast<int>(bs[8 * 77]);
00231   mAsic["RS_or_Discri"] = static_cast<int>(bs[8 * 77 + 1]);
00232
00233   mAsic["Header"] = asicid;
00234   for(int i = 0; i < 3; i++)
00235   {
00236     int B = 0;
00237     for(int j = 0; j < 10; j++)
00238       if(bs[8 * 102 + 2 + i * 10 + j] != 0) B += (1 << j);
00239     mAsic["B" + std::to_string(i)] = B;
00240   }
00241
00242   mAsic["Smalldac"]  = static_cast<int>(bs[8 * 106]);
00243   mAsic["DacSw"]     = static_cast<int>(bs[8 * 106 + 1]);
00244   mAsic["OtagBgSw"]  = static_cast<int>(bs[8 * 106 + 2]);
00245   mAsic["Trig2b"]    = static_cast<int>(bs[8 * 106 + 3]);
00246   mAsic["Trig1b"]    = static_cast<int>(bs[8 * 106 + 4]);
00247   mAsic["Trig0b"]    = static_cast<int>(bs[8 * 106 + 5]);
00248   mAsic["EnTrigOut"] = static_cast<int>(bs[8 * 106 + 6]);
00249   mAsic["DiscrOrOr"] = static_cast<int>(bs[8 * 106 + 7]);
00250
00251   mAsic["TrigExtVal"]   = static_cast<int>(bs[8 * 107]);
00252   mAsic["RazChnIntVal"] = static_cast<int>(bs[8 * 107 + 1]);
00253   mAsic["RazChnExtVal"] = static_cast<int>(bs[8 * 107 + 2]);
00254   mAsic["ScOn"]         = static_cast<int>(bs[8 * 107 + 3]);
00255   mAsic["CLKMux"]       = static_cast<int>(bs[8 * 107 + 4]);
00256
00257   // EnOCDout1b   EnOCDout2b   EnOCTransmitOn1b   EnOCTransmitOn2b   EnOCChipsatb   SelStartReadout
      SelEndReadout
00258   mAsic["SelEndReadout"]    = static_cast<int>(bs[8 * 108 + 1]);
00259   mAsic["SelStartReadout"]  = static_cast<int>(bs[8 * 108 + 2]);
00260   mAsic["EnOCChipsatb"]     = static_cast<int>(bs[8 * 108 + 3]);
00261   mAsic["EnOCTransmitOn2b"] = static_cast<int>(bs[8 * 108 + 4]);
00262   mAsic["EnOCTransmitOn1b"] = static_cast<int>(bs[8 * 108 + 5]);
00263   mAsic["EnOCDout2b"]       = static_cast<int>(bs[8 * 108 + 6]);
00264   mAsic["EnOCDout1b"]       = static_cast<int>(bs[8 * 108 + 7]);
00265   m_MapSC[asicid]           = mAsic;
00266 }
```

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

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

### 5.31.1 Detailed Description

### Copyright

2022 G.Grenier F.Lagarde

Definition in file DIFUnpacker.cc.

## 5.32 DIFUnpacker.cc

Go to the documentation of this file.

```
00001
00005 #include "DIFUnpacker.h"
00006
00007 #include "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;
       // jenlev 1
00113     if(DIFUnpacker::hasAnalogReadout(cb, idx)) fshift = DIFUnpacker::getAnalogPtr(vLines, cb, fshift);
       // to be implemented
00114   }
00115   else
00116     fshift = idx + DU::BCID_SHIFT + 3;
00117   if(cb[fshift] != DU::START_OF_FRAME)
00118   {
00119     std::cout << "This is not a start of frame " << to_hex(cb[fshift]) << " \n";
00120     return fshift;
00121   }
00122   do {
00123     // printf("fshift %d and %d \n",fshift,max_size);
00124     if(cb[fshift] == DU::END_OF_DIF) return fshift;
00125     if(cb[fshift] == DU::START_OF_FRAME) fshift++;
00126     if(cb[fshift] == DU::END_OF_FRAME)
00127     {
00128       fshift++;
00129       continue;
00130     }
00131     std::uint32_t header = DIFUnpacker::getFrameAsicHeader(&cb[fshift]);
00132     if(header == DU::END_OF_FRAME) return (fshift + 2);
00133     // std::cout<<header<<" "<<fshift<<std::endl;
00134     if(header < 1 || header > 48) { throw header + " Header problem " + fshift; }
```

```
00135     vFrame.push_back(&cb[fshift]);
00136     fshift += DU::FRAME_SIZE;
00137     if(fshift > max_size)
00138     {
00139       std::cout « "fshift " « fshift « " exceed " « max_size « "\n";
00140       return fshift;
00141     }
00142     if(cb[fshift] == DU::END_OF_FRAME) fshift++;
00143   } while(true);
00144 }
00145
00146 void DIFUnpacker::dumpFrameOld(const unsigned char* buf)
00147 {
00148   bool         PAD[128];
00149   bool         l0[64];
00150   bool         l1[64];
00151   std::uint8_t un{1};
00152   for(std::size_t ip = 0; ip < 128; ip++) { PAD[ip] = false; }  // init PADs
00153   std::uint32_t idx1{4};
00154   for(int ik = 0; ik < 4; ik++)
00155   {
00156     std::uint32_t PadEtat{swap_bytes(&buf[idx1])};
00157     idx1 += 4;
00158     for(int e = 0; e < 32; e++)
00159     {
00160       PAD[((3 - ik) * 32) + (31 - e)] = PadEtat & un;  // binary operation
00161       PadEtat                         = PadEtat » 1;  // décalage des bit de 1
00162     }
00163   }
00164   // fill bool arrays
00165   for(int p = 0; p < 64; p++)
00166   {
00167     l0[p] = static_cast<bool>(PAD[(2 * p)]);      // _Lev0 (PAD paire)
00168     l1[p] = static_cast<bool>(PAD[(2 * p) + 1]);  // _Lev1 (PAD impaires)
00169   }
00170   std::bitset<64> bs0(0);
00171   std::bitset<64> bs1(0);
00172   for(std::uint32_t ip = 0; ip < 64; ip++)
00173   {
00174     bs0.set(ip, l0[ip]);
00175     bs1.set(ip, l1[ip]);
00176   }
00177   std::cout « "\t \t" « bs0 « std::endl;
00178   std::cout « "\t \t" « bs1 « std::endl;
00179 }
00180
00181 std::uint32_t DIFUnpacker::swap_bytes(const unsigned char* buf)
00182 {
00183   unsigned char Swapped[4];
00184   for(std::size_t i = 0; i < 4; i++) Swapped[i] = buf[4 - 1 - i];
00185   return *reinterpret_cast<std::uint32_t*>(&Swapped[0]);
00186 }
```

## 5.33 /home/runner/work/streamout/streamout/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)

### 5.33.1 Detailed Description

**Copyright**

2022 G.Grenier F.Lagarde

Definition in file Formatters.cc.

### 5.33.2 Function Documentation

#### 5.33.2.1 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); }
```

#### 5.33.2.2 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); }
```

#### 5.33.2.3 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); }
```

#### 5.33.2.4 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); }
```

#### 5.33.2.5 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 }
```

#### 5.33.2.6 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); }
```

#### 5.33.2.7 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); }
```

#### 5.33.2.8 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); }
```

#### 5.33.2.9 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); }
```

#### 5.33.2.10 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 }
```

#### 5.33.2.11 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); }
```

#### 5.33.2.12 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); }
```

#### 5.33.2.13 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); }
```

#### 5.33.2.14 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); }
```

#### 5.33.2.15 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 }
```

#### 5.33.2.16 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); }
```

#### 5.33.2.17 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); }
```

#### 5.33.2.18 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); }
```

#### 5.33.2.19 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); }
```

#### 5.33.2.20 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 }
```

# 5.34 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); }
```

## 5.35 /home/runner/work/streamout/streamout/libs/core/src/RawBufferNavigator.cc File Reference

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

### 5.35.1 Detailed Description

**Copyright**

2022 G.Grenier F.Lagarde

Definition in file RawBufferNavigator.cc.

## 5.36 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 {
00018   StartAt(start);
00019   m_DIFstartIndex = DIFUnpacker::getStartOfDIF(m_Buffer.begin(), m_Buffer.size(), m_Start);
00020 }
00021
00022 bool RawBufferNavigator::validBuffer() { return m_DIFstartIndex != 0; }
00023
00024 std::uint32_t RawBufferNavigator::getStartOfDIF() { return m_DIFstartIndex; }
00025
00026 unsigned char* RawBufferNavigator::getDIFBufferStart() { return &(m_Buffer.begin()[m_DIFstartIndex]);
       }
00027
00028 std::uint32_t RawBufferNavigator::getDIFBufferSize() { return m_Buffer.size() - m_DIFstartIndex; }
00029
00030 Buffer RawBufferNavigator::getDIFBuffer() { return Buffer(getDIFBufferStart(), getDIFBufferSize()); }
00031
00032 DIFPtr& RawBufferNavigator::getDIFPtr()
00033 {
00034   m_TheDIFPtr.setBuffer(getDIFBufferStart(), getDIFBufferSize());
00035   return m_TheDIFPtr;
00036 }
00037
00038 std::uint32_t RawBufferNavigator::getEndOfDIFData() { return getDIFPtr().getGetFramePtrReturn() + 3; }
00039
```

```
00040 std::uint32_t RawBufferNavigator::getSizeAfterDIFPtr() { return getDIFBufferSize() -
      getDIFPtr().getGetFramePtrReturn(); }
00041
00042 std::uint32_t RawBufferNavigator::getDIF_CRC()
00043 {
00044   uint32_t i{getEndOfDIFData()};
00045   uint32_t ret{0};
00046   ret |= ((m_Buffer.begin()[i - 2]) « 8);
00047   ret |= m_Buffer.begin()[i - 1];
00048   return ret;
00049 }
00050
00051 bool RawBufferNavigator::hasSlowControlData() { return getDIFBufferStart()[getEndOfDIFData()] == 0xb1;
      }
00052
00053 Buffer RawBufferNavigator::getSCBuffer()
00054 {
00055   setSCBuffer();
00056   return m_SCbuffer;
00057 }
00058
00059 bool RawBufferNavigator::badSCData()
00060 {
00061   setSCBuffer();
00062   return m_BadSCdata;
00063 }
00064
00065 void RawBufferNavigator::setSCBuffer()
00066 {
00067   if(!hasSlowControlData()) return;
00068   if(m_SCbuffer.size() != 0) return;  // deja fait
00069   if(m_BadSCdata) return;
00070   m_SCbuffer.set(&(getDIFBufferStart()[getEndOfDIFData()]));
00071   // compute Slow Control size
00072   std::size_t maxsize{m_Buffer.size() - m_DIFstartIndex - getEndOfDIFData() + 1};  // should I +1 here
      ?
00073   uint32_t    k{1};                                                               // SC Header
00074   uint32_t    dif_ID{m_SCbuffer[1]};
00075   uint32_t    chipSize{m_SCbuffer[3]};
00076   while((dif_ID != 0xa1 && m_SCbuffer[k] != 0xa1 && k < maxsize) || (dif_ID == 0xa1 && m_SCbuffer[k +
      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] == 0xa1 && !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()]),
      getSizeAfterDIFPtr() - 3 - m_SCbuffer.size()); }
00102   else
00103     return Buffer(&(getDIFBufferStart()[getEndOfDIFData()]), getSizeAfterDIFPtr() - 3);  // remove the
      2 bytes for CRC and the DIF trailer
00104 }
```

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

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

### 5.37.1 Detailed Description

**Copyright**

2022 G.Grenier F.Lagarde

Definition in file SDHCAL_buffer_LoopCounter.cc.

## 5.38 SDHCAL_buffer_LoopCounter.cc

Go to the documentation of this file.

```
00001
00005 #include "SDHCAL_buffer_LoopCounter.h"
00006
00007 #include <fmt/core.h>
00008
00009 void SDHCAL_buffer_LoopCounter::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 SDHCAL_buffer_LoopCounter::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 }
```

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

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

### Classes

- class textDump

### 5.39.1 Detailed Description

**Copyright**

2022 G.Grenier F.Lagarde

Definition in file textDump.h.

## 5.40 textDump.h

Go to the documentation of this file.

```
00001
00005 #pragma once
00006
00007 #include "DIFPtr.h"
00008 #include "Interface.h"
00009 #include "spdlog/sinks/stdout_color_sinks.h"
00010
00011 #include <memory>
00012 #include <ostream>
00013 #include <spdlog/logger.h>
00014
00015 class textDump : public Interface
00016 {
00017 public:
00018   textDump()
00019   {
00020     m_InternalLogger = std::make_shared<spdlog::logger>("textDump",
      std::make_shared<spdlog::sinks::stdout_color_sink_mt>());
00021     m_InternalLogger->set_level(spdlog::level::trace);
00022   }
00023   void                              start();
00024   void                              processDIF(const DIFPtr&);
00025   void                              processFrame(const DIFPtr&, uint32_t frameIndex);
00026   void                              processPadInFrame(const DIFPtr&, uint32_t frameIndex, uint32_t
      channelIndex);
00027   void                              processSlowControl(Buffer);
00028   void                              end();
00029   std::shared_ptr<spdlog::logger>& print() { return m_InternalLogger; }
00030   void                              setLevel(const spdlog::level::level_enum& level) {
      m_InternalLogger->set_level(level); }
00031
00032 private:
00033   // This class is a dumb class to print on terminal so we need the logger + the standard one given by
      the interface.
00034   std::shared_ptr<spdlog::logger> m_InternalLogger{nullptr};
00035 };
```

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

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

### 5.41.1 Detailed Description

**Copyright**

2022 G.Grenier F.Lagarde

Definition in file textDump.cc.

## 5.42 textDump.cc

Go to the documentation of this file.

```
00001
00005 #include "textDump.h"
00006
00007 #include "DIFPtr.h"
00008
00009 void textDump::start() { print()->info("Will dump bunch of DIF data"); }
00010
00011 void textDump::processDIF(const DIFPtr& d)
00012 {
00013   print()->info("DIF number is {}", d.getDIFid());
00014   print()->info("DTC value is {}", d.getDTC());
00015   print()->info("GTC value is {}", d.getGTC());
00016   print()->info("DIF BCID is {}", d.getBCID());
00017   print()->info("Absolute BCID is {}", d.getAbsoluteBCID());
00018   print()->info("The number of frame is {}", d.getNumberOfFrames());
00019 }
00020
00021 void textDump::processFrame(const DIFPtr& d, uint32_t frameIndex)
00022 {
00023   print()->info("Displaying frame number {}", frameIndex);
00024   print()->info("ASIC ID is {}", d.getASICid(frameIndex));
00025   print()->info("Frame BCID is {}", d.getFrameBCID(frameIndex));
00026   print()->info("Frame Time To Trigger (a.k.a timestamp) is {}", d.getFrameTimeToTrigger(frameIndex));
00027 }
00028
00029 void textDump::processPadInFrame(const DIFPtr& d, uint32_t frameIndex, uint32_t channelIndex)
00030 {
00031   if(d.getThresholdStatus(frameIndex, channelIndex) > 0)
00032   {
00033     print()->info("Displaying channel number {}", channelIndex);
00034     print()->info("Threshold status is {}", d.getThresholdStatus(frameIndex, channelIndex));
00035   }
00036 }
00037
00038 void textDump::processSlowControl(Buffer) { print()->error("textDump::processSlowControl not
      implemented yet."); }
00039
00040 void textDump::end() { print()->info("textDump end of report"); }
```

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

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

### Classes

- class RawdataReader

### 5.43.1 Detailed Description

**Copyright**

2022 G.Grenier F.Lagarde

Definition in file RawdataReader.h.

## 5.44 RawdataReader.h

Go to the documentation of this file.

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

## 5.45 /home/runner/work/streamout/streamout/libs/interface/RawDataReader/src/RawdataReader.cc File Reference

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

### 5.45.1 Detailed Description

**Copyright**

2022 G.Grenier F.Lagarde

Definition in file RawdataReader.cc.

## 5.46 RawdataReader.cc

Go to the documentation of this file.

```
00001
00004 #include "RawdataReader.h"
00005
00006 #include <cstdint>
00007 #include <cstring>
00008 #include <stdexcept>
00009 #include <zlib.h>
00010
00012 std::size_t RawdataReader::m_BufferSize = 0x100000;
00013
00014 void RawdataReader::setDefaultBufferSize(const std::size_t& size) { m_BufferSize = size; }
00015
00016 RawdataReader::RawdataReader(const char* fileName)
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(), e.code().value());
00072     throw;
00073   }
00074 }
00075
00076 bool RawdataReader::nextEvent()
00077 {
00078   try
00079   {
00080     m_FileStream.read(reinterpret_cast<char*>(&m_EventNumber), sizeof(std::uint32_t));
00081     m_FileStream.read(reinterpret_cast<char*>(&m_NumberOfDIF), sizeof(std::uint32_t));
00082   }
00083   catch(const std::ios_base::failure& e)
```

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

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

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

### Classes

- class ROOTtreeDest
- struct ROOTtreeDest::DATA

### 5.47.1 Detailed Description

**Copyright**

2022 G.Grenier F.Lagarde

Definition in file ROOTtreeDest.h.

## 5.48 ROOTtreeDest.h

Go to the documentation of this file.

```
00001
00006 #pragma once
00007
00008 #include "Buffer.h"
00009 #include "DIFPtr.h"
00010 #include "Interface.h"
00011 #include "TTree.h"
00012
00013 class ROOTtreeDest : public Interface
00014 {
00015 public:
00016   typedef struct
00017   {
00018     UInt_t    DIFid, ASICid, CHANNELid;
00019     UInt_t    Thresh;
00020     UInt_t    DTC, GTC, DIF_BCID, frame_BCID, timeStamp;
00021     ULong64_t AbsoluteBCID;
00022   } DATA;
00023
00024   ROOTtreeDest();
00025
00026   void start();
00027   void processDIF(const DIFPtr&);
00028   void processFrame(const DIFPtr&, const std::uint32_t& frameIndex);
00029   void processPadInFrame(const DIFPtr&, const std::uint32_t& frameIndex, const std::uint32_t&
      channelIndex);
00030   void processSlowControl(const Buffer&) { ; }
00031   void end() { ; }
00032
00033 private:
00034   DATA   _data;
00035   TTree* _tree;
00036   void   dataReset();
00037 };
```

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

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

### 5.49.1 Detailed Description

**Copyright**

2022 G.Grenier F.Lagarde

Definition in file ROOTtreeDest.cc.

## 5.50 ROOTtreeDest.cc

Go to the documentation of this file.

```
00001
00006 #include "ROOTtreeDest.h"
00007
00008 ROOTtreeDest::ROOTtreeDest()
00009 {
00010   dataReset();
00011   _tree = new TTree("RawData", "Raw SDHCAL data tree");
00012   _tree->Branch("data", &_data,
       "DIFid/i:ASICid:CHANNELid:Thresh:DTC:GTC:DIF_BCID:frame_BCID:timeStamp:AbsoluteBCID/l");
```

```
00013 }
00014
00015 void ROOTtreeDest::dataReset()
00016 {
00017   _data.DIFid = _data.ASICid = _data.CHANNELid = 0;
00018   _data.Thresh                                 = 0;
00019   _data.DTC = _data.GTC = _data.DIF_BCID = _data.frame_BCID = _data.timeStamp = 0;
00020   _data.AbsoluteBCID                                                          = 0;
00021 }
00022
00023 void ROOTtreeDest::start() { dataReset(); }
00024
00025 void ROOTtreeDest::processDIF(const DIFPtr& d)
00026 {
00027   _data.DIFid        = d.getDIFid();
00028   _data.DTC          = d.getDTC();
00029   _data.GTC          = d.getGTC();
00030   _data.DIF_BCID     = d.getBCID();
00031   _data.AbsoluteBCID = d.getAbsoluteBCID();
00032 }
00033
00034 void ROOTtreeDest::processFrame(const DIFPtr& d, const std::uint32_t& frameIndex)
00035 {
00036   _data.ASICid     = d.getASICid(frameIndex);
00037   _data.frame_BCID = d.getFrameBCID(frameIndex);
00038   _data.timeStamp  = d.getFrameTimeToTrigger(frameIndex);
00039 }
00040
00041 void ROOTtreeDest::processPadInFrame(const DIFPtr& d, const std::uint32_t& frameIndex, const
      std::uint32_t& channelIndex)
00042 {
00043   _data.CHANNELid = channelIndex;
00044   _data.Thresh    = d.getThresholdStatus(frameIndex, channelIndex);
00045   if(_data.Thresh != 0) _tree->Fill();
00046 }
```