

streamout

Generated by Doxygen 1.9.2

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 Buffer Class Reference	5
3.1.1 Detailed Description	5
3.1.2 Constructor & Destructor Documentation	5
3.1.2.1 Buffer() [1/5]	6
3.1.2.2 Buffer() [2/5]	6
3.1.2.3 Buffer() [3/5]	6
3.1.2.4 Buffer() [4/5]	6
3.1.2.5 Buffer() [5/5]	6
3.1.2.6 ~Buffer()	7
3.1.3 Member Function Documentation	7
3.1.3.1 begin()	7
3.1.3.2 capacity()	7
3.1.3.3 end()	7
3.1.3.4 operator[]()	7
3.1.3.5 printBuffer() [1/2]	8
3.1.3.6 printBuffer() [2/2]	8
3.1.3.7 set()	8
3.1.3.8 setSize()	8
3.1.3.9 size()	8
3.2 ROOTtreeDest::DATA Struct Reference	9
3.2.1 Detailed Description	9
3.2.2 Member Data Documentation	9
3.2.2.1 AbsoluteBCID	9
3.2.2.2 ASICid	9
3.2.2.3 CHANNELid	9
3.2.2.4 DIF_BCID	10
3.2.2.5 DIFid	10
3.2.2.6 DTC	10
3.2.2.7 frame_BCID	10
3.2.2.8 GTC	10
3.2.2.9 Thresh	10
3.2.2.10 timeStamp	11
3.3 DIFPtr Class Reference	11
3.3.1 Detailed Description	11
3.3.2 Constructor & Destructor Documentation	12
3.3.2.1 DIFPtr()	12

3.3.3 Member Function Documentation	12
3.3.3.1 dumpDIFInfo()	12
3.3.3.2 getAbsoluteBCID()	12
3.3.3.3 getASICid()	13
3.3.3.4 getBCID()	13
3.3.3.5 getDIFid()	13
3.3.3.6 getDTC()	13
3.3.3.7 getFrameAsicHeader()	13
3.3.3.8 getFrameBCID()	14
3.3.3.9 getFrameLevel()	14
3.3.3.10 getFramePtr()	14
3.3.3.11 getFramesVector()	14
3.3.3.12 getFrameTimeToTrigger()	14
3.3.3.13 getGetFramePtrReturn()	15
3.3.3.14 getGTC()	15
3.3.3.15 getID()	15
3.3.3.16 getLines()	15
3.3.3.17 getLinesVector()	15
3.3.3.18 getNumberOfFrames()	15
3.3.3.19 getPtr()	16
3.3.3.20 getTASU1()	16
3.3.3.21 getTASU2()	16
3.3.3.22 getTDIF()	16
3.3.3.23 getTemperatureASU1()	16
3.3.3.24 getTemperatureASU2()	16
3.3.3.25 getTemperatureDIF()	17
3.3.3.26 getThresholdStatus()	17
3.3.3.27 hasAnalogReadout()	17
3.3.3.28 hasLine()	17
3.3.3.29 hasTemperature()	17
3.4 DIFSlowControl Class Reference	18
3.4.1 Detailed Description	18
3.4.2 Constructor & Destructor Documentation	18
3.4.2.1 DIFSlowControl()	18
3.4.3 Member Function Documentation	19
3.4.3.1 Dump()	19
3.4.3.2 getChipSlowControl() [1/2]	19
3.4.3.3 getChipSlowControl() [2/2]	20
3.4.3.4 getChipsMap()	20
3.4.3.5 getDIFid()	21
3.5 DIFUnpacker Class Reference	21
3.5.1 Detailed Description	21

3.5.2 Member Function Documentation	22
3.5.2.1 dumpFrameOld()	22
3.5.2.2 getAbsoluteBCID()	22
3.5.2.3 getAnalogPtr()	23
3.5.2.4 getBCID()	23
3.5.2.5 getDTC()	23
3.5.2.6 getFrameAsicHeader()	23
3.5.2.7 getFrameBCID()	24
3.5.2.8 getFrameLevel()	24
3.5.2.9 getFramePAD()	24
3.5.2.10 getFramePtr()	24
3.5.2.11 getGTC()	25
3.5.2.12 getID()	25
3.5.2.13 getLines()	25
3.5.2.14 getStartOfDIF()	26
3.5.2.15 getTASU1()	26
3.5.2.16 getTASU2()	26
3.5.2.17 getTDIF()	26
3.5.2.18 GrayToBin()	27
3.5.2.19 hasAnalogReadout()	27
3.5.2.20 hasLine()	27
3.5.2.21 hasTemperature()	27
3.5.2.22 swap_bytes()	28
3.6 DU Class Reference	28
3.6.1 Detailed Description	28
3.6.2 Member Data Documentation	28
3.6.2.1 ABCID_SHIFT	29
3.6.2.2 BCID_SHIFT	29
3.6.2.3 DTC_SHIFT	29
3.6.2.4 END_OF_DIF	29
3.6.2.5 END_OF_FRAME	29
3.6.2.6 END_OF_LINES	29
3.6.2.7 FRAME_ASIC_HEADER_SHIFT	30
3.6.2.8 FRAME_BCID_SHIFT	30
3.6.2.9 FRAME_DATA_SHIFT	30
3.6.2.10 FRAME_SIZE	30
3.6.2.11 GTC_SHIFT	30
3.6.2.12 ID_SHIFT	30
3.6.2.13 LINES_SHIFT	31
3.6.2.14 START_OF_DIF	31
3.6.2.15 START_OF_DIF_TEMP	31
3.6.2.16 START_OF_FRAME	31

3.6.2.17 START_OF_LINES	31
3.6.2.18 TASU1_SHIFT	31
3.6.2.19 TASU2_SHIFT	32
3.6.2.20 TDIF_SHIFT	32
3.7 RawdataReader Class Reference	32
3.7.1 Detailed Description	32
3.7.2 Constructor & Destructor Documentation	32
3.7.2.1 RawdataReader()	33
3.7.2.2 ~RawdataReader()	33
3.7.3 Member Function Documentation	33
3.7.3.1 closeFile()	33
3.7.3.2 getFileSize()	33
3.7.3.3 getSDHCALBuffer()	34
3.7.3.4 nextDIFbuffer()	34
3.7.3.5 nextEvent()	34
3.7.3.6 openFile()	35
3.7.3.7 setDefaultBufferSize()	35
3.8 ROOTtreeDest Class Reference	35
3.8.1 Detailed Description	36
3.8.2 Constructor & Destructor Documentation	36
3.8.2.1 ROOTtreeDest()	36
3.8.3 Member Function Documentation	36
3.8.3.1 end()	36
3.8.3.2 processDIF()	36
3.8.3.3 processFrame()	37
3.8.3.4 processPadInFrame()	37
3.8.3.5 processSlowControl()	37
3.8.3.6 start()	37
3.9 SDHCAL_buffer_loop< SOURCE, DESTINATION > Class Template Reference	38
3.9.1 Detailed Description	38
3.9.2 Constructor & Destructor Documentation	38
3.9.2.1 SDHCAL_buffer_loop()	38
3.9.3 Member Function Documentation	38
3.9.3.1 loop()	39
3.9.3.2 printAllCounters()	39
3.10 SDHCAL_buffer_LoopCounter Struct Reference	40
3.10.1 Detailed Description	40
3.10.2 Member Function Documentation	40
3.10.2.1 printAllCounters()	40
3.10.2.2 printCounter()	41
3.10.3 Member Data Documentation	41
3.10.3.1 DIFPtrValueAtReturnedPos	41

3.10.3.2 DIFStarter	41
3.10.3.3 hasBadSlowControl	41
3.10.3.4 hasSlowControl	41
3.10.3.5 NonZeroValusAtEndOfData	42
3.10.3.6 SizeAfterAllData	42
3.10.3.7 SizeAfterDIFPtr	42
3.11 SDHCAL_RawBuffer_Navigator Class Reference	42
3.11.1 Detailed Description	43
3.11.2 Constructor & Destructor Documentation	43
3.11.2.1 SDHCAL_RawBuffer_Navigator()	43
3.11.2.2 ~SDHCAL_RawBuffer_Navigator()	43
3.11.3 Member Function Documentation	43
3.11.3.1 badSCData()	43
3.11.3.2 getDIF_CRC()	44
3.11.3.3 getDIFBuffer()	44
3.11.3.4 getDIFBufferSize()	44
3.11.3.5 getDIFBufferStart()	44
3.11.3.6 getDIFPtr()	44
3.11.3.7 getEndOfAllData()	45
3.11.3.8 getEndOfDIFData()	45
3.11.3.9 getSCBuffer()	45
3.11.3.10 getSizeAfterDIFPtr()	45
3.11.3.11 getStartOfDIF()	45
3.11.3.12 hasSlowControlData()	46
3.11.3.13 StartAt()	46
3.11.3.14 validBuffer()	46
3.12 textDump Class Reference	46
3.12.1 Detailed Description	46
3.12.2 Constructor & Destructor Documentation	47
3.12.2.1 textDump()	47
3.12.3 Member Function Documentation	47
3.12.3.1 end()	47
3.12.3.2 processDIF()	47
3.12.3.3 processFrame()	47
3.12.3.4 processPadInFrame()	48
3.12.3.5 processSlowControl()	48
3.12.3.6 start()	48
4 File Documentation	49
4.1 /home/runner/work/streamout/streamout/libs/core/include/Bits.h File Reference	49
4.1.1 Detailed Description	49
4.1.2 Typedef Documentation	49

4.1.2.1 bit8_t	50
4.1.3 Function Documentation	50
4.1.3.1 operator<<()	50
4.2 Bits.h	50
4.3 /home/runner/work/streamout/streamout/libs/core/include/Buffer.h File Reference	50
4.4 Buffer.h	51
4.5 /home/runner/work/streamout/streamout/libs/core/include/DIFPtr.h File Reference	51
4.5.1 Detailed Description	51
4.6 DIFPtr.h	52
4.7 /home/runner/work/streamout/streamout/libs/core/include/DIFSlowControl.h File Reference	52
4.7.1 Detailed Description	53
4.8 DIFSlowControl.h	53
4.9 /home/runner/work/streamout/streamout/libs/core/include/DIFUnpacker.h File Reference	54
4.9.1 Detailed Description	54
4.10 DIFUnpacker.h	54
4.11 /home/runner/work/streamout/streamout/libs/core/include/SDHCAL_buffer_loop.h File Reference	55
4.11.1 Detailed Description	55
4.12 SDHCAL_buffer_loop.h	55
4.13 /home/runner/work/streamout/streamout/libs/core/include/SDHCAL_buffer_LoopCounter.h File Reference	57
4.13.1 Detailed Description	57
4.14 SDHCAL_buffer_LoopCounter.h	57
4.15 /home/runner/work/streamout/streamout/libs/core/include/SDHCAL_RawBuffer_Navigator.h File Reference	57
4.15.1 Detailed Description	58
4.16 SDHCAL_RawBuffer_Navigator.h	58
4.17 /home/runner/work/streamout/streamout/libs/core/include/Words.h File Reference	58
4.17.1 Detailed Description	59
4.18 Words.h	59
4.19 /home/runner/work/streamout/streamout/libs/core/src/Bits.cc File Reference	59
4.19.1 Detailed Description	59
4.19.2 Function Documentation	60
4.19.2.1 operator<<()	60
4.20 Bits.cc	60
4.21 /home/runner/work/streamout/streamout/libs/core/src/Buffer.cc File Reference	60
4.22 Buffer.cc	60
4.23 /home/runner/work/streamout/streamout/libs/core/src/DIFPtr.cc File Reference	60
4.24 DIFPtr.cc	61
4.25 /home/runner/work/streamout/streamout/libs/core/src/DIFSlowControl.cc File Reference	61
4.25.1 Detailed Description	61
4.26 DIFSlowControl.cc	61
4.27 /home/runner/work/streamout/streamout/libs/core/src/DIFUnpacker.cc File Reference	64
4.27.1 Detailed Description	65

4.28 DIFUnpacker.cc	65
4.29 /home/runner/work/streamout/streamout/libs/core/src/SDHCAL_buffer_LoopCounter.cc File Reference	67
4.29.1 Detailed Description	67
4.30 SDHCAL_buffer_LoopCounter.cc	68
4.31 /home/runner/work/streamout/streamout/libs/core/src/SDHCAL_RawBuffer_Navigator.cc File Reference	68
4.31.1 Detailed Description	68
4.32 SDHCAL_RawBuffer_Navigator.cc	68
4.33 /home/runner/work/streamout/streamout/libs/interface/Dump/include/textDump.h File Reference	70
4.33.1 Detailed Description	70
4.34 textDump.h	70
4.35 /home/runner/work/streamout/streamout/libs/interface/Dump/src/textDump.cc File Reference	71
4.35.1 Detailed Description	71
4.36 textDump.cc	71
4.37 /home/runner/work/streamout/streamout/libs/interface/RawDataReader/include/RawdataReader.h File Reference	71
4.37.1 Detailed Description	72
4.38 RawdataReader.h	72
4.39 /home/runner/work/streamout/streamout/libs/interface/RawDataReader/src/RawdataReader.cc File Reference	72
4.39.1 Detailed Description	73
4.40 RawdataReader.cc	73
4.41 /home/runner/work/streamout/streamout/libs/interface/ROOT/include/ROOTtreeDest.h File Reference	74
4.41.1 Detailed Description	74
4.42 ROOTtreeDest.h	75
4.43 /home/runner/work/streamout/streamout/libs/interface/ROOT/src/ROOTtreeDest.cc File Reference	75
4.43.1 Detailed Description	75
4.44 ROOTtreeDest.cc	75

Chapter 1

Class Index

1.1 Class List

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

Buffer	5
ROOTtreeDest::DATA	9
DIFPtr	11
DIFSlowControl	
Handler of DIF Slow Control info	18
DIFUnpacker	21
DU	28
RawdataReader	32
ROOTtreeDest	35
SDHCAL_buffer_loop< SOURCE, DESTINATION >	38
SDHCAL_buffer_LoopCounter	40
SDHCAL_RawBuffer_Navigator	42
textDump	46

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

/home/runner/work/streamout/streamout/libs/core/include/Bits.h	49
/home/runner/work/streamout/streamout/libs/core/include/Buffer.h	50
/home/runner/work/streamout/streamout/libs/core/include/DIFPtr.h	51
/home/runner/work/streamout/streamout/libs/core/include/DIFSlowControl.h	52
/home/runner/work/streamout/streamout/libs/core/include/DIFUnpacker.h	54
/home/runner/work/streamout/streamout/libs/core/include/SDHCAL_buffer_loop.h	55
/home/runner/work/streamout/streamout/libs/core/include/SDHCAL_buffer_LoopCounter.h	57
/home/runner/work/streamout/streamout/libs/core/include/SDHCAL_RawBuffer_Navigator.h	57
/home/runner/work/streamout/streamout/libs/core/include/Words.h	58
/home/runner/work/streamout/streamout/libs/core/src/Bits.cc	59
/home/runner/work/streamout/streamout/libs/core/src/Buffer.cc	60
/home/runner/work/streamout/streamout/libs/core/src/DIFPtr.cc	60
/home/runner/work/streamout/streamout/libs/core/src/DIFSlowControl.cc	61
/home/runner/work/streamout/streamout/libs/core/src/DIFUnpacker.cc	64
/home/runner/work/streamout/streamout/libs/core/src/SDHCAL_buffer_LoopCounter.cc	67
/home/runner/work/streamout/streamout/libs/core/src/SDHCAL_RawBuffer_Navigator.cc	68
/home/runner/work/streamout/streamout/libs/interface/Dump/include/textDump.h	70
/home/runner/work/streamout/streamout/libs/interface/Dump/src/textDump.cc	71
/home/runner/work/streamout/streamout/libs/interface/RawDataReader/include/RawdataReader.h	71
/home/runner/work/streamout/streamout/libs/interface/RawDataReader/src/RawdataReader.cc	72
/home/runner/work/streamout/streamout/libs/interface/ROOT/include/ROOTtreeDest.h	74
/home/runner/work/streamout/streamout/libs/interface/ROOT/src/ROOTtreeDest.cc	75

Chapter 3

Class Documentation

3.1 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](#) ()
- [bit8_t](#) * [end](#) ()
- [bit8_t](#) & [operator\[\]](#) (const std::size_t &pos)
- void [setSize](#) (const std::size_t &[size](#))
- void [printBuffer](#) (uint32_t start, uint32_t stop, std::ostream &flux=std::cout)
- void [printBuffer](#) (uint32_t start=0, std::ostream &flux=std::cout)
- virtual [~Buffer](#) ()

3.1.1 Detailed Description

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

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Buffer() [1/5]

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

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

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

3.1.2.2 Buffer() [2/5]

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

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

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

3.1.2.3 Buffer() [3/5]

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

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

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

3.1.2.4 Buffer() [4/5]

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

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

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

3.1.2.5 Buffer() [5/5]

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

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

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


3.1.2.6 ~Buffer()

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

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

```
00015 { std::cout << "SDHCAL_buffer destructor called" << std::endl; }
```

3.1.3 Member Function Documentation

3.1.3.1 begin()

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

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

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

3.1.3.2 capacity()

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

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

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

3.1.3.3 end()

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

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

```
00028 { return m_Buffer + m_Size; }
```

3.1.3.4 operator[]()

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

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

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

3.1.3.5 printBuffer() [1/2]

```
void Buffer::printBuffer (
    uint32_t start,
    uint32_t stop,
    std::ostream & flux = std::cout )
```

3.1.3.6 printBuffer() [2/2]

```
void Buffer::printBuffer (
    uint32_t start = 0,
    std::ostream & flux = std::cout ) [inline]
```

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

```
00033 { printBuffer(start, size()); }
```

3.1.3.7 set()

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

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

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

3.1.3.8 setSize()

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

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

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

3.1.3.9 size()

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

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

```
00023 { return m_Size; }
```

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

- [/home/runner/work/streamout/streamout/libs/core/include/Buffer.h](#)
- [/home/runner/work/streamout/streamout/libs/core/src/Buffer.cc](#)

3.2 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](#)

3.2.1 Detailed Description

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

3.2.2 Member Data Documentation

3.2.2.1 AbsoluteBCID

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

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

3.2.2.2 ASICid

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

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

3.2.2.3 CHANNELid

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

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

3.2.2.4 DIF_BCID

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

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

3.2.2.5 DIFid

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

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

3.2.2.6 DTC

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

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

3.2.2.7 frame_BCID

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

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

3.2.2.8 GTC

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

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

3.2.2.9 Thresh

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

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

3.2.2.10 timeStamp

UInt_t ROOTtreeDest::DATA::timeStamp

Definition at line 19 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](#)

3.3 DIFPtr Class Reference

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

Public Member Functions

- [DIFPtr](#) (unsigned char *p, const std::uint32_t &max_size)
- unsigned char * [getPtr](#) ()
- std::uint32_t [getGetFramePtrReturn](#) ()
- std::vector< unsigned char * > & [getFramesVector](#) ()
- std::vector< unsigned char * > & [getLinesVector](#) ()
- std::uint32_t [getID](#) ()
- std::uint32_t [getDTC](#) ()
- std::uint32_t [getGTC](#) ()
- std::uint64_t [getAbsoluteBCID](#) ()
- std::uint32_t [getBCID](#) ()
- std::uint32_t [getLines](#) ()
- bool [hasLine](#) (uint32_t line)
- std::uint32_t [getTASU1](#) ()
- std::uint32_t [getTASU2](#) ()
- std::uint32_t [getTDIF](#) ()
- float [getTemperatureDIF](#) ()
- float [getTemperatureASU1](#) ()
- float [getTemperatureASU2](#) ()
- bool [hasTemperature](#) ()
- bool [hasAnalogReadout](#) ()
- std::uint32_t [getNumberOfFrames](#) ()
- unsigned char * [getFramePtr](#) (uint32_t i)
- std::uint32_t [getFrameAsicHeader](#) (uint32_t i)
- std::uint32_t [getFrameBCID](#) (uint32_t i)
- std::uint32_t [getFrameTimeToTrigger](#) (uint32_t i)
- bool [getFrameLevel](#) (uint32_t i, uint32_t ipad, uint32_t ilevel)
- void [dumpDIFInfo](#) ()
- uint32_t [getDIFid](#) ()
- uint32_t [getASICid](#) (uint32_t i)
- uint32_t [getThresholdStatus](#) (uint32_t i, uint32_t ipad)

3.3.1 Detailed Description

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

3.3.2 Constructor & Destructor Documentation

3.3.2.1 DIFPtr()

```
DIFPtr::DIFPtr (
    unsigned char * p,
    const std::uint32_t & max_size )
```

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

```
00007                                     : theDIF_(p), theSize_(max_size)
00008 {
00009     theFrames_.clear();
00010     theLines_.clear();
00011     try
00012     {
00013         theGetFramePtrReturn_ = DIFUnpacker::getFramePtr(theFrames_, theLines_, theSize_, theDIF_);
00014     }
00015     catch(std::string e)
00016     {
00017         std::cout << "DIF " << getID() << " T ? " << hasTemperature() << " " << e << std::endl;
00018     }
00019 }
```

3.3.3 Member Function Documentation

3.3.3.1 dumpDIFInfo()

```
void DIFPtr::dumpDIFInfo ( ) [inline]
```

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

```
00041 {
00042     printf("DIF %d DTC %d GTC %d ABCID %lld BCID %d Lines %d Temperature %d \n", getID(), getDTC(),
00043         getGTC(), getAbsoluteBCID(), getBCID(), getLines(), hasTemperature());
00044     if(hasTemperature()) printf("T: ASU1 %d %f ASU2 %d %f DIF %d %f \n", getTASU1(),
00045         getTemperatureASU1(), getTASU2(), getTemperatureASU2(), getTDIF(), getTemperatureDIF());
00046     printf("Found %ld Lines and %ld Frames \n", theLines_.size(), theFrames_.size());
00047 }
```

3.3.3.2 getAbsoluteBCID()

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

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

```
00022 { return DIFUnpacker::getAbsoluteBCID(theDIF_); }
```

3.3.3.3 getASICid()

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

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

```
00049 { return getFrameAsicHeader(i) & 0xFF; }
```

3.3.3.4 getBCID()

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

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

```
00023 { return DIFUnpacker::getBCID(theDIF_); }
```

3.3.3.5 getDIFid()

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

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

```
00048 { return getID() & 0xFF; }
```

3.3.3.6 getDTC()

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

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

```
00020 { return DIFUnpacker::getDTC(theDIF_); }
```

3.3.3.7 getFrameAsicHeader()

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

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

```
00036 { return DIFUnpacker::getFrameAsicHeader(theFrames_[i]); }
```

3.3.3.8 getFrameBCID()

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

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

```
00037 { return DIFUnpacker::getFrameBCID(theFrames_[i]); }
```

3.3.3.9 getFrameLevel()

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

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

```
00039 { return DIFUnpacker::getFrameLevel(theFrames_[i], ipad, ilevel); }
```

3.3.3.10 getFramePtr()

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

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

```
00035 { return theFrames_[i]; }
```

3.3.3.11 getFramesVector()

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

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

```
00017 { return theFrames_; }
```

3.3.3.12 getFrameTimeToTrigger()

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

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

```
00038 { return getBCID() - getFrameBCID(i); }
```


3.3.3.13 getGetFramePtrReturn()

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

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

```
00016 { return theGetFramePtrReturn_; }
```

3.3.3.14 getGTC()

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

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

```
00021 { return DIFUnpacker::getGTC(theDIF_); }
```

3.3.3.15 getID()

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

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

```
00019 { return DIFUnpacker::getID(theDIF_); }
```

3.3.3.16 getLines()

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

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

```
00024 { return DIFUnpacker::getLines(theDIF_); }
```

3.3.3.17 getLinesVector()

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

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

```
00018 { return theLines_; }
```

3.3.3.18 getNumberOfFrames()

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

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

```
00034 { return theFrames_.size(); }
```

3.3.3.19 getPtr()

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

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

```
00015 { return theDIF_; }
```

3.3.3.20 getTASU1()

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

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

```
00026 { return DIFUnpacker::getTASU1(theDIF_); }
```

3.3.3.21 getTASU2()

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

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

```
00027 { return DIFUnpacker::getTASU2(theDIF_); }
```

3.3.3.22 getTDIF()

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

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

```
00028 { return DIFUnpacker::getTDIF(theDIF_); }
```

3.3.3.23 getTemperatureASU1()

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

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

```
00030 { return (getTASU1() » 3) * 0.0625; }
```

3.3.3.24 getTemperatureASU2()

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

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

```
00031 { return (getTASU2() » 3) * 0.0625; }
```

3.3.3.25 getTemperatureDIF()

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

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

```
00029 { return 0.508 * getTDIF() - 9.659; }
```

3.3.3.26 getThresholdStatus()

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

Definition at line 50 of file [DIFPtr.h](#).

```
00050 { return (((uint32_t)getFrameLevel(i, ipad, 1)) < 1) | ((uint32_t)getFrameLevel(i, ipad, 0)); }
```

3.3.3.27 hasAnalogReadout()

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

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

```
00033 { return DIFUnpacker::hasAnalogReadout(theDIF_); }
```

3.3.3.28 hasLine()

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

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

```
00025 { return DIFUnpacker::hasLine(line, theDIF_); }
```

3.3.3.29 hasTemperature()

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

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

```
00032 { return DIFUnpacker::hasTemperature(theDIF_); }
```

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

- [/home/runner/work/streamout/streamout/libs/core/include/DIFPtr.h](#)
- [/home/runner/work/streamout/streamout/libs/core/src/DIFPtr.cc](#)

3.4 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 &asidid)
Get one chip map.
- int [getChipSlowControl](#) (const std::int8_t &asidid, const std::string ¶m)
Get one Chip value.
- void [Dump](#) ()
print out full map

3.4.1 Detailed Description

Handler of DIF Slow Control info.

Author

L.Mirabito

Date

March 2010

Version

1.0

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

3.4.2 Constructor & Destructor Documentation

3.4.2.1 DIFSlowControl()

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

Constructor.

Parameters

<i>version</i>	Data format version
<i>DIFid</i>	DIF id
<i>buf</i>	Pointer to the Raw data buffer

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

```

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

3.4.3 Member Function Documentation

3.4.3.1 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();
00048         it++)
00049     {
00049         std::cout << "ASIC " << it->first << std::endl;
00050         for(std::map<std::string, int>::iterator jt = (it->second).begin(); jt != (it->second).end();
00051             jt++) std::cout << jt->first << " : " << jt->second << std::endl;
00051     }
00052 }
```

3.4.3.2 getChipSlowControl() [1/2]

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

Get one chip map.

Parameters

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

Returns

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

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

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

3.4.3.3 getChipSlowControl() [2/2]

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

Get one Chip value.

Parameters

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

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

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

3.4.3.4 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; }
```

3.4.3.5 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](#)

3.5 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)

3.5.1 Detailed Description

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

3.5.2 Member Function Documentation

3.5.2.1 dumpFrameOld()

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

Definition at line 140 of file [DIFUnpacker.cc](#).

```
00141 {
00142     bool          PAD[128];
00143     bool          l0[64];
00144     bool          l1[64];
00145     std::uint8_t  un{1};
00146     for(std::size_t ip = 0; ip < 128; ip++) { PAD[ip] = false; } // init PADs
00147     std::uint32_t idx1{4};
00148     for(int ik = 0; ik < 4; ik++)
00149     {
00150         std::uint32_t PadEtat{swap_bytes(&buf[idx1])};
00151         idx1 += 4;
00152         for(int e = 0; e < 32; e++)
00153         {
00154             PAD[((3 - ik) * 32) + (31 - e)] = PadEtat & un; // binary operation
00155             PadEtat = PadEtat >> 1; // décalage des bit de 1
00156         }
00157     }
00158     // fill bool arrays
00159     for(int p = 0; p < 64; p++)
00160     {
00161         l0[p] = static_cast<bool>(PAD[(2 * p)]); // _Lev0 (PAD paire)
00162         l1[p] = static_cast<bool>(PAD[(2 * p) + 1]); // _Lev1 (PAD impaires)
00163     }
00164     std::bitset<64> bs0(0);
00165     std::bitset<64> bs1(0);
00166     for(std::uint32_t ip = 0; ip < 64; ip++)
00167     {
00168         bs0.set(ip, l0[ip]);
00169         bs1.set(ip, l1[ip]);
00170     }
00171     std::cout << "\t \t" << bs0 << std::endl;
00172     std::cout << "\t \t" << bs1 << std::endl;
00173 }
```

3.5.2.2 getAbsoluteBCID()

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

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

```
00048 {
00049     std::uint64_t Shift{16777216ULL}; // to shift the value from the 24 first bits
00050     std::uint64_t pos{idx + DU::ABCID_SHIFT};
00051     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]));
00052     return LBC;
00053 }
```


3.5.2.3 getAnalogPtr()

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

Definition at line 86 of file [DIFUnpacker.cc](#).

```
00087 {
00088     std::uint32_t fshift{idx};
00089     if(cb[fshift] != DU::START_OF_LINES) return fshift;
00090     fshift++;
00091     while(cb[fshift] != DU::END_OF_LINES)
00092     {
00093         vLines.push_back(&cb[fshift]);
00094         std::uint32_t nchip{cb[fshift]};
00095         fshift += 1 + nchip * 64 * 2;
00096     }
00097     return fshift++;
00098 }
```

3.5.2.4 getBCID()

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

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

```
00055 { return (cb[idx + DU::BCID_SHIFT] << 16) + (cb[idx + DU::BCID_SHIFT + 1] << 8) + cb[idx +
    DU::BCID_SHIFT + 2]; }
```

3.5.2.5 getDTC()

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

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

```
00043 { 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]; }
```

3.5.2.6 getFrameAsicHeader()

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

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

```
00070 { return (framePtr[DU::FRAME_ASIC_HEADER_SHIFT]); }
```

3.5.2.7 getFrameBCID()

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

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

```
00073 {
00074     std::uint32_t igray = (framePtr[DU::FRAME_BCID_SHIFT] « 16) + (framePtr[DU::FRAME_BCID_SHIFT + 1] «
        8) + framePtr[DU::FRAME_BCID_SHIFT + 2];
00075     return DIFUnpacker::GrayToBin(igray);
00076 }
```

3.5.2.8 getFrameLevel()

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

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

```
00084 { return ((framePtr[DU::FRAME_DATA_SHIFT + ((3 - ip / 16) * 4 + (ip % 16) / 4)] » (7 - ((ip % 16) %
    4) * 2 + level))) & 0x1); }
```

3.5.2.9 getFramePAD()

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

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

```
00079 {
00080     std::uint32_t* iframe{(std::uint32_t*)&framePtr[DU::FRAME_DATA_SHIFT]};
00081     return ((iframe[3 - ip / 32] » (ip % 32)) & 0x1);
00082 }
```

3.5.2.10 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 100 of file [DIFUnpacker.cc](#).

```
00101 {
00102     std::uint32_t fshift{0};
00103     if(DATA_FORMAT_VERSION >= 13)
00104     {
00105         fshift = idx + DU::LINES_SHIFT + 1;
00106         if(DIFUnpacker::hasTemperature(cb, idx)) fshift = idx + DU::TDIF_SHIFT + 1;
        // jenlev 1
    }
```

```

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

```

3.5.2.11 getGTC()

```

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

```

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

```

00045 { 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]; }

```

3.5.2.12 getID()

```

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

```

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

```

00041 { return cb[idx + DU::ID_SHIFT]; }

```

3.5.2.13 getLines()

```

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

```

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

```

00056 { return (cb[idx + DU::LINES_SHIFT] >> 4) & 0x5; }

```

3.5.2.14 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 28 of file [DIFUnpacker.cc](#).

```
00029 {
00030     std::uint32_t id0{0};
00031     for(std::uint32_t i = start; i < size_buf; i++)
00032     {
00033         if(cbuf[i] != DU::START_OF_DIF && cbuf[i] != DU::START_OF_DIF_TEMP) continue;
00034         id0 = i;
00035         // if (cbuf[id0+DU::ID_SHIFT]>0xFF) continue;
00036         break;
00037     }
00038     return id0;
00039 }
```

3.5.2.15 getTASU1()

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

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

```
00060 { 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]; }
```

3.5.2.16 getTASU2()

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

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

```
00062 { 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]; }
```

3.5.2.17 getTDIF()

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

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

```
00064 { return (cb[idx + DU::TDIF_SHIFT]); }
```

3.5.2.18 GrayToBin()

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

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

```
00014 {
00015     std::uint64_t ish{1};
00016     std::uint64_t anss{n};
00017     std::uint64_t idiv{0};
00018     std::uint64_t ishmax{sizeof(std::uint64_t) * 8};
00019     while(true)
00020     {
00021         idiv = anss >> ish;
00022         anss ^= idiv;
00023         if(idiv <= 1 || ish == ishmax) return anss;
00024         ish <<= 1;
00025     }
00026 }
```

3.5.2.19 hasAnalogReadout()

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

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

```
00068 { return (DIFUnpacker::getLines(cb, idx) != 0); }
```

3.5.2.20 hasLine()

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

Definition at line 58 of file [DIFUnpacker.cc](#).

```
00058 { return ((cb[idx + DU::LINES_SHIFT] >> line) & 0x1); }
```

3.5.2.21 hasTemperature()

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

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

```
00066 { return (cb[idx] == DU::START_OF_DIF_TEMP); }
```

3.5.2.22 swap_bytes()

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

Definition at line 175 of file [DIFUnpacker.cc](#).

```
00176 {
00177     unsigned char Swapped[4];
00178     for(std::size_t i = 0; i < 4; i++) Swapped[i] = buf[4 - 1 - i];
00179     return *reinterpret_cast<std::uint32_t*>(&Swapped[0]);
00180 }
```

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](#)

3.6 DU Class Reference

```
#include <Words.h>
```

Static Public Attributes

- static const std::uint32_t [START_OF_DIF](#) {0xB0}
- static const std::uint32_t [START_OF_DIF_TEMP](#) {0xBB}
- static const std::uint32_t [END_OF_DIF](#) {0xA0}
- static const std::uint32_t [START_OF_LINES](#) {0xC4}
- static const std::uint32_t [END_OF_LINES](#) {0xD4}
- static const std::uint32_t [START_OF_FRAME](#) {0xB4}
- static const std::uint32_t [END_OF_FRAME](#) {0xA3}
- static const std::uint32_t [ID_SHIFT](#) {1}
- static const std::uint32_t [DTC_SHIFT](#) {2}
- static const std::uint32_t [GTC_SHIFT](#) {10}
- static const std::uint32_t [ABCID_SHIFT](#) {14}
- static const std::uint32_t [BCID_SHIFT](#) {20}
- static const std::uint32_t [LINES_SHIFT](#) {23}
- static const std::uint32_t [TASU1_SHIFT](#) {24}
- static const std::uint32_t [TASU2_SHIFT](#) {28}
- static const std::uint32_t [TDIF_SHIFT](#) {32}
- static const std::uint32_t [FRAME_ASIC_HEADER_SHIFT](#) {0}
- static const std::uint32_t [FRAME_BCID_SHIFT](#) {1}
- static const std::uint32_t [FRAME_DATA_SHIFT](#) {4}
- static const std::uint32_t [FRAME_SIZE](#) {20}

3.6.1 Detailed Description

Definition at line 7 of file [Words.h](#).

3.6.2 Member Data Documentation

3.6.2.1 ABCID_SHIFT

```
const std::uint32_t DU::ABCID_SHIFT {14} [static]
```

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

3.6.2.2 BCID_SHIFT

```
const std::uint32_t DU::BCID_SHIFT {20} [static]
```

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

3.6.2.3 DTC_SHIFT

```
const std::uint32_t DU::DTC_SHIFT {2} [static]
```

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

3.6.2.4 END_OF_DIF

```
const std::uint32_t DU::END_OF_DIF {0xA0} [static]
```

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

3.6.2.5 END_OF_FRAME

```
const std::uint32_t DU::END_OF_FRAME {0xA3} [static]
```

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

3.6.2.6 END_OF_LINES

```
const std::uint32_t DU::END_OF_LINES {0xD4} [static]
```

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

3.6.2.7 FRAME_ASIC_HEADER_SHIFT

```
const std::uint32_t DU::FRAME_ASIC_HEADER_SHIFT {0} [static]
```

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

3.6.2.8 FRAME_BCID_SHIFT

```
const std::uint32_t DU::FRAME_BCID_SHIFT {1} [static]
```

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

3.6.2.9 FRAME_DATA_SHIFT

```
const std::uint32_t DU::FRAME_DATA_SHIFT {4} [static]
```

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

3.6.2.10 FRAME_SIZE

```
const std::uint32_t DU::FRAME_SIZE {20} [static]
```

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

3.6.2.11 GTC_SHIFT

```
const std::uint32_t DU::GTC_SHIFT {10} [static]
```

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

3.6.2.12 ID_SHIFT

```
const std::uint32_t DU::ID_SHIFT {1} [static]
```

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

3.6.2.13 LINES_SHIFT

```
const std::uint32_t DU::LINES_SHIFT {23} [static]
```

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

3.6.2.14 START_OF_DIF

```
const std::uint32_t DU::START_OF_DIF {0xB0} [static]
```

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

3.6.2.15 START_OF_DIF_TEMP

```
const std::uint32_t DU::START_OF_DIF_TEMP {0xBB} [static]
```

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

3.6.2.16 START_OF_FRAME

```
const std::uint32_t DU::START_OF_FRAME {0xB4} [static]
```

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

3.6.2.17 START_OF_LINES

```
const std::uint32_t DU::START_OF_LINES {0xC4} [static]
```

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

3.6.2.18 TASU1_SHIFT

```
const std::uint32_t DU::TASU1_SHIFT {24} [static]
```

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

3.6.2.19 TASU2_SHIFT

```
const std::uint32_t DU::TASU2_SHIFT {28} [static]
```

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

3.6.2.20 TDIF_SHIFT

```
const std::uint32_t DU::TDIF_SHIFT {32} [static]
```

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

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

- [/home/runner/work/streamout/streamout/libs/core/include/Words.h](#)

3.7 RawdataReader Class Reference

```
#include <RawdataReader.h>
```

Public Member Functions

- [RawdataReader](#) (const char *fileName)
- float [getFileSize](#) ()
- void [openFile](#) (const char *fileName)
- void [closeFile](#) ()
- bool [nextEvent](#) ()
- bool [nextDIFbuffer](#) ()
- [Buffer](#) [getSDHCALBuffer](#) ()
- virtual [~RawdataReader](#) ()

Static Public Member Functions

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

3.7.1 Detailed Description

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

3.7.2 Constructor & Destructor Documentation

3.7.2.1 RawdataReader()

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

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

```
00016 {
00017     m_buf.reserve(m_BufferSize);
00018     openFile(fileName);
00019 }
```

3.7.2.2 ~RawdataReader()

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

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

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

3.7.3 Member Function Documentation

3.7.3.1 closeFile()

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

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

```
00033 {
00034     try
00035     {
00036         if(m_FileStream.is_open()) m_FileStream.close();
00037     }
00038     catch(const std::ios_base::failure& e)
00039     {
00040         std::cout<<"Caught an ios_base::failure in closeFile : "<<e.what()<<" "<<e.code()<<std::endl;
00041         throw;
00042     }
00043 }
```

3.7.3.2 getFileSize()

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

Definition at line 109 of file [RawdataReader.cc](#).

```
00109 { return m_FileSize; }
```

3.7.3.3 getSDHCALBuffer()

Buffer RawdataReader::getSDHCALBuffer ()

Definition at line 105 of file [RawdataReader.cc](#).

```
00105 { return m_Buffer; }
```

3.7.3.4 nextDIFbuffer()

bool RawdataReader::nextDIFbuffer ()

Definition at line 80 of file [RawdataReader.cc](#).

```
00081 {
00082     try
00083     {
00084         static int DIF_processed{0};
00085         if(DIF_processed >= m_NumberOfDIF)
00086         {
00087             DIF_processed = 0;
00088             return false;
00089         }
00090         else
00091         {
00092             DIF_processed++;
00093             std::uint32_t bsize{0};
00094             m_FileStream.read(reinterpret_cast<char*>(&bsize), sizeof(std::uint32_t));
00095             m_FileStream.read(reinterpret_cast<char*>(&m_buf[0]), bsize);
00096         }
00097     }
00098     catch(const std::ios_base::failure& e)
00099     {
00100         return false;
00101     }
00102     return true;
00103 }
```

3.7.3.5 nextEvent()

bool RawdataReader::nextEvent ()

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

```
00067 {
00068     try
00069     {
00070         m_FileStream.read(reinterpret_cast<char*>(&m_EventNumber), sizeof(std::uint32_t));
00071         m_FileStream.read(reinterpret_cast<char*>(&m_NumberOfDIF), sizeof(std::uint32_t));
00072     }
00073     catch(const std::ios_base::failure& e)
00074     {
00075         return false;
00076     }
00077     return true;
00078 }
```

3.7.3.6 openFile()

```
void RawdataReader::openFile (
    const char * fileName )
```

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

```
00046 {
00047     try
00048     {
00049         m_FileStream.rdbuf()->pubsetbuf(0, 0);
00050         m_FileStream.exceptions(std::ifstream::failbit | std::ifstream::badbit);
00051         m_FileStream.open(fileName, std::ios::in | std::ios::binary | std::ios::ate); // Start at the end
00052         to directly calculate the size of the file then come back to beginning
00053         m_FileStream.rdbuf()->pubsetbuf(0, 0);
00054         if(m_FileStream.is_open())
00055         {
00056             setFileSize(m_FileStream.tellg());
00057             m_FileStream.seekg(0, std::ios::beg);
00058         }
00059     } catch(const std::ios_base::failure& e)
00060     {
00061         std::cout<<"Caught an ios_base::failure in openFile : "<<e.what()<<" "<<e.code()<<std::endl;
00062         throw;
00063     }
00064 }
```

3.7.3.7 setDefaultBufferSize()

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

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

```
00013 { m_BufferSize = size; }
```

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](#)

3.8 ROOTtreeDest Class Reference

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

Classes

- struct [DATA](#)

Public Member Functions

- [ROOTtreeDest](#) ()
- void [start](#) ()
- void [processDIF](#) (DIFPtr *)
- void [processFrame](#) (DIFPtr *, std::uint32_t frameIndex)
- void [processPadInFrame](#) (DIFPtr *, std::uint32_t frameIndex, std::uint32_t channelIndex)
- void [processSlowControl](#) (const Buffer &)
- void [end](#) ()

3.8.1 Detailed Description

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

3.8.2 Constructor & Destructor Documentation

3.8.2.1 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/1");  
00013 }
```

3.8.3 Member Function Documentation

3.8.3.1 end()

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

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

```
00030 { ; }
```

3.8.3.2 processDIF()

```
void ROOTtreeDest::processDIF (  
    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 }
```

3.8.3.3 processFrame()

```
void ROOTtreeDest::processFrame (
    DIFPtr * d,
    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 }
```

3.8.3.4 processPadInFrame()

```
void ROOTtreeDest::processPadInFrame (
    DIFPtr * d,
    std::uint32_t frameIndex,
    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 }
```

3.8.3.5 processSlowControl()

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

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

```
00029 { ; }
```

3.8.3.6 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](#)

3.9 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, std::ostream &out=std::cout, bool verbose=false, std::ostream &verbose_out=std::cout)
- void [loop](#) (const std::int32_t &m_NbrEventsToProcess=0)
- void [printAllCounters](#) ()

3.9.1 Detailed Description

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

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

3.9.2 Constructor & Destructor Documentation

3.9.2.1 SDHCAL_buffer_loop()

```
template<typename SOURCE , typename DESTINATION >
SDHCAL_buffer_loop< SOURCE, DESTINATION >::SDHCAL_buffer_loop (
    SOURCE & source,
    DESTINATION & dest,
    bool debug = false,
    std::ostream & out = std::cout,
    bool verbose = false,
    std::ostream & verbose_out = std::cout ) [inline]
```

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

```
00031
00032         m_Source(source), m_Destination(dest), m_Debug(debug), m_DebugOut(out), m_Verbose(verbose),
00033         m_VerboseOut(verbose_out)
00034     {
```

3.9.3 Member Function Documentation

3.9.3.1 loop()

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

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

```
00036 {
00037     m_Destination.start();
00038     while(m_Source.nextEvent() && (m_NbrEventsToProcess == 0 || m_NbrEventsToProcess >= m_NbrEvents))
00039     {
00040         while(m_Source.nextDIFbuffer())
00041         {
00042             Buffer buffer = m_Source.getSDHCALBuffer();
00043             unsigned char* debug_variable_1 = buffer.end();
00044             SDHCAL_RawBuffer_Navigator bufferNavigator(buffer);
00045             unsigned char* debug_variable_2 = bufferNavigator.getDIFBuffer().end();
00046             if(m_Verbose) m_VerboseOut << "DIF BUFFER END " << (unsigned int*)debug_variable_1 << " " <<
00047             (unsigned int*)debug_variable_2 << std::endl;
00048             if(m_Debug) assert(debug_variable_1 == debug_variable_2);
00049             uint32_t idstart = bufferNavigator.getStartOfDIF();
00049             if(m_Debug && idstart == 0) buffer.printBuffer();
00050             c.DIFStarter[idstart]++;
00051             if(!bufferNavigator.validBuffer()) continue;
00052             DIFPtr* d = bufferNavigator.getDIFPtr();
00053             if(m_Debug) assert(d != NULL);
00054             if(d != NULL)
00055             {
00056                 c.DIFPtrValueAtReturnedPos[bufferNavigator.getDIFBufferStart()[d->getGetFramePtrReturn()]]++;
00057                 if(m_Debug) assert(bufferNavigator.getDIFBufferStart()[d->getGetFramePtrReturn()] == 0xa0);
00058             }
00059             c.SizeAfterDIFPtr[bufferNavigator.getSizeAfterDIFPtr()]++;
00060             m_Destination.processDIF(d);
00061             for(uint32_t i = 0; i < d->getNumberOfFrames(); i++)
00062             {
00063                 m_Destination.processFrame(d, i);
00064                 for(uint32_t j = 0; j < 64; j++) m_Destination.processPadInFrame(d, i, j);
00065             }
00066             bool processSC = false;
00067             if(bufferNavigator.hasSlowControlData())
00068             {
00069                 c.hasSlowControl++;
00070                 processSC = true;
00071             }
00072             if(bufferNavigator.badSCData())
00073             {
00074                 c.hasBadSlowControl++;
00075                 processSC = false;
00076             }
00077             if(processSC) { m_Destination.processSlowControl(bufferNavigator.getSCBuffer()); }
00078             Buffer eod = bufferNavigator.getEndOfAllData();
00079             c.SizeAfterAllData[eod.size()]++;
00080             unsigned char* debug_variable_3 = eod.end();
00081             if(m_Verbose) m_VerboseOut << "END DATA BUFFER END " << (unsigned int*)debug_variable_1 << " " <<
00082             (unsigned int*)debug_variable_3 << std::endl;
00083             if(m_Debug) assert(debug_variable_1 == debug_variable_3);
00084             if(m_Verbose)
00085             {
00086                 m_VerboseOut << "End of Data remaining stuff : ";
00087                 eod.printBuffer();
00088             }
00089             int nonzeroCount = 0;
00090             for(unsigned char* it = eod.begin(); it != eod.end(); it++)
00091                 if(static_cast<int>(*it) != 0) nonzeroCount++;
00092             c.NonZeroValusAtEndOfData[nonzeroCount]++;
00093             } // end of DIF while loop
00094             m_NbrEvents++;
00095         } // end of event while loop
00096     }
00097     m_Destination.end();
00098 }
```

3.9.3.2 printAllCounters()

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

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

```
00100 { c.printAllCounters(m_DebugOut); }
```

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

- [/home/runner/work/streamout/streamout/libs/core/include/SDHCAL_buffer_loop.h](#)

3.10 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, std::ostream &out=std::cout)
- void [printAllCounters](#) (std::ostream &out=std::cout)

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](#)

3.10.1 Detailed Description

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

3.10.2 Member Function Documentation

3.10.2.1 printAllCounters()

```
void SDHCAL_buffer_LoopCounter::printAllCounters (
    std::ostream & out = std::cout )
```

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

```
00008 {
00009     out << "BUFFER LOOP FINAL STATISTICS : " << std::endl;
00010     printCounter("Start of DIF header", DIFStarter, out);
00011     printCounter("Value after DIF data are processed", DIFPtrValueAtReturnedPos, out);
00012     printCounter("Size remaining in buffer after end of DIF data", SizeAfterDIFPtr, out);
00013     out << "Number of Slow Control found " << hasSlowControl << " out of which " << hasBadSlowControl << "
are bad" << std::endl;
00014     printCounter("Size remaining after all of data have been processed", SizeAfterAllData, out);
00015     printCounter("Number on non zero values in end of data buffer", NonZeroValusAtEndOfData, out);
00016 }
```

3.10.2.2 printCounter()

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

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

```
00019 {
00020     out << " statistics for " << description << " : ";
00021     for(std::map<int, int>::const_iterator it = m.begin(); it != m.end(); it++)
00022     {
00023         if(it != m.begin()) out << ",";
00024         out << " [" << it->first << "]" = " << it->second;
00025     }
00026     out << std::endl;
00027 }
```

3.10.3 Member Data Documentation

3.10.3.1 DIFPtrValueAtReturnedPos

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

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

3.10.3.2 DIFStarter

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

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

3.10.3.3 hasBadSlowControl

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

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

3.10.3.4 hasSlowControl

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

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

3.10.3.5 NonZeroValusAtEndOfData

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

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

3.10.3.6 SizeAfterAllData

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

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

3.10.3.7 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](#)

3.11 SDHCAL_RawBuffer_Navigator Class Reference

```
#include <SDHCAL_RawBuffer_Navigator.h>
```

Public Member Functions

- [SDHCAL_RawBuffer_Navigator](#) (const [Buffer](#) &b, const int &start=-1)
- [~SDHCAL_RawBuffer_Navigator](#) ()
- 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)

3.11.1 Detailed Description

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

3.11.2 Constructor & Destructor Documentation

3.11.2.1 SDHCAL_RawBuffer_Navigator()

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

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

```
00014                                     : m_Buffer(b)
00015 {
00016     StartAt(start);
00017     m_DIFstartIndex = DIFUnpacker::getStartOfDIF(m_Buffer.begin(), m_Buffer.size(), m_Start);
00018 }
```

3.11.2.2 ~SDHCAL_RawBuffer_Navigator()

```
SDHCAL_RawBuffer_Navigator::~SDHCAL_RawBuffer_Navigator ( )
```

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

```
00021 {
00022     if(m_TheDIFPtr != nullptr) delete m_TheDIFPtr;
00023 }
```

3.11.3 Member Function Documentation

3.11.3.1 badSCData()

```
bool SDHCAL_RawBuffer_Navigator::badSCData ( )
```

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

```
00063 {
00064     setSCBuffer();
00065     return m_BadSCdata;
00066 }
```

3.11.3.2 getDIF_CRC()

```
uint32_t SDHCAL_RawBuffer_Navigator::getDIF_CRC ( )
```

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

```
00046 {
00047     uint32_t i{getEndOfDIFData()};
00048     uint32_t ret{0};
00049     ret |= (m_Buffer.begin()[i - 2]) << 8);
00050     ret |= m_Buffer.begin()[i - 1];
00051     return ret;
00052 }
```

3.11.3.3 getDIFBuffer()

```
Buffer SDHCAL_RawBuffer_Navigator::getDIFBuffer ( )
```

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

```
00033 { return Buffer(getDIFBufferStart(), getDIFBufferSize()); }
```

3.11.3.4 getDIFBufferSize()

```
std::uint32_t SDHCAL_RawBuffer_Navigator::getDIFBufferSize ( )
```

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

```
00031 { return m_Buffer.size() - m_DIFstartIndex; }
```

3.11.3.5 getDIFBufferStart()

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

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

```
00029 { return &(m_Buffer.begin()[m_DIFstartIndex]); }
```

3.11.3.6 getDIFPtr()

```
DIFPtr * SDHCAL_RawBuffer_Navigator::getDIFPtr ( )
```

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

```
00036 {
00037     if(m_TheDIFPtr == nullptr) m_TheDIFPtr = new DIFPtr(getDIFBufferStart(), getDIFBufferSize());
00038     return m_TheDIFPtr;
00039 }
```

3.11.3.7 getEndOfAllData()

Buffer SDHCAL_RawBuffer_Navigator::getEndOfAllData ()

Definition at line 101 of file SDHCAL_RawBuffer_Navigator.cc.

```

00102 {
00103     setSCBuffer();
00104     if(hasSlowControlData() && !m_BadSCdata) { return Buffer(&(m_SCbuffer.begin()[m_SCbuffer.size()]),
00105         getSizeAfterDIFPtr() - 3 - m_SCbuffer.size()); }
00105     else
00106         return Buffer(&(getDIFBufferStart()[getEndOfDIFData()]), getSizeAfterDIFPtr() - 3); // remove the
00107         2 bytes for CRC and the DIF trailer
00107 }

```

3.11.3.8 getEndOfDIFData()

std::uint32_t SDHCAL_RawBuffer_Navigator::getEndOfDIFData ()

Definition at line 41 of file SDHCAL_RawBuffer_Navigator.cc.

```

00041 { return getDIFPtr()->getGetFramePtrReturn() + 3; }

```

3.11.3.9 getSCBuffer()

Buffer SDHCAL_RawBuffer_Navigator::getSCBuffer ()

Definition at line 56 of file SDHCAL_RawBuffer_Navigator.cc.

```

00057 {
00058     setSCBuffer();
00059     return m_SCbuffer;
00060 }

```

3.11.3.10 getSizeAfterDIFPtr()

std::uint32_t SDHCAL_RawBuffer_Navigator::getSizeAfterDIFPtr ()

Definition at line 43 of file SDHCAL_RawBuffer_Navigator.cc.

```

00043 { return getDIFBufferSize() - getDIFPtr()->getGetFramePtrReturn(); }

```

3.11.3.11 getStartOfDIF()

std::uint32_t SDHCAL_RawBuffer_Navigator::getStartOfDIF ()

Definition at line 27 of file SDHCAL_RawBuffer_Navigator.cc.

```

00027 { return m_DIFstartIndex; }

```

3.11.3.12 hasSlowControlData()

```
bool SDHCAL_RawBuffer_Navigator::hasSlowControlData ( )
```

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

```
00054 { return getDIFBufferStart()[getEndOfDIFData()] == 0xb1; }
```

3.11.3.13 StartAt()

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

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

```
00010 {
00011     if(start >= 0) m_Start = start;
00012 }
```

3.11.3.14 validBuffer()

```
bool SDHCAL_RawBuffer_Navigator::validBuffer ( )
```

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

```
00025 { return m_DIFstartIndex != 0; }
```

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

- [/home/runner/work/streamout/streamout/libs/core/include/SDHCAL_RawBuffer_Navigator.h](#)
- [/home/runner/work/streamout/streamout/libs/core/src/SDHCAL_RawBuffer_Navigator.cc](#)

3.12 textDump Class Reference

```
#include <textDump.h>
```

Public Member Functions

- [textDump](#) (std::ostream &out=std::cout)
- void [start](#) ()
- void [processDIF](#) (DIFPtr *)
- void [processFrame](#) (DIFPtr *, uint32_t frameIndex)
- void [processPadInFrame](#) (DIFPtr *, uint32_t frameIndex, uint32_t channelIndex)
- void [processSlowControl](#) (Buffer)
- void [end](#) ()

3.12.1 Detailed Description

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

3.12.2 Constructor & Destructor Documentation

3.12.2.1 textDump()

```
textDump::textDump (
    std::ostream & out = std::cout ) [inline], [explicit]
```

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

```
00016 : _out(out) { ; }
```

3.12.3 Member Function Documentation

3.12.3.1 end()

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

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

```
00038 { _out << "textDump end of report" << std::endl; }
```

3.12.3.2 processDIF()

```
void textDump::processDIF (
    DIFPtr * d )
```

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

```
00012 {
00013     if(NULL == d) return;
00014     _out << "DIF number is " << d->getDIFid() << std::endl;
00015     _out << " DTC value is " << d->getDTC() << std::endl;
00016     _out << " GTC value is " << d->getGTC() << std::endl;
00017     _out << " DIF BCID is " << d->getBCID() << std::endl;
00018     _out << " Absolute BCID is " << d->getAbsoluteBCID() << std::endl;
00019     _out << " The number of frame is " << d->getNumberOfFrames() << std::endl;
00020 }
```

3.12.3.3 processFrame()

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

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

```
00023 {
00024     _out << " Displaying frame number " << frameIndex << std::endl;
00025     _out << " ASIC ID is " << d->getASICid(frameIndex) << std::endl;
00026     _out << " Frame BCID is " << d->getFrameBCID(frameIndex) << std::endl;
00027     _out << " Frame Time To Trigger (a.k.a timestamp) is " << d->getFrameTimeToTrigger(frameIndex) <<
        std::endl;
00028 }
```

3.12.3.4 processPadInFrame()

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

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

```
00031 {
00032     _out << "    Displaying channel number " << channelIndex << std::endl;
00033     _out << "    Threshold status is " << d->getThresholdStatus(frameIndex, channelIndex) << std::endl;
00034 }
```

3.12.3.5 processSlowControl()

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

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

```
00036 { _out << "textDump::processSlowControl not implemented yet." << std::endl; }
```

3.12.3.6 start()

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

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

```
00009 { _out << "Will dump bunch of DIF data" << std::endl; }
```

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](#)

Chapter 4

File Documentation

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

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

Typedefs

- using [bit8_t](#) = std::uint8_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.

4.1.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

4.1.2 Typedef Documentation

4.1.2.1 bit8_t

using `bit8_t` = `std::uint8_t`

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

4.1.3 Function Documentation

4.1.3.1 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 10 of file [Bits.cc](#).

```
00010 { return os << c + 0; }
```

4.2 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
00013 std::ostream& operator<<(std::ostream& os, const bit8_t& c);
```

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

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

Classes

- class [Buffer](#)

4.4 Buffer.h

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

```

00001
00006 #pragma once
00007
00008 #include "Bits.h"
00009
00010 #include <array>
00011 #include <iostream>
00012 #include <vector>
00013
00014 class Buffer
00015 {
00016 public:
00017     Buffer() : m_Buffer(nullptr), m_Size(0), m_Capacity(0) {}
00018     Buffer(const bit8_t b[], const std::size_t& i) : m_Buffer(const_cast<bit8_t*>(&b[0])), m_Size(i),
m_Capacity(i) {}
00019     Buffer(const char b[], const std::size_t& i) : m_Buffer(const_cast<bit8_t*>(reinterpret_cast<const
bit8_t*>(&b[0])), m_Size(i), m_Capacity(i) {}
00020     template<typename T> Buffer(const std::vector<T>& rawdata) :
m_Buffer(const_cast<bit8_t*>(reinterpret_cast<const bit8_t*>(rawdata.data()))), m_Size(rawdata.size()
* sizeof(T)), m_Capacity(rawdata.capacity() * sizeof(T)) {}
00021     template<typename T, std::size_t N> Buffer(const std::array<T, N>& rawdata) :
m_Buffer(const_cast<bit8_t*>(reinterpret_cast<const bit8_t*>(rawdata.data()))), m_Size(rawdata.size()
* sizeof(T)), m_Capacity(rawdata.size() * sizeof(T)) {}
00022
00023     std::size_t size() const { return m_Size; }
00024     std::size_t capacity() const { return m_Capacity; }
00025
00026     void set(unsigned char* b) { m_Buffer = b; }
00027     bit8_t* begin() { return m_Buffer; }
00028     bit8_t* end() { return m_Buffer + m_Size; }
00029     bit8_t& operator[](const std::size_t& pos) { return m_Buffer[pos]; }
00030
00031     void setSize(const std::size_t& size) { m_Size = size; }
00032     void printBuffer(uint32_t start, uint32_t stop, std::ostream& flux = std::cout);
00033     void printBuffer(uint32_t start = 0, std::ostream& flux = std::cout) { printBuffer(start, size()); }
00034     virtual ~Buffer();
00035
00036 private:
00037     bit8_t* m_Buffer{nullptr};
00038     std::size_t m_Size{0};
00039     std::size_t m_Capacity{0};
00040 };

```

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

```

#include "DIFUnpacker.h"
#include <iostream>
#include <vector>

```

Classes

- class [DIFPtr](#)

4.5.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

4.6 DIFPtr.h

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

```

00001
00005 #pragma once
00006 #include "DIFUnpacker.h"
00007
00008 #include <iostream>
00009 #include <vector>
00010
00011 class DIFPtr
00012 {
00013 public:
00014     DIFPtr(unsigned char* p, const std::uint32_t& max_size);
00015     inline unsigned char* getPtr() { return theDIF_; }
00016     inline std::uint32_t getGetFramePtrReturn() { return theGetFramePtrReturn_; }
00017     inline std::vector<unsigned char*> getFramesVector() { return theFrames_; }
00018     inline std::vector<unsigned char*> getLinesVector() { return theLines_; }
00019     inline std::uint32_t getID() { return DIFUnpacker::getID(theDIF_); }
00020     inline std::uint32_t getDTC() { return DIFUnpacker::getDTC(theDIF_); }
00021     inline std::uint32_t getGTC() { return DIFUnpacker::getGTC(theDIF_); }
00022     inline std::uint64_t getAbsoluteBCID() { return
DIFUnpacker::getAbsoluteBCID(theDIF_); }
00023     inline std::uint32_t getBCID() { return DIFUnpacker::getBCID(theDIF_); }
00024     inline std::uint32_t getLines() { return DIFUnpacker::getLines(theDIF_); }
00025     inline bool hasLine(uint32_t line) { return DIFUnpacker::hasLine(line,
theDIF_); }
00026     inline std::uint32_t getTASU1() { return DIFUnpacker::getTASU1(theDIF_); }
00027     inline std::uint32_t getTASU2() { return DIFUnpacker::getTASU2(theDIF_); }
00028     inline std::uint32_t getTDIF() { return DIFUnpacker::getTDIF(theDIF_); }
00029     inline float getTemperatureDIF() { return 0.508 * getTDIF() - 9.659; }
00030     inline float getTemperatureASU1() { return (getTASU1() >> 3) * 0.0625; }
00031     inline float getTemperatureASU2() { return (getTASU2() >> 3) * 0.0625; }
00032     inline bool hasTemperature() { return DIFUnpacker::hasTemperature(theDIF_); }
00033     inline bool hasAnalogReadout() { return
DIFUnpacker::hasAnalogReadout(theDIF_); }
00034     inline std::uint32_t getNumberOfFrames() { return theFrames_.size(); }
00035     inline unsigned char* getFramePtr(uint32_t i) { return theFrames_[i]; }
00036     inline std::uint32_t getFrameAsicHeader(uint32_t i) { return
DIFUnpacker::getFrameAsicHeader(theFrames_[i]); }
00037     inline std::uint32_t getFrameBCID(uint32_t i) { return
DIFUnpacker::getFrameBCID(theFrames_[i]); }
00038     inline std::uint32_t getFrameTimeToTrigger(uint32_t i) { return getBCID() -
getFrameBCID(i); }
00039     inline bool getFrameLevel(uint32_t i, uint32_t ipad, uint32_t ilevel) {
return DIFUnpacker::getFrameLevel(theFrames_[i], ipad, ilevel); }
00040     void dumpDIFInfo()
00041     {
00042         printf("DIF %d DTC %d GTC %d ABCID %lld BCID %d Lines %d Temperature %d \n", getID(), getDTC(),
getGTC(), getAbsoluteBCID(), getBCID(), getLines(), hasTemperature());
00043
00044         if(hasTemperature()) printf("T: ASU1 %d %f ASU2 %d %f DIF %d %f \n", getTASU1(),
getTemperatureASU1(), getTASU2(), getTemperatureASU2(), getTDIF(), getTemperatureDIF());
00045         printf("Found %ld Lines and %ld Frames \n", theLines_.size(), theFrames_.size());
00046     }
00047     // Addition by GG
00048     inline uint32_t getDIFid() { return getID() & 0xFF; }
00049     inline uint32_t getASICid(uint32_t i) { return getFrameAsicHeader(i) & 0xFF; }
00050     inline uint32_t getThresholdStatus(uint32_t i, uint32_t ipad) { return (((uint32_t)getFrameLevel(i,
ipad, 1)) << 1) | ((uint32_t)getFrameLevel(i, ipad, 0))); }
00051
00052 private:
00053     std::uint32_t theSize_;
00054     std::uint32_t theGetFramePtrReturn_;
00055     unsigned char* theDIF_;
00056     std::vector<unsigned char*> theFrames_;
00057     std::vector<unsigned char*> theLines_;
00058 };

```

4.7 /home/runner/work/streamout/streamout/libs/core/include/DIFSlowControl.h File Reference

```

#include <bitset>
#include <cstdint>
#include <iostream>

```

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

Classes

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

4.7.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

4.8 DIFSlowControl.h

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

```
00001
00005 #pragma once
00006
00007 #include <bitset>
00008 #include <cstdint>
00009 #include <iostream>
00010 #include <map>
00011 #include <string>
00020 class DIFSlowControl
00021 {
00022 public:
00024
00029 DIFSlowControl(const std::uint8_t& version, const std::uint8_t& DIFid, unsigned char* buf);
00030
00032 inline std::uint8_t getDIFid();
00033
00035
00038 inline std::map<int, std::map<std::string, int>> getChipsMap();
00039
00041
00045 inline std::map<std::string, int> getChipSlowControl(const int& asicid);
00046
00048
00052 inline int getChipSlowControl(const std::int8_t& asicid, const std::string& param);
00053
00055 void Dump();
00056
00057 private:
00059 DIFSlowControl() = delete;
00061 void FillHR1(const int& header_shift, unsigned char* cbuf);
00063 void FillHR2(const int& header_shift, unsigned char* cbuf);
00065 void FillAsicHR1(const std::bitset<72 * 8>& bs);
00067 void FillAsicHR2(const std::bitset<109 * 8>& bs);
00068
00069 unsigned int m_DIFid{0};
00070 unsigned int m_Version{0};
00071 unsigned int m_AsicType{0}; // asicType_
00072 unsigned int m_NbrAsic{0};
00073 std::map<int, std::map<std::string, int>> m_MapSC;
00074 };
```

4.9 /home/runner/work/streamout/streamout/libs/core/include/↵ DIFUnpacker.h File Reference

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

Classes

- class [DIFUnpacker](#)

4.9.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

4.10 DIFUnpacker.h

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

```
00001
00005 #pragma once
00006
00007 #include <cstdint>
00008 #include <iostream>
00009 #include <vector>
00010
00011 class DIFUnpacker
00012 {
00013 public:
00014     static std::uint64_t GrayToBin(const std::uint64_t& n);
00015     static std::uint32_t getStartOfDIF(const unsigned char* cbuf, const std::uint32_t& size_buf, const
std::uint32_t& start = 92);
00016     static std::uint32_t getID(const unsigned char* cb, const std::uint32_t& idx = 0);
00017     static std::uint32_t getDTC(const unsigned char* cb, const std::uint32_t& idx = 0);
00018     static std::uint32_t getGTC(const unsigned char* cb, const std::uint32_t& idx = 0);
00019     static std::uint64_t getAbsoluteBCID(const unsigned char* cb, const std::uint32_t& idx = 0);
00020     static std::uint32_t getBCID(const unsigned char* cb, const std::uint32_t& idx = 0);
00021     static std::uint32_t getLines(const unsigned char* cb, const std::uint32_t& idx = 0);
00022     static bool hasLine(const std::uint32_t& line, const unsigned char* cb, const
std::uint32_t& idx = 0);
00023     static std::uint32_t getTASU1(const unsigned char* cb, const std::uint32_t& idx = 0);
00024     static std::uint32_t getTASU2(const unsigned char* cb, const std::uint32_t& idx = 0);
00025     static std::uint32_t getTDIF(const unsigned char* cb, const std::uint32_t& idx = 0);
00026     static bool hasTemperature(const unsigned char* cb, const std::uint32_t& idx = 0);
00027     static bool hasAnalogReadout(const unsigned char* cb, const std::uint32_t& idx = 0);
00028
00029     static std::uint32_t getFrameAsicHeader(const unsigned char* framePtr);
00030     static std::uint32_t getFrameBCID(const unsigned char* framePtr);
00031
00032     static bool getFramePAD(const unsigned char* framePtr, const std::uint32_t& ip);
00033     static bool getFrameLevel(const unsigned char* framePtr, const std::uint32_t& ip, const
std::uint32_t& level);
00034
00035     static std::uint32_t getAnalogPtr(std::vector<unsigned char*>& vLines, unsigned char* cb, const
std::uint32_t& idx = 0);
00036     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);
00037     static void dumpFrameOld(const unsigned char* buf);
00038     static std::uint32_t swap_bytes(const unsigned char* buf); // Stolen from DCBufferReader
00039 };
```


4.11 /home/runner/work/streamout/streamout/libs/core/include/SDHCAL_buffer_loop.h File Reference

```
#include "Buffer.h"
#include "SDHCAL_RawBuffer_Navigator.h"
#include "SDHCAL_buffer_LoopCounter.h"
#include <cassert>
#include <iostream>
#include <ostream>
```

Classes

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

4.11.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

4.12 SDHCAL_buffer_loop.h

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

```
00001
00005 #pragma once
00006
00007 #include "Buffer.h"
00008 #include "SDHCAL_RawBuffer_Navigator.h"
00009 #include "SDHCAL_buffer_LoopCounter.h"
00010
00011 #include <cassert>
00012 #include <iostream>
00013 #include <ostream>
00014 // function to loop on buffers
00015 //
00016 // template class should implement
00017 // bool SOURCE::next();
00018 // SDHCAL_buffer SOURCE::getSDHCALBuffer();
00019 //
00020 // void DESTINATION::start();
00021 // void DESTINATION::processDIF(DIFPtr*);
00022 // void DESTINATION::processFrame(DIFPtr*,uint32_t frameIndex);
00023 // void DESTINATION::processPadInFrame(DIFPtr*,uint32_t frameIndex, uint32_t channelIndex);
00024 // void DESTINATION::processSlowControl(SDHCAL_buffer);
00025 // void DESTINATION::end();
00026 //
00027
00028 template<typename SOURCE, typename DESTINATION> class SDHCAL_buffer_loop
00029 {
00030 public:
00031     SDHCAL_buffer_loop(SOURCE& source, DESTINATION& dest, bool debug = false, std::ostream& out =
std::cout, bool verbose = false, std::ostream& verbose_out = std::cout) :
00032         m_Source(source), m_Destination(dest), m_Debug(debug), m_DebugOut(out), m_Verbose(verbose),
m_VerboseOut(verbose_out)
00033     {
00034     }
00035     void loop(const std::int32_t& m_NbrEventsToProcess = 0)
00036     {
00037         m_Destination.start();
00038         while(m_Source.nextEvent() && (m_NbrEventsToProcess == 0 || m_NbrEventsToProcess >= m_NbrEvents))
```

```

00039     {
00040         while(m_Source.nextDIFbuffer())
00041         {
00042             Buffer                buffer                = m_Source.getSDHCALBuffer();
00043             unsigned char*        debug_variable_1 = buffer.end();
00044             SDHCAL_RawBuffer_Navigator bufferNavigator(buffer);
00045             unsigned char*        debug_variable_2 = bufferNavigator.getDIFBuffer().end();
00046             if(m_Verbose) m_VerboseOut << "DIF BUFFER END " << (unsigned int*)debug_variable_1 << " " <<
(unsigned int*)debug_variable_2 << std::endl;
00047             if(m_Debug) assert(debug_variable_1 == debug_variable_2);
00048             uint32_t idstart = bufferNavigator.getStartOfDIF();
00049             if(m_Debug && idstart == 0) buffer.printBuffer();
00050             c.DIFStarter[idstart]++;
00051             if(!bufferNavigator.validBuffer()) continue;
00052             DIFPtr* d = bufferNavigator.getDIFPtr();
00053             if(m_Debug) assert(d != NULL);
00054             if(d != NULL)
00055             {
00056                 c.DIFPtrValueAtReturnedPos[bufferNavigator.getDIFBufferStart() [d->getGetFramePtrReturn()]]++;
00057                 if(m_Debug) assert(bufferNavigator.getDIFBufferStart() [d->getGetFramePtrReturn()] == 0xa0);
00058             }
00059             c.SizeAfterDIFPtr[bufferNavigator.getSizeAfterDIFPtr() ]++;
00060             m_Destination.processDIF(d);
00061             for(uint32_t i = 0; i < d->getNumberOfFrames(); i++)
00062             {
00063                 m_Destination.processFrame(d, i);
00064                 for(uint32_t j = 0; j < 64; j++) m_Destination.processPadInFrame(d, i, j);
00065             }
00066
00067             bool processSC = false;
00068             if(bufferNavigator.hasSlowControlData())
00069             {
00070                 c.hasSlowControl++;
00071                 processSC = true;
00072             }
00073             if(bufferNavigator.badSCData())
00074             {
00075                 c.hasBadSlowControl++;
00076                 processSC = false;
00077             }
00078             if(processSC) { m_Destination.processSlowControl(bufferNavigator.getSCBuffer()); }
00079
00080             Buffer eod = bufferNavigator.getEndOfAllData();
00081             c.SizeAfterAllData[eod.size() ]++;
00082             unsigned char* debug_variable_3 = eod.end();
00083             if(m_Verbose) m_VerboseOut << "END DATA BUFFER END " << (unsigned int*)debug_variable_1 << " " <<
(unsigned int*)debug_variable_3 << std::endl;
00084             if(m_Debug) assert(debug_variable_1 == debug_variable_3);
00085             if(m_Verbose)
00086             {
00087                 m_VerboseOut << "End of Data remaining stuff : ";
00088                 eod.printBuffer();
00089             }
00090
00091             int nonzeroCount = 0;
00092             for(unsigned char* it = eod.begin(); it != eod.end(); it++)
00093                 if(static_cast<int>(*it) != 0) nonzeroCount++;
00094             c.NonZeroValusAtEndOfData[nonzeroCount]++;
00095             } // end of DIF while loop
00096             m_NbrEvents++;
00097             } // end of event while loop
00098             m_Destination.end();
00099         }
00100         void printAllCounters() { c.printAllCounters(m_DebugOut); }
00101
00102     private:
00103         SDHCAL_buffer_LoopCounter c;
00104         SOURCE&                    m_Source;
00105         DESTINATION&                m_Destination;
00106         bool                       m_Debug{false};
00107         std::ostream&                m_DebugOut{std::cout};
00108         bool                       m_Verbose{false};
00109         std::ostream&                m_VerboseOut{std::cout};
00110         std::uint32_t               m_NbrEvents{1};
00111     };

```

4.13 /home/runner/work/streamout/streamout/libs/core/include/↵ SDHCAL_buffer_LoopCounter.h File Reference

```
#include <iostream>
#include <map>
#include <string>
```

Classes

- struct [SDHCAL_buffer_LoopCounter](#)

4.13.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

4.14 SDHCAL_buffer_LoopCounter.h

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

```
00001
00005 #pragma once
00006
00007 #include <iostream>
00008 #include <map>
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, std::ostream& out =
std::cout);
00023     void printAllCounters(std::ostream& out = std::cout);
00024 };
```

4.15 /home/runner/work/streamout/streamout/libs/core/include/↵ SDHCAL_RawBuffer_Navigator.h File Reference

```
#include "Buffer.h"
#include "DIFPtr.h"
#include <iostream>
```

Classes

- class [SDHCAL_RawBuffer_Navigator](#)

4.15.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

4.16 SDHCAL_RawBuffer_Navigator.h

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

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

4.17 /home/runner/work/streamout/streamout/libs/core/include/Words.h

File Reference

Classes

- class [DU](#)

4.17.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

4.18 Words.h

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

```
00001
00005 #pragma once
00006
00007 class DU
00008 {
00009 public:
00010     static const std::uint32_t START_OF_DIF{0xB0};
00011     static const std::uint32_t START_OF_DIF_TEMP{0xBB};
00012     static const std::uint32_t END_OF_DIF{0xA0};
00013     static const std::uint32_t START_OF_LINES{0xC4};
00014     static const std::uint32_t END_OF_LINES{0xD4};
00015
00016     static const std::uint32_t START_OF_FRAME{0xB4};
00017     static const std::uint32_t END_OF_FRAME{0xA3};
00018
00019     static const std::uint32_t ID_SHIFT{1};
00020     static const std::uint32_t DTC_SHIFT{2};
00021     static const std::uint32_t GTC_SHIFT{10};
00022     static const std::uint32_t ABCID_SHIFT{14};
00023     static const std::uint32_t BCID_SHIFT{20};
00024     static const std::uint32_t LINES_SHIFT{23};
00025     static const std::uint32_t TASU1_SHIFT{24};
00026     static const std::uint32_t TASU2_SHIFT{28};
00027     static const std::uint32_t TDIF_SHIFT{32};
00028
00029     static const std::uint32_t FRAME_ASIC_HEADER_SHIFT{0};
00030     static const std::uint32_t FRAME_BCID_SHIFT{1};
00031     static const std::uint32_t FRAME_DATA_SHIFT{4};
00032     static const std::uint32_t FRAME_SIZE{20};
00033 };
```

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

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

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`.

4.19.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

4.19.2 Function Documentation

4.19.2.1 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 10 of file [Bits.cc](#).

```
00010 { return os << c + 0; }
```

4.20 Bits.cc

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

```
00001
00006 #include "Bits.h"
00007
00008 #include <iostream>
00009
00010 std::ostream& operator<<(std::ostream& os, const bit8_t& c) { return os << c + 0; }
```

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

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

4.22 Buffer.cc

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

```
00001
00006 #include "Buffer.h"
00007
00008 void Buffer::printBuffer(unsigned int start, unsigned int stop, std::ostream& flux)
00009 {
00010     flux << std::hex;
00011     for(unsigned int k = start; k < stop; k++) flux << (unsigned int)(m_Buffer[k]) << " - ";
00012     flux << std::dec << std::endl;
00013 }
00014
00015 Buffer::~Buffer() { std::cout << "SDHCAL_buffer destructor called" << std::endl; }
```

4.23 /home/runner/work/streamout/streamout/libs/core/src/DIFPtr.cc File Reference

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

4.24 DIFPtr.cc

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

```

00001
00005 #include "DIFPtr.h"
00006
00007 DIFPtr::DIFPtr(unsigned char* p, const std::uint32_t& max_size) : theDIF_(p), theSize_(max_size)
00008 {
00009     theFrames_.clear();
00010     theLines_.clear();
00011     try
00012     {
00013         theGetFramePtrReturn_ = DIFUnpacker::getFramePtr(theFrames_, theLines_, theSize_, theDIF_);
00014     }
00015     catch(std::string e)
00016     {
00017         std::cout << "DIF " << getID() << " T ? " << hasTemperature() << " " << e << std::endl;
00018     }
00019 }
```

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

```

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

4.25.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

4.26 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*
00011     cbuf) : m_Version(version), m_DIFId(DIFId), m_AsicType(2)
00012 {
00013     if(cbuf[0] != 0xb1) return;
00014     int header_shift{6};
00015     if(m_Version < 8) m_NbrAsic = cbuf[5];
00016     else
00017     {
00018         m_DIFId = cbuf[1];
00019         m_NbrAsic = cbuf[2];
00020         header_shift = 3;
00021     }
00022     int size_hardroc1 = m_NbrAsic * 72 + header_shift + 1;
00023     if(cbuf[size_hardroc1 - 1] != 0xal) size_hardroc1 = 0;
00024     int size_hardroc2 = m_NbrAsic * 109 + header_shift + 1;
00025     if(cbuf[size_hardroc2 - 1] != 0xal) size_hardroc2 = 0;
00026     if(size_hardroc1 != 0)
```

```

00027 {
00028     FillHR1(header_shift, cbuf);
00029     m_AsicType = 1;
00030 }
00031 else if(size_hardroc2 != 0)
00032     FillHR2(header_shift, cbuf);
00033 else
00034     return;
00035 }
00036
00037 inline std::uint8_t DIFSlowControl::getDIFId() { return m_DIFId; }
00038
00039 inline std::map<int, std::map<std::string, int> DIFSlowControl::getChipsMap() { return m_MapSC; }
00040
00041 inline std::map<std::string, int> DIFSlowControl::getChipSlowControl(const int& asicid) { return
    m_MapSC[asicid]; }
00042
00043 inline int DIFSlowControl::getChipSlowControl(const std::int8_t& asicid, const std::string& param) {
    return getChipSlowControl(asicid)[param]; }
00044
00045 void DIFSlowControl::Dump()
00046 {
00047     for(std::map<int, std::map<std::string, int>::iterator it = m_MapSC.begin(); it != m_MapSC.end();
        it++)
00048     {
00049         std::cout << "ASIC " << it->first << std::endl;
00050         for(std::map<std::string, int>::iterator jt = (it->second).begin(); jt != (it->second).end();
            jt++) std::cout << jt->first << " : " << jt->second << std::endl;
00051     }
00052 }
00053
00054 void DIFSlowControl::FillHR1(const int& header_shift, unsigned char* cbuf)
00055 {
00056     int nasic{cbuf[header_shift - 1]};
00057     int idx{header_shift};
00058     for(int k = 0; k < nasic; k++)
00059     {
00060         std::bitset<72 * 8> bs;
00061         // printf("%x %x \n",cbuf[idx+k*72+69],cbuf[idx+k*72+70]);
00062         for(int l = 71; l >= 0; l--)
00063         {
00064             // printf("%d %x : %d -->",l,cbuf[idx+k*72+1],(71-l)*8);
00065             for(int m = 0; m < 8; m++)
00066             {
00067                 if(((1 < m) & cbuf[idx + k * 72 + 1]) != 0) bs.set((71 - l) * 8 + m, 1);
00068                 else
00069                     bs.set((71 - l) * 8 + m, 0);
00070                 // printf("%d", (int) bs[(71-l)*8+m]);
00071             }
00072             // printf("\n");
00073         }
00074         FillAsicHR1(bs);
00075     }
00076 }
00077
00078 void DIFSlowControl::FillHR2(const int& header_shift, unsigned char* cbuf)
00079 {
00080     // int scsizer=cbuf[header_shift-1]*109+(header_shift-1)+2;
00081     int nasic{cbuf[header_shift - 1]};
00082     int idx{header_shift};
00083     // std::cout<<" DIFSlowControl::FillHR nasic "<<nasic<<std::endl;
00084     for(int k = 0; k < nasic; k++)
00085     {
00086         std::bitset<109 * 8> bs;
00087         // printf("%x %x \n",cbuf[idx+k*109+69],cbuf[idx+k*109+70]);
00088         for(int l = 108; l >= 0; l--)
00089         {
00090             // printf("%d %x : %d -->",l,cbuf[idx+k*109+1],(71-l)*8);
00091             for(int m = 0; m < 8; m++)
00092             {
00093                 if(((1 < m) & cbuf[idx + k * 109 + 1]) != 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_Discr"] = 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>(bs[16]);
00160     mAsic["EN_Out_Discr"] = static_cast<int>(bs[8]);
00161     mAsic["EN_Transmit_On"] = static_cast<int>(bs[7]);
00162     mAsic["EN_Dout"] = static_cast<int>(bs[6]);
00163     mAsic["EN_RamFull"] = static_cast<int>(bs[5]);
00164     m_MapSC[asid] = mAsic;
00165 }
00166
00167 void DIFSlowControl::FillAsicHR2(const std::bitset<109 * 8>& bs)
00168 {
00169     int asid{0};
00170     for(int j = 0; j < 8; j++)
00171         if(bs[j + (108 - 7) * 8 + 2] != 0) asid += (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["Sw100f1"] = 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 }

```

4.27 /home/runner/work/streamout/streamout/libs/core/src/↵ DIFUnpacker.cc File Reference

```

#include "DIFUnpacker.h"
#include "Words.h"
#include <bitset>
#include <cstdint>

```

```
#include <iostream>
```

4.27.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

4.28 DIFUnpacker.cc

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

```
00001
00005 #include "DIFUnpacker.h"
00006
00007 #include "Words.h"
00008
00009 #include <bitset>
00010 #include <cstdint>
00011 #include <iostream>
00012
00013 std::uint64_t DIFUnpacker::GrayToBin(const std::uint64_t& n)
00014 {
00015     std::uint64_t ish{1};
00016     std::uint64_t anss{n};
00017     std::uint64_t idiv{0};
00018     std::uint64_t ishmax{sizeof(std::uint64_t) * 8};
00019     while(true)
00020     {
00021         idiv = anss » ish;
00022         anss ^= idiv;
00023         if(idiv <= 1 || ish == ishmax) return anss;
00024         ish «= 1;
00025     }
00026 }
00027
00028 std::uint32_t DIFUnpacker::getStartOfDIF(const unsigned char* cbuf, const std::uint32_t& size_buf,
00029     const std::uint32_t& start)
00030 {
00031     std::uint32_t id0{0};
00032     for(std::uint32_t i = start; i < size_buf; i++)
00033     {
00034         if(cbuf[i] != DU::START_OF_DIF && cbuf[i] != DU::START_OF_DIF_TEMP) continue;
00035         id0 = i;
00036         // if (cbuf[id0+DU::ID_SHIFT]>0xFF) continue;
00037         break;
00038     }
00039     return id0;
00040 }
00041
00042 std::uint32_t DIFUnpacker::getID(const unsigned char* cb, const std::uint32_t& idx) { return cb[idx +
00043     DU::ID_SHIFT]; }
00044
00045 std::uint32_t DIFUnpacker::getDTC(const unsigned char* cb, const std::uint32_t& idx) { return (cb[idx
00046     + DU::DTC_SHIFT] « 24) + (cb[idx + DU::DTC_SHIFT + 1] « 16) + (cb[idx + DU::DTC_SHIFT + 2] « 8) +
00047     cb[idx + DU::DTC_SHIFT + 3]; }
00048
00049 std::uint32_t DIFUnpacker::getGTC(const unsigned char* cb, const std::uint32_t& idx) { return (cb[idx
00050     + DU::GTC_SHIFT] « 24) + (cb[idx + DU::GTC_SHIFT + 1] « 16) + (cb[idx + DU::GTC_SHIFT + 2] « 8) +
00051     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] «
00058     16) | (cb[pos + 4] « 8) | (cb[pos + 5]));
00059     return LBC;
00060 }
```

```

00055 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]; }
00056 std::uint32_t DIFUnpacker::getLines(const unsigned char* cb, const std::uint32_t& idx) { return
(cb[idx + DU::LINES_SHIFT] >> 4) & 0x5; }
00057
00058 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); }
00059
00060 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]; }
00061
00062 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]; }
00063
00064 std::uint32_t DIFUnpacker::getTDIF(const unsigned char* cb, const std::uint32_t& idx) { return (cb[idx
+ DU::TDIF_SHIFT]); }
00065
00066 bool DIFUnpacker::hasTemperature(const unsigned char* cb, const std::uint32_t& idx) { return (cb[idx]
== DU::START_OF_DIF_TEMP); }
00067
00068 bool DIFUnpacker::hasAnalogReadout(const unsigned char* cb, const std::uint32_t& idx) { return
(DIFUnpacker::getLines(cb, idx) != 0); }
00069
00070 std::uint32_t DIFUnpacker::getFrameAsicHeader(const unsigned char* framePtr) { return
(framePtr[DU::FRAME_ASIC_HEADER_SHIFT]); }
00071
00072 std::uint32_t DIFUnpacker::getFrameBCID(const unsigned char* framePtr)
00073 {
00074     std::uint32_t igray = (framePtr[DU::FRAME_BCID_SHIFT] << 16) + (framePtr[DU::FRAME_BCID_SHIFT + 1] <<
8) + framePtr[DU::FRAME_BCID_SHIFT + 2];
00075     return DIFUnpacker::GrayToBin(igray);
00076 }
00077
00078 bool DIFUnpacker::getFramePAD(const unsigned char* framePtr, const std::uint32_t& ip)
00079 {
00080     std::uint32_t* iframe{(std::uint32_t*)&framePtr[DU::FRAME_DATA_SHIFT]};
00081     return ((iframe[3 - ip / 32] >> (ip % 32)) & 0x1);
00082 }
00083
00084 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); }
00085
00086 std::uint32_t DIFUnpacker::getAnalogPtr(std::vector<unsigned char*>& vLines, unsigned char* cb, const
std::uint32_t& idx)
00087 {
00088     std::uint32_t fshift{idx};
00089     if(cb[fshift] != DU::START_OF_LINES) return fshift;
00090     fshift++;
00091     while(cb[fshift] != DU::END_OF_LINES)
00092     {
00093         vLines.push_back(&cb[fshift]);
00094         std::uint32_t nchip{cb[fshift]};
00095         fshift += 1 + nchip * 64 * 2;
00096     }
00097     return fshift++;
00098 }
00099
00100 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)
00101 {
00102     std::uint32_t fshift{0};
00103     if(DATA_FORMAT_VERSION >= 13)
00104     {
00105         fshift = idx + DU::LINES_SHIFT + 1;
00106         if(DIFUnpacker::hasTemperature(cb, idx)) fshift = idx + DU::TDIF_SHIFT + 1;
00107         // jenlev 1
00108         if(DIFUnpacker::hasAnalogReadout(cb, idx)) fshift = DIFUnpacker::getAnalogPtr(vLines, cb, fshift);
00109         // to be implemented
00110     }
00111     else
00112     {
00113         std::uint32_t fshift = idx + DU::BCID_SHIFT + 3;
00114         if(cb[fshift] != DU::START_OF_FRAME)
00115         {
00116             std::cout << "This is not a start of frame " << cb[fshift] << "\n";
00117             return fshift;
00118         }
00119         do {
00120             // printf("fshift %d and %d \n", fshift, max_size);
00121             if(cb[fshift] == DU::END_OF_DIF) return fshift;
00122             if(cb[fshift] == DU::START_OF_FRAME) fshift++;
00123             if(cb[fshift] == DU::END_OF_FRAME)
00124             {
00125                 fshift++;
00126                 continue;
00127             }
00128         } while(fshift < max_size);
00129     }
00130 }

```

```

00124     }
00125     std::uint32_t header = DIFUnpacker::getFrameAsicHeader(&cb[fshift]);
00126     if(header == DU::END_OF_FRAME) return (fshift + 2);
00127     // std::cout<<header<<" "<<fshift<<std::endl;
00128     if(header < 1 || header > 48) { throw header + " Header problem " + fshift; }
00129     vFrame.push_back(&cb[fshift]);
00130     fshift += DU::FRAME_SIZE;
00131     if(fshift > max_size)
00132     {
00133         std::cout << "fshift " << fshift << " exceed " << max_size << "\n";
00134         return fshift;
00135     }
00136     if(cb[fshift] == DU::END_OF_FRAME) fshift++;
00137 } while(true);
00138 }
00139
00140 void DIFUnpacker::dumpFrameOld(const unsigned char* buf)
00141 {
00142     bool PAD[128];
00143     bool l0[64];
00144     bool l1[64];
00145     std::uint8_t un{1};
00146     for(std::size_t ip = 0; ip < 128; ip++) { PAD[ip] = false; } // init PADs
00147     std::uint32_t idx1{4};
00148     for(int ik = 0; ik < 4; ik++)
00149     {
00150         std::uint32_t PadEtat{swap_bytes(&buf[idx1])};
00151         idx1 += 4;
00152         for(int e = 0; e < 32; e++)
00153         {
00154             PAD[(3 - ik) * 32 + (31 - e)] = PadEtat & un; // binary operation
00155             PadEtat = PadEtat >> 1; // décalage des bit de 1
00156         }
00157     }
00158     // fill bool arrays
00159     for(int p = 0; p < 64; p++)
00160     {
00161         l0[p] = static_cast<bool>(PAD[(2 * p)]); // _Lev0 (PAD paire)
00162         l1[p] = static_cast<bool>(PAD[(2 * p) + 1]); // _Lev1 (PAD impaires)
00163     }
00164     std::bitset<64> bs0(0);
00165     std::bitset<64> bs1(0);
00166     for(std::uint32_t ip = 0; ip < 64; ip++)
00167     {
00168         bs0.set(ip, l0[ip]);
00169         bs1.set(ip, l1[ip]);
00170     }
00171     std::cout << "\t \t" << bs0 << std::endl;
00172     std::cout << "\t \t" << bs1 << std::endl;
00173 }
00174
00175 std::uint32_t DIFUnpacker::swap_bytes(const unsigned char* buf)
00176 {
00177     unsigned char Swapped[4];
00178     for(std::size_t i = 0; i < 4; i++) Swapped[i] = buf[4 - 1 - i];
00179     return *reinterpret_cast<std::uint32_t*>(&Swapped[0]);
00180 }

```

4.29 /home/runner/work/streamout/streamout/libs/core/src/SDHCAL_buffer_LoopCounter.cc File Reference

```
#include "SDHCAL_buffer_LoopCounter.h"
```

4.29.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

4.30 SDHCAL_buffer_LoopCounter.cc

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

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

4.31 /home/runner/work/streamout/streamout/libs/core/src/SDHCAL_RawBuffer_Navigator.cc File Reference

```
#include "SDHCAL_RawBuffer_Navigator.h"
```

4.31.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

4.32 SDHCAL_RawBuffer_Navigator.cc

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

```
00001
00005 #include "SDHCAL_RawBuffer_Navigator.h"
00006
00007 int SDHCAL_RawBuffer_Navigator::m_Start = 92;
00008
00009 void SDHCAL_RawBuffer_Navigator::StartAt(const int& start)
00010 {
00011     if(start >= 0) m_Start = start;
00012 }
00013
00014 SDHCAL_RawBuffer_Navigator::SDHCAL_RawBuffer_Navigator(const Buffer& b, const int& start) :
m_Buffer(b)
00015 {
00016     StartAt(start);
00017     m_DIFstartIndex = DIFUnpacker::getStartOfDIF(m_Buffer.begin(), m_Buffer.size(), m_Start);
00018 }
00019
```

```

00020 SDHCAL_RawBuffer_Navigator::~SDHCAL_RawBuffer_Navigator()
00021 {
00022     if(m_TheDIFPtr != nullptr) delete m_TheDIFPtr;
00023 }
00024
00025 bool SDHCAL_RawBuffer_Navigator::validBuffer() { return m_DIFstartIndex != 0; }
00026
00027 std::uint32_t SDHCAL_RawBuffer_Navigator::getStartOfDIF() { return m_DIFstartIndex; }
00028
00029 unsigned char* SDHCAL_RawBuffer_Navigator::getDIFBufferStart() { return
    &(m_Buffer.begin()[m_DIFstartIndex]); }
00030
00031 std::uint32_t SDHCAL_RawBuffer_Navigator::getDIFBufferSize() { return m_Buffer.size() -
    m_DIFstartIndex; }
00032
00033 Buffer SDHCAL_RawBuffer_Navigator::getDIFBuffer() { return Buffer(getDIFBufferStart(),
    getDIFBufferSize()); }
00034
00035 DIFPtr* SDHCAL_RawBuffer_Navigator::getDIFPtr()
00036 {
00037     if(m_TheDIFPtr == nullptr) m_TheDIFPtr = new DIFPtr(getDIFBufferStart(), getDIFBufferSize());
00038     return m_TheDIFPtr;
00039 }
00040
00041 std::uint32_t SDHCAL_RawBuffer_Navigator::getEndOfDIFData() { return
    getDIFPtr()->getGetFramePtrReturn() + 3; }
00042
00043 std::uint32_t SDHCAL_RawBuffer_Navigator::getSizeAfterDIFPtr() { return getDIFBufferSize() -
    getDIFPtr()->getGetFramePtrReturn(); }
00044
00045 uint32_t SDHCAL_RawBuffer_Navigator::getDIF_CRC()
00046 {
00047     uint32_t i{getEndOfDIFData()};
00048     uint32_t ret{0};
00049     ret |= (m_Buffer.begin()[i - 2]) << 8;
00050     ret |= m_Buffer.begin()[i - 1];
00051     return ret;
00052 }
00053
00054 bool SDHCAL_RawBuffer_Navigator::hasSlowControlData() { return getDIFBufferStart()[getEndOfDIFData()]
    == 0xb1; }
00055
00056 Buffer SDHCAL_RawBuffer_Navigator::getSCBuffer()
00057 {
00058     setSCBuffer();
00059     return m_SCbuffer;
00060 }
00061
00062 bool SDHCAL_RawBuffer_Navigator::badSCData()
00063 {
00064     setSCBuffer();
00065     return m_BadSCdata;
00066 }
00067
00068 void SDHCAL_RawBuffer_Navigator::setSCBuffer()
00069 {
00070     if(!hasSlowControlData()) return;
00071     if(m_SCbuffer.size() != 0) return; // deja fait
00072     if(m_BadSCdata) return;
00073     m_SCbuffer.set(&(getDIFBufferStart()[getEndOfDIFData()]));
00074     // compute Slow Control size
00075     std::size_t maxsize{m_Buffer.size() - m_DIFstartIndex - getEndOfDIFData() + 1}; // should I +1 here
    ?
00076     uint32_t k{1}; // SC Header
00077     uint32_t dif_ID{m_SCbuffer[1]};
00078     uint32_t chipSize{m_SCbuffer[3]};
00079     while((dif_ID != 0x1 && m_SCbuffer[k] != 0x1 && k < maxsize) || (dif_ID == 0x1 && m_SCbuffer[k +
    2] == chipSize && k < maxsize))
    {
00080     {
00081         k += 2; // DIF ID + ASIC Header
00082         uint32_t scsize = m_SCbuffer[k];
00083         if(scsize != 74 && scsize != 109)
00084         {
00085             std::cout << "PROBLEM WITH SC SIZE " << scsize << std::endl;
00086             k = 0;
00087             m_BadSCdata = true;
00088             break;
00089         }
00090         k++; // skip size bit
00091         k += scsize; // skip the data
00092     }
00093     if(m_SCbuffer[k] == 0x1 && !m_BadSCdata) m_SCbuffer.setSize(k + 1); // add the trailer
00094     else
00095     {
00096         m_BadSCdata = true;
00097         std::cout << "PROBLEM SC TRAILER NOT FOUND " << std::endl;
00098     }
    }

```

```

00099 }
00100
00101 Buffer SDHCAL_RawBuffer_Navigator::getEndOfAllData()
00102 {
00103     setSCBuffer();
00104     if(hasSlowControlData() && !m_BadSCdata) { return Buffer(&(m_SCbuffer.begin()[m_SCbuffer.size()]),
00105         getSizeAfterDIFPtr() - 3 - m_SCbuffer.size()); }
00106     else
00107         return Buffer(&(getDIFBufferStart()[getEndOfDIFData()]), getSizeAfterDIFPtr() - 3); // remove the
00108         2 bytes for CRC and the DIF trailer
00109 }

```

4.33 /home/runner/work/streamout/streamout/libs/interface/↵ Dump/include/textDump.h File Reference

```

#include "Buffer.h"
#include "DIFPtr.h"
#include <iostream>
#include <ostream>

```

Classes

- class [textDump](#)

4.33.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

4.34 textDump.h

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

```

00001
00005 #pragma once
00006
00007 #include "Buffer.h"
00008 #include "DIFPtr.h"
00009
00010 #include <iostream>
00011 #include <ostream>
00012
00013 class textDump
00014 {
00015 public:
00016     explicit textDump(std::ostream& out = std::cout) : _out(out) { ; }
00017     void start();
00018     void processDIF(DIFPtr*);
00019     void processFrame(DIFPtr*, uint32_t frameIndex);
00020     void processPadInFrame(DIFPtr*, uint32_t frameIndex, uint32_t channelIndex);
00021     void processSlowControl(Buffer);
00022     void end();
00023
00024 private:
00025     std::ostream& _out;
00026 };

```


4.35 /home/runner/work/streamout/streamout/libs/interface/Dump/src/textDump.cc File Reference

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

4.35.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

4.36 textDump.cc

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

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

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

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

Classes

- class [RawdataReader](#)

4.37.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

4.38 RawdataReader.h

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

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

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

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

4.39.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

4.40 RawdataReader.cc

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

```
00001
00004 #include "RawdataReader.h"
00005
00006 #include <cstring>
00007 #include <stdexcept>
00008 #include <zlib.h>
00009
00011 std::size_t RawdataReader::m_BufferSize = 0x100000;
00012
00013 void RawdataReader::setDefaultBufferSize(const std::size_t& size) { m_BufferSize = size; }
00014
00015 RawdataReader::RawdataReader(const char* fileName)
00016 {
00017     m_buf.reserve(m_BufferSize);
00018     openFile(fileName);
00019 }
00020
00021 void RawdataReader::uncompress()
00022 {
00023     // static const std::size_t size_buffer{0x20000};
00024     /*std::size_t shift{3*sizeof(uint32_t)+sizeof(uint64_t)};
00025     static bit8_t obuf[size_buffer];
00026     std::size_t size_buffer_end;
00027     int rc=:uncompress(obuf,&size_buffer_end,
00028 &m_Buffer.at<bit8_t>(shift),m_Buffer.size<bit8_t>()-shift);
00029 memcpy(&m_Buffer.at<bit8_t>(shift),obuf,size_buffer_end);
00030 m_Buffer.setSize(size_buffer_end+shift);*/
00031 }
00032 void RawdataReader::closeFile()
00033 {
00034     try
00035     {
00036         if(m_FileStream.is_open()) m_FileStream.close();
00037     }
00038     catch(const std::ios_base::failure& e)
00039     {
00040         std::cout<<"Caught an ios_base::failure in closeFile : "<<e.what()<<" "<<e.code()<<std::endl;
00041         throw;
00042     }
00043 }
00044
00045 void RawdataReader::openFile(const char* fileName)
00046 {
00047     try
00048     {
00049         m_FileStream.rdbuf()->pubsetbuf(0, 0);
00050         m_FileStream.exceptions(std::ifstream::failbit | std::ifstream::badbit);
00051         m_FileStream.open(fileName, std::ios::in | std::ios::binary | std::ios::ate); // Start at the end
00052         to directly calculate the size of the file then come back to beginning
00053         m_FileStream.rdbuf()->pubsetbuf(0, 0);
00054         if(m_FileStream.is_open())
00055         {
00056             setFileSize(m_FileStream.tellg());
00057             m_FileStream.seekg(0, std::ios::beg);
00058         }
00059     }
00060     catch(const std::ios_base::failure& e)
00061     {
00062         std::cout<<"Caught an ios_base::failure in openFile : "<<e.what()<<" "<<e.code()<<std::endl;
00063         throw;
00064     }
00065 }
00066 bool RawdataReader::nextEvent()
00067 {
```

```

00068     try
00069     {
00070         m_FileStream.read(reinterpret_cast<char*>(&m_EventNumber), sizeof(std::uint32_t));
00071         m_FileStream.read(reinterpret_cast<char*>(&m_NumberOfDIF), sizeof(std::uint32_t));
00072     }
00073     catch(const std::ios_base::failure& e)
00074     {
00075         return false;
00076     }
00077     return true;
00078 }
00079
00080 bool RawdataReader::nextDIFbuffer()
00081 {
00082     try
00083     {
00084         static int DIF_processed{0};
00085         if(DIF_processed >= m_NumberOfDIF)
00086         {
00087             DIF_processed = 0;
00088             return false;
00089         }
00090         else
00091         {
00092             DIF_processed++;
00093             std::uint32_t bsize{0};
00094             m_FileStream.read(reinterpret_cast<char*>(&bsize), sizeof(std::uint32_t));
00095             m_FileStream.read(reinterpret_cast<char*>(&m_buf[0]), bsize);
00096         }
00097     }
00098     catch(const std::ios_base::failure& e)
00099     {
00100         return false;
00101     }
00102     return true;
00103 }
00104
00105 Buffer RawdataReader::getSDHCALBuffer() { return m_Buffer; }
00106
00107 void RawdataReader::setFileSize(const std::size_t& size) { m_FileSize = size; }
00108
00109 float RawdataReader::getFileSize() { return m_FileSize; }

```

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

```

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

```

Classes

- class [ROOTtreeDest](#)
- struct [ROOTtreeDest::DATA](#)

4.41.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

4.42 ROOTtreeDest.h

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

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

4.43 /home/runner/work/streamout/streamout/libs/interface/ROOT/src/ROOTtreeDest.cc File Reference

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

4.43.1 Detailed Description

Copyright

2022 G.Grenier F.Lagarde

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

4.44 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,
00013         "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(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(DIFPtr* d, 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(DIFPtr* d, std::uint32_t frameIndex, std::uint32_t channelIndex)
00042 {
00043     _data.CHANNELid = channelIndex;
00044     _data.Thresh = d->getThresholdStatus(frameIndex, channelIndex);
00045     if(_data.Thresh != 0) _tree->Fill();
00046 }
```