TLC CHAIN

v.0.1

Generated by Doxygen 1.12.0

1 Topic Index 1

1.1 Topics . . . 1

2 Data Structure Index 3

2.1 Data Structures . . . 3

3 File Index 5

3.1 File List . . . 5

4 Topic Documentation 7

4.1 TLC_CHAIN . . . 7

4.1.1 Detailed Description . . . 7

5 Data Structure Documentation 9

5.1 _byte_stream_t Struct Reference . . . 9

5.1.1 Field Documentation . . . 9

5.1.1.1 id . . . 9

5.1.1.2 len . . . 9

5.2 _cc_encoder_info_t Struct Reference . . . 9

5.2.1 Field Documentation . . . 10

5.2.1.1 puncVect . . . 10

5.3 _cc_hard_dec_info_t Struct Reference . . . 10

5.3.1 Field Documentation . . . 10

5.3.1.1 dist . . . 10

5.3.1.2 path . . . 10

5.4 _cc_par_t Struct Reference . . . 11

5.4.1 Field Documentation . . . 11

5.4.1.1 kLen . . . 11

5.4.1.2 memFact . . . 11

5.4.1.3 vitDM . . . 11

5.5 _cc_soft_dec_info_t Struct Reference . . . 11

5.5.1 Field Documentation . . . 12

5.5.1.1 dist . . . 12

5.5.1.2 path . . . 12

5.6 _cc_trcore_t Struct Reference . . . 12

5.6.1 Field Documentation . . . 12

5.6.1.1 nextSt . . . 12

5.7 _cc_trellis_t Struct Reference . . . 12

5.8 _chan_par_t Struct Reference . . . 13

5.8.1 Field Documentation . . . 13

5.8.1.1 EbN0 . . . 13

5.9 _complex_stream_t Struct Reference . . . 13

5.9.1 Field Documentation . . . 13

5.9.1.1 id . . . 13

5.9.1.2 len . . . 14
5.10 _complex_t Struct Reference . . . 14
5.10.1 Field Documentation . . . 14
5.10.1.1 im . . . 14
5.11 _debug_par_t Struct Reference . . . 14
5.12 _float_stream_t Struct Reference . . . 15
5.12.1 Field Documentation . . . 15
5.12.1.1 id . . . 15
5.12.1.2 len . . . 15
5.13 _mod_maptable_t Struct Reference . . . 15
5.14 _mod_par_t Struct Reference . . . 15
5.14.1 Field Documentation . . . 16
5.14.1.1 bps . . . 16
5.14.1.2 order . . . 16
5.14.1.3 phOfst . . . 16

6 File Documentation **17**

6.1 C:/Users/ValmoFil/Music/extra/tlc_chain/src/channel.c File Reference . . . 17
6.1.1 Detailed Description . . . 17
6.1.2 Function Documentation . . . 18
6.1.2.1 Channel_AWGN() . . . 18
6.1.2.2 Channel_BSC() . . . 18
6.1.2.3 Channel_ListParameters() . . . 19
6.1.2.4 GetComplexSgnPower() . . . 19
6.2 C:/Users/ValmoFil/Music/extra/tlc_chain/src/channel.h File Reference . . . 20
6.2.1 Detailed Description . . . 21
6.2.2 Enumeration Type Documentation . . . 21
6.2.2.1 channel_t . . . 21
6.2.3 Function Documentation . . . 21
6.2.3.1 Channel_AWGN() . . . 21
6.2.3.2 Channel_BSC() . . . 22
6.2.3.3 Channel_ListParameters() . . . 22
6.3 channel.h . . . 23
6.4 C:/Users/ValmoFil/Music/extra/tlc_chain/src/convolutional.c File Reference . . . 24
6.4.1 Detailed Description . . . 25
6.4.2 Function Documentation . . . 25
6.4.2.1 CnvCod_Encoder() . . . 25
6.4.2.2 CnvCod_HardDecoder() . . . 26
6.4.2.3 CnvCod_ListParameters() . . . 27
6.4.2.4 CnvCod_SoftDecoder() . . . 28
6.4.2.5 ComputeEncBit() . . . 29
6.4.2.6 ComputeTrellisDiagram() . . . 29

6.4.2.7 CountByteOnes() . . . 30
6.4.2.8 EstimateEuclideanDist() . . . 30
6.4.2.9 FindMinSurvPathHard() . . . 31
6.4.2.10 FindMinSurvPathSoft() . . . 31
6.4.2.11 HardDepuncturer() . . . 31
6.4.2.12 IsKlenValid() . . . 32
6.4.2.13 IsRateValid() . . . 32
6.4.2.14 RetrieveConnectorPuncturationVectors() . . . 32
6.4.2.15 SoftDepuncturer() . . . 33
6.4.3 Variable Documentation . . . 33
6.4.3.1 CC_RATE_ARRAY . . . 33
6.5 C:/Users/ValmoFil/Music/extra/tlc_chain/src/convolutional.h File Reference . . . 34
6.5.1 Detailed Description . . . 35
6.5.2 Macro Definition Documentation . . . 35
6.5.2.1 CC_CVMATRIX . . . 35
6.5.2.2 CC_RATE . . . 36
6.5.2.3 CC_VDM_STR . . . 36
6.5.2.4 DEF_CC_RATES_ARRAY . . . 36
6.5.2.5 DEF_CC_RATES_IDX . . . 36
6.5.2.6 DEF_CC_RATES_VAL . . . 36
6.5.2.7 LIST_OF_CC_RATES . . . 37
6.5.3 Enumeration Type Documentation . . . 37
6.5.3.1 cc_dec_method_t . . . 37
6.5.3.2 cc_klen_t . . . 37
6.5.4 Function Documentation . . . 38
6.5.4.1 CnvCod_Encoder() . . . 38
6.5.4.2 CnvCod_HardDecoder() . . . 38
6.5.4.3 CnvCod_ListParameters() . . . 39
6.5.4.4 CnvCod_SoftDecoder() . . . 40
6.6 convolutional.h . . . 41
6.7 C:/Users/ValmoFil/Music/extra/tlc_chain/src/debug.c File Reference . . . 43
6.7.1 Detailed Description . . . 44
6.7.2 Function Documentation . . . 44
6.7.2.1 Debug_CheckWrongBits() . . . 44
6.7.2.2 Debug_GenerateRandomBytes() . . . 44
6.7.2.3 Debug_ListParameters() . . . 45
6.7.2.4 Debug_PrintByteStream() . . . 45
6.7.2.5 Debug_PrintComplexStream() . . . 45
6.7.2.6 Debug_PrintFloatStream() . . . 46
6.7.2.7 Debug_PrintParameters() . . . 46
6.7.2.8 Debug_WriteByteStreamToCsv() . . . 46
6.7.2.9 Debug_WriteComplexStreamToCsv() . . . 47

6.7.2.10 IsSrcLenValid() . . . 47
6.8 C:/Users/ValmoFil/Music/extra/tlc_chain/src/debug.h File Reference . . . 48
6.8.1 Detailed Description . . . 49
6.8.2 Macro Definition Documentation . . . 49
6.8.2.1 PID_NCOLS_BYTE . . . 49
6.8.3 Enumeration Type Documentation . . . 49
6.8.3.1 print_label_t . . . 49
6.8.4 Function Documentation . . . 50
6.8.4.1 Debug_CheckWrongBits() . . . 50
6.8.4.2 Debug_GenerateRandomBytes() . . . 50
6.8.4.3 Debug_ListParameters() . . . 51
6.8.4.4 Debug_PrintByteStream() . . . 51
6.8.4.5 Debug_PrintComplexStream() . . . 51
6.8.4.6 Debug_PrintFloatStream() . . . 52
6.8.4.7 Debug_PrintParameters() . . . 52
6.8.4.8 Debug_WriteByteStreamToCsv() . . . 52
6.8.4.9 Debug_WriteComplexStreamToCsv() . . . 53
6.9 debug.h . . . 53
6.10 C:/Users/ValmoFil/Music/extra/tlc_chain/src/error.c File Reference . . . 54
6.10.1 Detailed Description . . . 54
6.10.2 Function Documentation . . . 55
6.10.2.1 Error_HandleErr() . . . 55
6.11 C:/Users/ValmoFil/Music/extra/tlc_chain/src/error.h File Reference . . . 55
6.11.1 Detailed Description . . . 56
6.11.2 Enumeration Type Documentation . . . 56
6.11.2.1 alarm_t . . . 56
6.11.2.2 error_t . . . 56
6.11.3 Function Documentation . . . 57
6.11.3.1 Error_HandleErr() . . . 57
6.12 error.h . . . 58
6.13 C:/Users/ValmoFil/Music/extra/tlc_chain/src/main.c File Reference . . . 58
6.13.1 Detailed Description . . . 59
6.13.2 Macro Definition Documentation . . . 60
6.13.2.1 DEF_STREAM_ALLOCATE . . . 60
6.13.2.2 DEF_STREAM_DECLARE . . . 60
6.13.2.3 DEF_STREAM_FREE . . . 60
6.13.2.4 LEN_CC_PUN_BY . . . 60
6.13.2.5 LIST_OF_STREAMS . . . 60
6.13.3 Function Documentation . . . 61
6.13.3.1 main() . . . 61
6.14 C:/Users/ValmoFil/Music/extra/tlc_chain/src/memory.c File Reference . . . 62
6.14.1 Detailed Description . . . 62

6.14.2 Function Documentation . . . 62
6.14.2.1 AllocateByteStream() . . . 62
6.14.2.2 AllocateComplexStream() . . . 63
6.14.2.3 AllocateFloatStream() . . . 63
6.14.2.4 FreeByteStream() . . . 63
6.14.2.5 FreeComplexStream() . . . 64
6.14.2.6 FreeFloatStream() . . . 64
6.14.2.7 Memory_AllocateStream() . . . 64
6.14.2.8 Memory_FreeStream() . . . 65
6.15 memory.h . . . 65
6.16 C:/Users/ValmoFil/Music/extra/tlc_chain/src/modulation.c File Reference . . . 66
6.16.1 Detailed Description . . . 67
6.16.2 Function Documentation . . . 67
6.16.2.1 GetGraySequence() . . . 67
6.16.2.2 GetMappingTable() . . . 67
6.16.2.3 GetPskTable() . . . 68
6.16.2.4 GetQamTable() . . . 68
6.16.2.5 IsQamBpsValid() . . . 68
6.16.2.6 Modulation_HardDemapper() . . . 69
6.16.2.7 Modulation_ListParameters() . . . 69
6.16.2.8 Modulation_Mapper() . . . 69
6.16.2.9 Modulation_SoftDemapper() . . . 70
6.17 C:/Users/ValmoFil/Music/extra/tlc_chain/src/modulation.h File Reference . . . 70
6.17.1 Detailed Description . . . 71
6.17.2 Macro Definition Documentation . . . 72
6.17.2.1 MOD_TYPE_STR . . . 72
6.17.3 Enumeration Type Documentation . . . 72
6.17.3.1 modulation_t . . . 72
6.17.4 Function Documentation . . . 72
6.17.4.1 Modulation_HardDemapper() . . . 72
6.17.4.2 Modulation_ListParameters() . . . 73
6.17.4.3 Modulation_Mapper() . . . 73
6.17.4.4 Modulation_SoftDemapper() . . . 73
6.18 modulation.h . . . 74
6.19 C:/Users/ValmoFil/Music/extra/tlc_chain/src/system.h File Reference . . . 75
6.19.1 Detailed Description . . . 76
6.19.2 Macro Definition Documentation . . . 76
6.19.2.1 BI2BY_LEN . . . 76
6.19.2.2 BY2BI_LEN . . . 76
6.19.2.3 BY2BI_SHIFT . . . 77
6.20 system.h . . . 77

Chapter 1

Topic Index

1.1 Topics

Here is a list of all topics with brief descriptions:

TLC_CHAIN 7

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

_byte_stream_t 9
_cc_encoder_info_t 9
_cc_hard_dec_info_t 10
_cc_par_t 11
_cc_soft_dec_info_t 11
_cc_trcore_t 12
_cc_trellis_t 12
_chan_par_t 13
_complex_stream_t 13
_complex_t 14
_debug_par_t 14
_float_stream_t 15
_mod_maptable_t 15
_mod_par_t 15

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

C:/Users/ValmoFil/Music/extra/tlc_chain/src/channel.c
Channel library 17
C:/Users/ValmoFil/Music/extra/tlc_chain/src/channel.h
Channel library header 20
C:/Users/ValmoFil/Music/extra/tlc_chain/src/convolutional.c
Convolutional coding library 24
C:/Users/ValmoFil/Music/extra/tlc_chain/src/convolutional.h
Convolutional coding library header 34
C:/Users/ValmoFil/Music/extra/tlc_chain/src/debug.c
Debug library 43
C:/Users/ValmoFil/Music/extra/tlc_chain/src/debug.h
Debug library header 48
C:/Users/ValmoFil/Music/extra/tlc_chain/src/error.c
Error library 54
C:/Users/ValmoFil/Music/extra/tlc_chain/src/error.h
Error handling library header 55
C:/Users/ValmoFil/Music/extra/tlc_chain/src/main.c
Main file 58
C:/Users/ValmoFil/Music/extra/tlc_chain/src/memory.c
Memory ibrary 62
C:/Users/ValmoFil/Music/extra/tlc_chain/src/memory.h 65
C:/Users/ValmoFil/Music/extra/tlc_chain/src/modulation.c
Modulation library 66
C:/Users/ValmoFil/Music/extra/tlc_chain/src/modulation.h
Modulation library header 70
C:/Users/ValmoFil/Music/extra/tlc_chain/src/system.h
Dynamic memory allocation library header 75

Chapter 4

Topic Documentation

4.1 TLC_CHAIN

Files

- file channel.c
  Channel library.
- file channel.h
  Channel library header.
- file convolutional.c
  Convolutional coding library.
- file convolutional.h
  Convolutional coding library header.
- file debug.c
  Debug library.
- file debug.h
  Debug library header.
- file error.c
  Error library.
- file error.h
  Error handling library header.
- file main.c
  Main file.
- file memory.c
  Memory ibrary.
- file system.h
  Dynamic memory allocation library header.
- file modulation.c
  Modulation library.
- file modulation.h
  Modulation library header.

4.1.1 Detailed Description

Main library containing all DVB-S telecommunication chain functions.

Chapter 5

Data Structure Documentation

5.1 _byte_stream_t Struct Reference

Data Fields

- byte_t * **pBuf**
- len_t len
- memory_type_t id

5.1.1 Field Documentation

5.1.1.1 id

```
memory_type_t id
```

- buffer length [B]

5.1.1.2 len

```
len_t len
```

- buffer pointer

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

- C:/Users/ValmoFil/Music/extra/tlc_chain/src/memory.h

5.2 _cc_encoder_info_t Struct Reference

Data Fields

- uint8_t **connVect** [CC_NBRANCHES]
- uint8_t puncVect [CC_PUNCTLEN]

5.2.1 Field Documentation

5.2.1.1 puncVect

```
uint8_t puncVect[CC_PUNCTLEN]
```

- connector vector

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

- C:/Users/ValmoFil/Music/extra/tlc_chain/src/convolutional.h

5.3 _cc_hard_dec_info_t Struct Reference

Data Fields

- len_t **iter** [CC_NTRELSTATES]
- uint32_t dist [CC_NTRELSTATES]
- uint8_t path [CC_NTRELSTATES][CC_MEM_DIM]

5.3.1 Field Documentation

5.3.1.1 dist

```
uint32_t dist[CC_NTRELSTATES]
```

- iteration counters

5.3.1.2 path

```
uint8_t path[CC_NTRELSTATES][CC_MEM_DIM]
```

- Hamming distances

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

- C:/Users/ValmoFil/Music/extra/tlc_chain/src/convolutional.h

5.4 _cc_par_t Struct Reference

Data Fields

- cc_rate_t **cRate**
- cc_klen_t kLen
- uint16_t memFact
- cc_dec_method_t vitDM

5.4.1 Field Documentation

5.4.1.1 kLen

```
cc_klen_t kLen
```

- code rate

5.4.1.2 memFact

```
uint16_t memFact
```

- constrain length

5.4.1.3 vitDM

```
cc_dec_method_t vitDM
```

- memory factor

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

- C:/Users/ValmoFil/Music/extra/tlc_chain/src/convolutional.h

5.5 _cc_soft_dec_info_t Struct Reference

Data Fields

- len_t **iter** [CC_NTRELSTATES]
- float dist [CC_NTRELSTATES]
- uint8_t path [CC_NTRELSTATES][CC_MEM_DIM]

5.5.1 Field Documentation

5.5.1.1 dist

```
float dist[CC_NTRELSTATES]
```

- iteration counters

5.5.1.2 path

```
uint8_t path[CC_NTRELSTATES][CC_MEM_DIM]
```

- Euclidean distances

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

- C:/Users/ValmoFil/Music/extra/tlc_chain/src/convolutional.h

5.6 _cc_trcore_t Struct Reference

Data Fields

- uint8_t **outBits** [CC_NBRANCHES]
- uint8_t nextSt [CC_NBRANCHES]

5.6.1 Field Documentation

5.6.1.1 nextSt

```
uint8_t nextSt[CC_NBRANCHES]
```

- output bits

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

- C:/Users/ValmoFil/Music/extra/tlc_chain/src/convolutional.h

5.7 _cc_trellis_t Struct Reference

Data Fields

- cc_trcore_t **trSt** [CC_NTRELSTATES]

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

- C:/Users/ValmoFil/Music/extra/tlc_chain/src/convolutional.h

5.8 _chan_par_t Struct Reference

Data Fields

- uint32_t **seed**
- channel_t **type**
- uint8_t **bps**
- union {
  float **Peb**
  float EbN0
  };

5.8.1 Field Documentation

5.8.1.1 EbN0

```
float EbN0
```

- BSC error probability

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

- C:/Users/ValmoFil/Music/extra/tlc_chain/src/channel.h

5.9 _complex_stream_t Struct Reference

Data Fields

- complex_t * **pBuf**
- len_t len
- memory_type_t id

5.9.1 Field Documentation

5.9.1.1 id

```
memory_type_t id
```

- buffer length [B]

5.9.1.2 len

```
len_t len
```

- buffer pointer

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

- C:/Users/ValmoFil/Music/extra/tlc_chain/src/memory.h

5.10 _complex_t Struct Reference

Data Fields

- float **re**
- float im

5.10.1 Field Documentation

5.10.1.1 im

```
float im
```

- real part

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

- C:/Users/ValmoFil/Music/extra/tlc_chain/src/system.h

5.11 _debug_par_t Struct Reference

Data Fields

- cc_par_t **ccPar**
- mod_par_t **modPar**
- chan_par_t **chanPar**

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

- C:/Users/ValmoFil/Music/extra/tlc_chain/src/debug.h

5.12 _float_stream_t Struct Reference

Data Fields

- float * **pBuf**
- len_t len
- memory_type_t id

5.12.1 Field Documentation

5.12.1.1 id

```
memory_type_t id
```

- buffer length [B]

5.12.1.2 len

```
len_t len
```

- buffer pointer

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

- C:/Users/ValmoFil/Music/extra/tlc_chain/src/memory.h

5.13 _mod_maptable_t Struct Reference

Data Fields

- byte_t **bits** [MOD_ORDER]
- complex_t **symbs** [MOD_ORDER]

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

- C:/Users/ValmoFil/Music/extra/tlc_chain/src/modulation.h

5.14 _mod_par_t Struct Reference

Data Fields

- modulation_t **type**
- uint8_t order
- uint8_t bps
- float phOfst

5.14.1 Field Documentation

5.14.1.1 bps

```
uint8_t bps
```

- modulation order (aka "M")

5.14.1.2 order

```
uint8_t order
```

- modulation type

5.14.1.3 phOfst

```
float phOfst
```

- number of bits per symbol (aka "L")

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

- C:/Users/ValmoFil/Music/extra/tlc_chain/src/modulation.h

Chapter 6

File Documentation

6.1 C:/Users/ValmoFil/Music/extra/tlc_chain/src/channel.c File Reference

Channel library.

```
#include "channel.h"
```

Functions

- static float GetComplexSgnPower (const complex_stream_t *inStream)

  Function for estimating the average power of a complex stream.
- error_t Channel_ListParameters (chan_par_t *ioParams)

  Function for retrieving and listing channel parameters into dedicated structure.
- error_t Channel_BSC (const byte_stream_t *inStream, byte_stream_t *outStream, const chan_par_t *p↩ Params)

  Function for applying Binary Symmetric Channel (BSC) corruption.
- error_t Channel_AWGN (const complex_stream_t *inStream, complex_stream_t *outStream, const chan_par_t *pParams)

  Function for applying Additive White Gaussian Noise (AWGN) corruption based on Box-Muller method.

6.1.1 Detailed Description

Channel library.

Author

Filippo Valmori

Date

26/08/2024

Library containing channel functions.

6.1.2 Function Documentation

6.1.2.1 Channel_AWGN()

```
error_t Channel_AWGN (
            const complex_stream_t * inStream,
            complex_stream_t * outStream,
            const chan_par_t * pParams)
```

Function for applying Additive White Gaussian Noise (AWGN) corruption based on Box-Muller method.

Parameters

in	*inStream*	input stream
out	*outStream*	output stream
in	*pParams*	poiter to channel parameters structure

Returns

error ID

- noise mean value
- signal average power [lin]
- signal-to-noise-ratio [dB]
- target noise variance (N0)
- random variables uniformly distributed between 0 and 1
- random variables normally distributed as Mu|Sigma2
- link random seed to current time
- link random seed to provided argument

6.1.2.2 Channel_BSC()

```
error_t Channel_BSC (
            const byte_stream_t * inStream,
            byte_stream_t * outStream,
            const chan_par_t * pParams)
```

Function for applying Binary Symmetric Channel (BSC) corruption.

Parameters

in	*inStream*	input stream
out	*outStream*	output stream
in	*pParams*	poiter to channel parameters structure

Returns

error ID

- link random seed to current time
- link random seed to provided argument

6.1.2.3 Channel_ListParameters()

```
error_t Channel_ListParameters (
            chan_par_t * ioParams)
```

Function for retrieving and listing channel parameters into dedicated structure.

Parameters

out	*ioParams*	pointer to i/o parameters structure to be filled

Returns

error ID

6.1.2.4 GetComplexSgnPower()

```
static float GetComplexSgnPower (
            const complex_stream_t * inStream)  [static]
```

Function for estimating the average power of a complex stream.

Parameters

in	*inStream*	input stream

Returns

signal linea average power

6.2 C:/Users/ValmoFil/Music/extra/tlc_chain/src/channel.h File Reference

Channel library header.

```
#include "error.h"
#include "memory.h"
#include "modulation.h"
#include "system.h"
```

Data Structures

- struct _chan_par_t

Macros

- #define **SEED2TIME** UINT32_MAX

  wildcard for linkinf seed to current time
- #define **CHAN_TYPE** CHAN_AWGN

  channel type
- #define **BSC_PEB** 3.5E-2

  BSC channel bit-error probability.
- #define **AWGN_EBN0** 2.9f

  AWGN channel energy-per-bit-to-noise-power-spectral-density ratio [dB] (NB: SNR=Eb/N0∗log2(M)/sps)
- #define **CHAN_SEED** SEED2TIME

  random seed (linked dinamically to current time via SEED2TIME wildcard)

Typedefs

- typedef struct _chan_par_t **chan_par_t**

Enumerations

- enum channel_t { **CHAN_BSC** = 0 , CHAN_AWGN , CHAN_NUM }

Functions

- error_t Channel_ListParameters (chan_par_t ∗ioParams)

  Function for retrieving and listing channel parameters into dedicated structure.
- error_t Channel_BSC (const byte_stream_t ∗inStream, byte_stream_t ∗outStream, const chan_par_t ∗p↩ Params)

  Function for applying Binary Symmetric Channel (BSC) corruption.
- error_t Channel_AWGN (const complex_stream_t ∗inStream, complex_stream_t ∗outStream, const chan_par_t ∗pParams)

  Function for applying Additive White Gaussian Noise (AWGN) corruption based on Box-Muller method.

6.2.1 Detailed Description

Channel library header.

Author

Filippo Valmori

Date

26/08/2024

6.2.2 Enumeration Type Documentation

6.2.2.1 channel_t

```
enum channel_t
```

- import error library
- import memory library
- import modulation library
- import system library

Enumerator

CHAN_AWGN	• BSC channel ID
CHAN_NUM	• AWGN channel ID

6.2.3 Function Documentation

6.2.3.1 Channel_AWGN()

```
error_t Channel_AWGN (
            const complex_stream_t * inStream,
            complex_stream_t * outStream,
            const chan_par_t * pParams)
```

Function for applying Additive White Gaussian Noise (AWGN) corruption based on Box-Muller method.

Parameters

in	*inStream*	input stream
out	*outStream*	output stream
in	*pParams*	poiter to channel parameters structure

Returns

error ID

- noise mean value
- signal average power [lin]
- signal-to-noise-ratio [dB]
- target noise variance (N0)
- random variables uniformly distributed between 0 and 1
- random variables normally distributed as Mu|Sigma2
- link random seed to current time
- link random seed to provided argument

6.2.3.2 Channel_BSC()

```
error_t Channel_BSC (
            const byte_stream_t * inStream,
            byte_stream_t * outStream,
            const chan_par_t * pParams)
```

Function for applying Binary Symmetric Channel (BSC) corruption.

Parameters

in	*inStream*	input stream
out	*outStream*	output stream
in	*pParams*	poiter to channel parameters structure

Returns

error ID

- link random seed to current time
- link random seed to provided argument

6.2.3.3 Channel_ListParameters()

```
error_t Channel_ListParameters (
            chan_par_t * ioParams)
```

Function for retrieving and listing channel parameters into dedicated structure.

Parameters

out	*ioParams*	pointer to i/o parameters structure to be filled

Returns

error ID

6.3 channel.h

Go to the documentation of this file.

```
00001
00010 #ifndef CHANNEL_H
00011 #define CHANNEL_H
00012
00013
00014 /*****************/
00015 /*** INCLUDES ***/
00016 /*****************/
00017
00018 #include "error.h"
00019 #include "memory.h"
00020 #include "modulation.h"
00021 #include "system.h"
00025 /*****************/
00026 /*** TYPEDEFS ***/
00027 /*****************/
00028
00029 typedef enum
00030 {
00031   CHAN_BSC = 0,
00032   CHAN_AWGN,
00033   // keep NUM as final entry
00034   CHAN_NUM
00035 } channel_t;
00036
00037
00038 typedef struct _chan_par_t
00039 {
00040   uint32_t seed;
00041   channel_t type;
00042   uint8_t bps;
00043   union
00044   {
00045     float Peb;
00046     float EbN0;
00047   };
00048 } chan_par_t;
00049
00050
00051
00052 /******************/
00053 /*** CONSTANTS ***/
00054 /******************/
00055
00056 #define SEED2TIME         UINT32_MAX
00057
00058
00059
00060 /******************/
00061 /*** PARAMETERS ***/
00062 /******************/
00063
00064 #define CHAN_TYPE         CHAN_AWGN
00065 #define BSC_PEB           3.5E-2
00066 #define AWGN_EBN0         2.9f
00067 #define CHAN_SEED         SEED2TIME
00068
00069
00070
00071 /*************************/
00072 /*** PUBLIC PROTOTYPES ***/
00073 /*************************/
00074
00075 error_t Channel_ListParameters( chan_par_t * ioParams );
00076 error_t Channel_BSC( const byte_stream_t * inStream, byte_stream_t *outStream, const chan_par_t *
      pParams );
```

```
00077 error_t Channel_AWGN( const complex_stream_t * inStream, complex_stream_t * outStream, const
      chan_par_t * pParams );
00078
00079
00080 #endif
```

6.4 C:/Users/ValmoFil/Music/extra/tlc_chain/src/convolutional.c File Reference

Convolutional coding library.

```
#include "convolutional.h"
```

Functions

- static bool IsRateValid (cc_rate_t rateVal)

  Function for checking the correctness of the code rate parameter.
- static bool IsKlenValid (cc_klen_t kVal)

  Function for checking the correctness of the constraint length parameter.
- static error_t RetrieveConnectorPuncturationVectors (cc_encoder_info_t *ioInfo, const cc_par_t *pParams)

  Function for retrieving connection and puncturation vectors as a function of the selected parameters.
- static uint8_t ComputeEncBit (uint8_t cState, uint8_t cvVal, cc_klen_t kLen)

  Function for computing the next encoded bit.
- static error_t ComputeTrellisDiagram (cc_trellis_t *ioTrellisDiagr, const uint8_t *conVect, const cc_par_t *p↩ Params)

  Function for retrieving depunctured trellis diagram information.
- static error_t HardDepuncturer (byte_t *ioBuffer, len_t inLenBi, len_t outLenBi, const uint8_t *punctVect, const cc_par_t *pParams)

  Function to hard-depuncture a byte stream to base code rate.
- static uint8_t CountByteOnes (byte_t inByte)

  Function for counting the number of 1-bits within the input byte.
- static uint8_t FindMinSurvPathHard (const cc_hard_dec_info_t *inPaths)

  Function to find the hard survivor path with minimum distance.
- static error_t SoftDepuncturer (float *ioBuffer, len_t inLenBi, len_t outLenBi, const uint8_t *punctVect, const cc_par_t *pParams)

  Function to soft-depuncture an LLR stream to base code rate.
- static float EstimateEuclideanDist (const float *inBuf, uint8_t trlByte, uint8_t erasMask)

  Function for calculating Euclidean distance between LLR values and specific trellis state.
- static uint8_t FindMinSurvPathSoft (const cc_soft_dec_info_t *inPaths)

  Function to find the soft survivor path with minimum distance.
- error_t CnvCod_ListParameters (cc_par_t *ioParams)

  Function for retrieving and listing convolution encoding parameters into dedicated structure.
- error_t CnvCod_Encoder (const byte_stream_t *inStream, byte_stream_t *outStream, const cc_par_t *p↩ Params)

  Function for convolutional encoding (including puncturing).
- error_t CnvCod_HardDecoder (const byte_stream_t *inStream, byte_stream_t *outStream, const cc_par_t *pParams)

  Function for convolutional hard-decoding by implementing Viterbi algorithm (including depuncturing).
- error_t CnvCod_SoftDecoder (const float_stream_t *inStream, byte_stream_t *outStream, const cc_par_t *pParams)

  Function for convolutional soft-decoding by implementing Viterbi algorithm (including depuncturing).

Variables

- const cc_rate_t CC_RATE_ARRAY [CC_RATE_NUM]

6.4.1 Detailed Description

Convolutional coding library.

Author

Filippo Valmori

Date

26/08/2024

Library containing convolutional coding functions.

6.4.2 Function Documentation

6.4.2.1 CnvCod_Encoder()

```
error_t CnvCod_Encoder (
            const byte_stream_t * inStream,
            byte_stream_t * outStream,
            const cc_par_t * pParams)
```

Function for convolutional encoding (including puncturing).

Parameters

in	*inStream*	input/source stream
out	*outStream*	output/encoded stream
in	*pParams*	pointer to convolutional coding parameters structure

Returns

errod ID

- unpunctured coded length [B]
- unpunctured coded length [b]
- input buffer length [b]
- expected punctured coded length [b]

- expected punctured coded length [b]
- retrieve convolutional encoder info
- update byte index for reading input buffer
- update bit index for reading input buffer
- update encoder state with latest input bit
- update byte index for output stream writing
- update bit index for output stream writing
- apply puncturing if selected Rc is higher than 1/2
- j-th bit of unpunctured output stream
- check if puncturation has to applied now
- check if computed and theoretical punctured length match
- copy temporary buffer content to output one

6.4.2.2 CnvCod_HardDecoder()

```
error_t CnvCod_HardDecoder (
            const byte_stream_t * inStream,
            byte_stream_t * outStream,
            const cc_par_t * pParams)
```

Function for convolutional hard-decoding by implementing Viterbi algorithm (including depuncturing).

Parameters

in	*inStream*	input stream
out	*outStream*	output stream
in	*pParams*	pointer to convolutional coding parameters structure

Returns

error ID

- retrieve convolutional encoder info
- compute convolutional decoder trellis diagram
- enable only all-zero state at the beginning
- check if depuncturing is needed
- use no-erasure mask for no-puncturing case
- current pair of input bits
- retrieve erasure mask in case depuncturing has been applied

check if j-th state was active in previous iteration

compute Hamming distance assuming hypIdx input bit

compute next state assuming hypIdx input bit

if there's not yet a survivor path for nextSt state at i-th cycle

update state iteration counter

update state distance

update state path among previous states

if a survivor path for nextSt state at i-th cycle already exists, check if new candidate is better

- if input bit stream is over, flush decoder memory and extract final info bits

look for minimum distance survivor path

check if memory has been completely filled

set departure state

set arrival state

set output bit to '0'

set output bit to '1'

- if input bit stream is not over but memory is full, extract oldest info bit
- keep all survivor paths

6.4.2.3 CnvCod_ListParameters()

```
error_t CnvCod_ListParameters (
            cc_par_t * ioParams)
```

Function for retrieving and listing convolution encoding parameters into dedicated structure.

Parameters

in	*ioParams*	pointer to i/o parameters structure to be filled

Returns

error ID

6.4.2.4 CnvCod_SoftDecoder()

```
error_t CnvCod_SoftDecoder (
            const float_stream_t * inStream,
            byte_stream_t * outStream,
            const cc_par_t * pParams)
```

Function for convolutional soft-decoding by implementing Viterbi algorithm (including depuncturing).

Parameters

in	*inStream*	input stream
out	*outStream*	output stream
in	*pParams*	pointer to convolutional coding parameters structure

Returns

error ID

- retrieve convolutional encoder info
- compute convolutional decoder trellis diagram
- enable only the all-zero state at the beginning
- use no-erasure mask for no-puncturing case
- apply depuuncturing if needed
- estimate specific erasure mask if depuncturing has been applied
- check if j-th state was active in the previous iteration

compute Euclidean distance assuming hypIdx-value input bit

compute trellis next state assuming hypIdx-value input bit

iIf there's not yet a survivor path for nextSt state at i-th cycle

update the state iteration counter

update the state distance

update the state path among previous states

if a survivor path for nextSt state at i-th cycle already exists, check if the new candidate is better

- if input bit stream is over, flush the decoder memory and extract the final info bits

look for the minimum distance survivor path

check if memory has been completely filled

departure state

arrival state

set output bit to '0'

set output bit to '1'

- if input bit stream is not over but memory is full, extract the oldest info bit
- set output bit to '0'
- set output bit to '1'
- keep all survivor paths

6.4.2.5 ComputeEncBit()

```
static uint8_t ComputeEncBit (
            uint8_t cState,
            uint8_t cvVal,
            cc_klen_t kLen)  [static]
```

Function for computing the next encoded bit.

Parameters

in	*cState*	current convolutional state
in	*cvVal*	connection vector value
in	*kLen*	encoder constraint length

Returns

next encoded bit

6.4.2.6 ComputeTrellisDiagram()

```
static error_t ComputeTrellisDiagram (
            cc_trellis_t * ioTrellisDiagr,
            const uint8_t * conVect,
            const cc_par_t * pParams)  [static]
```

Function for retrieving depunctured trellis diagram information.

Parameters

out	*ioTrellisDiagr*	pointer to trellis structure to be filled
in	*conVect*	connection vector
in	*pParams*	pointer to parameters structure

Returns

error ID

- state IDs
- state update due to new 0-bit input

estimate encoded bit from i-th connection branch

store 1st encoded bit into trellis

store 2nd encoded bit into trellis

- store next state into trellis (due to 0-bit input from j-th state)
- state update due to new 1-bit input
- store next state into trellis (due to 1-bit input from j-th state)

6.4.2.7 CountByteOnes()

```
static uint8_t CountByteOnes (
            byte_t inByte)  [static]
```

Function for counting the number of 1-bits within the input byte.

Parameters

in	*inByte*	input byte

Returns

number of '1's

6.4.2.8 EstimateEuclideanDist()

```
static float EstimateEuclideanDist (
            const float * inBuf,
            uint8_t trlByte,
            uint8_t erasMask)  [static]
```

Function for calculating Euclidean distance between LLR values and specific trellis state.

Parameters

in	*inBuf*	input LLR buffer
in	*trlByte*	output bits for specific trellis state
in	*erasMask*	depuncturing erasure mask

Returns

estimated Euclidean distance

6.4.2.9 FindMinSurvPathHard()

```
static uint8_t FindMinSurvPathHard (
            const cc_hard_dec_info_t * inPaths)  [static]
```

Function to find the hard survivor path with minimum distance.

Parameters

in	*inPaths*	current Viterbi paths

Returns

index of minimum distance trellis state

6.4.2.10 FindMinSurvPathSoft()

```
static uint8_t FindMinSurvPathSoft (
            const cc_soft_dec_info_t * inPaths)  [static]
```

Function to find the soft survivor path with minimum distance.

Parameters

in	*InPaths*	current Viterbi paths

Returns

index of minimum distance trellis state

6.4.2.11 HardDepuncturer()

```
static error_t HardDepuncturer (
            byte_t * ioBuffer,
            len_t inLenBi,
            len_t outLenBi,
            const uint8_t * punctVect,
            const cc_par_t * pParams)  [static]
```

Function to hard-depuncture a byte stream to base code rate.

Parameters

in,out	*ioBuffer*	i/o buffer
in	*inLenBi*	input/punctured length [b]
in	*outLenBi*	output/unpunctured length [b]
in	*punctVect*	puncturing vector
in	*pParams*	coding parameters

Returns

error ID

- final bit index of input stream length
- each erasure bit restored has '0' value
- set output bit to '0'
- set output bit to '1'

6.4.2.12 IsKlenValid()

```
static bool IsKlenValid (
            cc_klen_t kVal)  [static]
```

Function for checking the correctness of the constraint length parameter.

Parameters

in	*kVal*	constraint length value

Returns

validity outcome

6.4.2.13 IsRateValid()

```
static bool IsRateValid (
            cc_rate_t rateVal)  [static]
```

Function for checking the correctness of the code rate parameter.

Parameters

in	*rateVal*	code rate value

Returns

validity outcome

6.4.2.14 RetrieveConnectorPuncturationVectors()

```
static error_t RetrieveConnectorPuncturationVectors (
            cc_encoder_info_t * ioInfo,
            const cc_par_t * pParams)  [static]
```

Function for retrieving connection and puncturation vectors as a function of the selected parameters.

Parameters

out	*ioInfo*	pointer to i/o structure to be filled
in	*pParams*	pointer to convolutional coding parameters structure

Returns

error ID

- link connector vector
- link puncturation vector

6.4.2.15 SoftDepuncturer()

```
static error_t SoftDepuncturer (
            float * ioBuffer,
            len_t inLenBi,
            len_t outLenBi,
            const uint8_t * punctVect,
            const cc_par_t * pParams)  [static]
```

Function to soft-depuncture an LLR stream to base code rate.

Parameters

in,out	*ioBuffer*	i/o buffer
in	*inLenBi*	input/punctured length [b]
in	*outLenBi*	output/unpunctured length [b]
in	*punctVect*	puncturing vector
in	*pParams*	coding parameters

Returns

error ID

- reading index over LLR array
- reading index within punturing vector
- each erasure LLR restored has 0-value

6.4.3 Variable Documentation

6.4.3.1 CC_RATE_ARRAY

```
const cc_rate_t CC_RATE_ARRAY[CC_RATE_NUM]
```

Initial value:

```
=
{

}
```

6.5 C:/Users/ValmoFil/Music/extra/tlc_chain/src/convolutional.h File Reference

Convolutional coding library header.

```
#include "error.h"
#include "memory.h"
#include "system.h"
```

Data Structures

- struct _cc_par_t
- struct _cc_encoder_info_t
- struct _cc_trcore_t
- struct _cc_trellis_t
- struct _cc_hard_dec_info_t
- struct _cc_soft_dec_info_t

Macros

- #define CC_RATE CC_RATE_12

  Convolutional code rate.
- #define **CC_KLEN** CC_KLEN_7

  Convolutional constrain length.
- #define **CC_MEMFACT** 10u

  Viterbi decoder memory factor.
- #define **CC_VITDM** CC_VITDM_HARD

  Viterbi decoding method.
- #define **CC_NBRANCHES** 2u
- #define **CC_INMASK** ((LSBIT_MASK<<CC_NBRANCHES)-1)
- #define **CC_PUNCTLEN** (CC_NBRANCHES∗CC_RATE)
- #define **CC_NTRELSTATES** (1<<(CC_KLEN-1)) /∗∗ Number of trellis states ∗/
- #define **CC_MEM_DIM** (CC_NTRELSTATES∗CC_MEMFACT) /∗∗ Viterbi decoder memory dimension for survivor paths storage ∗/
- #define CC_CVMATRIX
- #define **CC_PUNC_VECT_23** ((uint8_t[ ]) {1,1,0,1}) /∗∗ Puncturing vector for Rc = 2/3 ∗/
- #define **CC_PUNC_VECT_34** ((uint8_t[ ]) {1,1,0,1,1,0}) /∗∗ Puncturing vector for Rc = 3/4 ∗/
- #define **CC_PUNC_VECT_56** ((uint8_t[ ]) {1,1,0,1,1,0,0,1,1,0}) /∗∗ Puncturing vector for Rc = 5/6 ∗/
- #define **CC_PUNC_VECT_78** ((uint8_t[ ]) {1,1,0,1,0,1,0,1,1,0,0,1,1,0}) /∗∗ Puncturing vector for Rc = 7/8 ∗/
- #define LIST_OF_CC_RATES(ENTRY)
- #define DEF_CC_RATES_VAL(num, den)
- #define DEF_CC_RATES_IDX(num, den)
- #define DEF_CC_RATES_ARRAY(num, den)
- #define CC_VDM_STR(x)

  macro to convert decoding method value into string

Typedefs

- typedef struct _cc_par_t **cc_par_t**
- typedef struct _cc_encoder_info_t **cc_encoder_info_t**
- typedef struct _cc_trcore_t **cc_trcore_t**
- typedef struct _cc_trellis_t **cc_trellis_t**
- typedef struct _cc_hard_dec_info_t **cc_hard_dec_info_t**
- typedef struct _cc_soft_dec_info_t **cc_soft_dec_info_t**

Enumerations

- enum **cc_rate_t**
- enum **cc_rate_idx_t** { **LIST_OF_CC_RATES** =(DEF_CC_RATES_IDX) , **CC_RATE_NUM** }
- enum cc_klen_t {
  CC_KLEN_3 = 3 , CC_KLEN_4 = 4 , CC_KLEN_5 = 5 , CC_KLEN_6 = 6 ,
  CC_KLEN_7 = 7 , CC_KLEN_8 = 8 , CC_KLEN_MIN = CC_KLEN_3 , **CC_KLEN_MAX** = CC_KLEN_8 }
- enum cc_dec_method_t { **CC_VITDM_HARD** = 0 , CC_VITDM_SOFT , CC_VITDM_NUM }

Functions

- error_t CnvCod_ListParameters (cc_par_t *ioParams)

  Function for retrieving and listing convolution encoding parameters into dedicated structure.
- error_t CnvCod_Encoder (const byte_stream_t *inStream, byte_stream_t *outStream, const cc_par_t *p↩ Params)

  Function for convolutional encoding (including puncturing).
- error_t CnvCod_HardDecoder (const byte_stream_t *inStream, byte_stream_t *outStream, const cc_par_t *pParams)

  Function for convolutional hard-decoding by implementing Viterbi algorithm (including depuncturing).
- error_t CnvCod_SoftDecoder (const float_stream_t *inStream, byte_stream_t *outStream, const cc_par_t *pParams)

  Function for convolutional soft-decoding by implementing Viterbi algorithm (including depuncturing).

6.5.1 Detailed Description

Convolutional coding library header.

Author

Filippo Valmori

Date

26/08/2024

6.5.2 Macro Definition Documentation

6.5.2.1 CC_CVMATRIX

```
#define CC_CVMATRIX
```

Value:

```
((uint8_t[][CC_NBRANCHES]) {{7,5},{15,11}, \
{23,25},{47,53},{79,109},{159,229}})
```

6.5.2.2 CC_RATE

```
#define CC_RATE CC_RATE_12
```

Convolutional code rate.

- import error library
- import memory library
- import system library

6.5.2.3 CC_VDM_STR

```
#define CC_VDM_STR(
              x)
```

Value:

```
((x == CC_VITDM_HARD) ? "Hard" : \
(x == CC_VITDM_SOFT) ? "Soft" : \
"N/A")
```

macro to convert decoding method value into string

6.5.2.4 DEF_CC_RATES_ARRAY

```
#define DEF_CC_RATES_ARRAY(
              num,
              den)
```

Value:

```
CC_RATE_##num##den
```

6.5.2.5 DEF_CC_RATES_IDX

```
#define DEF_CC_RATES_IDX(
              num,
              den)
```

Value:

```
CC_RATE_IDX_##num##den
```

6.5.2.6 DEF_CC_RATES_VAL

```
#define DEF_CC_RATES_VAL(
              num,
              den)
```

Value:

```
CC_RATE_##num##den = num
```

6.5.2.7 LIST_OF_CC_RATES

```
#define LIST_OF_CC_RATES(
              ENTRY)
```

Value:

```
  ENTRY( 1, 2 ),                \
  ENTRY( 2, 3 ),                \
  ENTRY( 3, 4 ),                \
  ENTRY( 5, 6 ),                \
  ENTRY( 7, 8 )
```

6.5.3 Enumeration Type Documentation

6.5.3.1 cc_dec_method_t

```
enum cc_dec_method_t
```

Enumerator

CC_VITDM_SOFT	• viterbi hard decoding method ID
CC_VITDM_NUM	• viterbi soft decoding method ID

6.5.3.2 cc_klen_t

```
enum cc_klen_t
```

Enumerator

CC_KLEN_4	• constraint length 3 ID
CC_KLEN_5	• constraint length 4 ID
CC_KLEN_6	• constraint length 5 ID
CC_KLEN_7	• constraint length 6 ID
CC_KLEN_8	• constraint length 7 ID
CC_KLEN_MIN	• constraint length 8 ID

6.5.4 Function Documentation

6.5.4.1 CnvCod_Encoder()

```
error_t CnvCod_Encoder (
            const byte_stream_t * inStream,
            byte_stream_t * outStream,
            const cc_par_t * pParams)
```

Function for convolutional encoding (including puncturing).

Parameters

in	*inStream*	input/source stream
out	*outStream*	output/encoded stream
in	*pParams*	pointer to convolutional coding parameters structure

Returns

errod ID

- unpunctured coded length [B]
- unpunctured coded length [b]
- input buffer length [b]
- expected punctured coded length [b]
- expected punctured coded length [b]
- retrieve convolutional encoder info
- update byte index for reading input buffer
- update bit index for reading input buffer
- update encoder state with latest input bit
- update byte index for output stream writing
- update bit index for output stream writing
- apply puncturing if selected Rc is higher than 1/2
- j-th bit of unpunctured output stream
- check if puncturation has to applied now
- check if computed and theoretical punctured length match
- copy temporary buffer content to output one

6.5.4.2 CnvCod_HardDecoder()

```
error_t CnvCod_HardDecoder (
            const byte_stream_t * inStream,
            byte_stream_t * outStream,
            const cc_par_t * pParams)
```

Function for convolutional hard-decoding by implementing Viterbi algorithm (including depuncturing).

Parameters

in	*inStream*	input stream
out	*outStream*	output stream
in	*pParams*	pointer to convolutional coding parameters structure

Returns

error ID

- retrieve convolutional encoder info
- compute convolutional decoder trellis diagram
- enable only all-zero state at the beginning
- check if depuncturing is needed
- use no-erasure mask for no-puncturing case
- current pair of input bits
- retrieve erasure mask in case depuncturing has been applied

check if j-th state was active in previous iteration

compute Hamming distance assuming hypIdx input bit

compute next state assuming hypIdx input bit

if there's not yet a survivor path for nextSt state at i-th cycle

update state iteration counter

update state distance

update state path among previous states

if a survivor path for nextSt state at i-th cycle already exists, check if new candidate is better

- if input bit stream is over, flush decoder memory and extract final info bits

look for minimum distance survivor path

check if memory has been completely filled

set departure state

set arrival state

set output bit to '0'

set output bit to '1'

- if input bit stream is not over but memory is full, extract oldest info bit
- keep all survivor paths

6.5.4.3 CnvCod_ListParameters()

```
error_t CnvCod_ListParameters (
            cc_par_t * ioParams)
```

Function for retrieving and listing convolution encoding parameters into dedicated structure.

Parameters

in	*ioParams*	pointer to i/o parameters structure to be filled

Returns

error ID

6.5.4.4 CnvCod_SoftDecoder()

```
error_t CnvCod_SoftDecoder (
            const float_stream_t * inStream,
            byte_stream_t * outStream,
            const cc_par_t * pParams)
```

Function for convolutional soft-decoding by implementing Viterbi algorithm (including depuncturing).

Parameters

in	*inStream*	input stream
out	*outStream*	output stream
in	*pParams*	pointer to convolutional coding parameters structure

Returns

error ID

- retrieve convolutional encoder info
- compute convolutional decoder trellis diagram
- enable only the all-zero state at the beginning
- use no-erasure mask for no-puncturing case
- apply depuuncturing if needed
- estimate specific erasure mask if depuncturing has been applied
- check if j-th state was active in the previous iteration

compute Euclidean distance assuming hypIdx-value input bit

compute trellis next state assuming hypIdx-value input bit

iIf there's not yet a survivor path for nextSt state at i-th cycle

update the state iteration counter

update the state distance

update the state path among previous states

if a survivor path for nextSt state at i-th cycle already exists, check if the new candidate is better

- if input bit stream is over, flush the decoder memory and extract the final info bits

look for the minimum distance survivor path

check if memory has been completely filled

departure state

arrival state

set output bit to '0'

set output bit to '1'

- if input bit stream is not over but memory is full, extract the oldest info bit
- set output bit to '0'
- set output bit to '1'
- keep all survivor paths

6.6 convolutional.h

Go to the documentation of this file.

```
00001
00010 #ifndef CONVOLUTIONAL_H
00011 #define CONVOLUTIONAL_H
00012
00013
00014 /*****************/
00015 /*** INCLUDES ***/
00016 /*****************/
00017
00018 #include "error.h"
00019 #include "memory.h"
00020 #include "system.h"
00024 /*******************/
00025 /*** PARAMETERS ***/
00026 /*******************/
00027
00028 #define CC_RATE                 CC_RATE_12
00029 #define CC_KLEN                 CC_KLEN_7
00030 #define CC_MEMFACT              10u
00031 #define CC_VITDM                CC_VITDM_HARD
00032
00033
00034
00035 /******************/
00036 /*** CONSTANTS ***/
00037 /******************/
00038
00039 #define CC_NBRANCHES            2u
00040 #define CC_INMASK               ((LSBIT_MASK«CC_NBRANCHES)-1)
00041 #define CC_PUNCTLEN             (CC_NBRANCHES*CC_RATE)
00042 #define CC_NTRELSTATES          (1«(CC_KLEN-1))
00043 #define CC_MEM_DIM              (CC_NTRELSTATES*CC_MEMFACT)
00045 #define CC_CVMATRIX             ((uint8_t[][CC_NBRANCHES]) {{7,5},{15,11}, \
00046                                   {23,25},{47,53},{79,109},{159,229}})
00048 #define CC_PUNC_VECT_23         ((uint8_t[]) {1,1,0,1})
00049 #define CC_PUNC_VECT_34         ((uint8_t[]) {1,1,0,1,1,0})
00050 #define CC_PUNC_VECT_56         ((uint8_t[]) {1,1,0,1,1,0,0,1,1,0})
00051 #define CC_PUNC_VECT_78         ((uint8_t[]) {1,1,0,1,0,1,0,1,1,0,0,1,1,0})
00055 /*****************/
00056 /*** TYPEDEFS ***/
00057 /*****************/
```

```
00058
00059 // code rates (specify numerator and denominator of each code rate entry)
00060 #define LIST_OF_CC_RATES(ENTRY) \
00061   ENTRY( 1, 2 ),                \
00062   ENTRY( 2, 3 ),                \
00063   ENTRY( 3, 4 ),                \
00064   ENTRY( 5, 6 ),                \
00065   ENTRY( 7, 8 )
00066
00067 #define DEF_CC_RATES_VAL(num,den) CC_RATE_##num##den = num
00068 typedef enum
00069 {
00070   LIST_OF_CC_RATES(DEF_CC_RATES_VAL)
00071 } cc_rate_t;
00072
00073 #define DEF_CC_RATES_IDX(num,den) CC_RATE_IDX_##num##den
00074 typedef enum
00075 {
00076   LIST_OF_CC_RATES(DEF_CC_RATES_IDX),
00077   CC_RATE_NUM
00078 } cc_rate_idx_t;
00079
00080 #define DEF_CC_RATES_ARRAY(num,den) CC_RATE_##num##den
00081
00082 #define CC_VDM_STR(x)    ((x == CC_VITDM_HARD) ? "Hard" : \
00083                           (x == CC_VITDM_SOFT) ? "Soft" : \
00084                           "N/A")
00085
00086 // constraint lengths
00087 typedef enum
00088 {
00089   CC_KLEN_3   = 3,
00090   CC_KLEN_4   = 4,
00091   CC_KLEN_5   = 5,
00092   CC_KLEN_6   = 6,
00093   CC_KLEN_7   = 7,
00094   CC_KLEN_8   = 8,
00095   CC_KLEN_MIN = CC_KLEN_3,
00096   CC_KLEN_MAX = CC_KLEN_8
00097 } cc_klen_t;
00098
00099
00100 // decoding methods
00101 typedef enum
00102 {
00103   CC_VITDM_HARD = 0,
00104   CC_VITDM_SOFT,
00105   // keep NUM as final entry
00106   CC_VITDM_NUM
00107 } cc_dec_method_t;
00108
00109
00110 typedef struct _cc_par_t
00111 {
00112   cc_rate_t cRate;
00113   cc_klen_t kLen;
00114   uint16_t memFact;
00115   cc_dec_method_t vitDM;
00116 } cc_par_t;
00117
00118
00119 typedef struct _cc_encoder_info_t
00120 {
00121   uint8_t connVect[CC_NBRANCHES];
00122   uint8_t puncVect[CC_PUNCTLEN];
00123 } cc_encoder_info_t;
00124
00125
00126 typedef struct _cc_trcore_t
00127 {
00128   uint8_t outBits[CC_NBRANCHES];
00129   uint8_t nextSt[CC_NBRANCHES];
00130 } cc_trcore_t;
00131
00132
00133 typedef struct _cc_trellis_t
00134 {
00135   cc_trcore_t trSt[CC_NTRELSTATES];
00136 } cc_trellis_t;
00137
00138
00139 typedef struct _cc_hard_dec_info_t
00140 {
00141   len_t iter[CC_NTRELSTATES];
00142   uint32_t dist[CC_NTRELSTATES];
00143   uint8_t path[CC_NTRELSTATES][CC_MEM_DIM];
00144 } cc_hard_dec_info_t;
```

```
00145
00146
00147
00148 typedef struct _cc_soft_dec_info_t
00149 {
00150   len_t iter[CC_NTRELSTATES];
00151   float dist[CC_NTRELSTATES];
00152   uint8_t path[CC_NTRELSTATES][CC_MEM_DIM];
00153 } cc_soft_dec_info_t;
00154
00155
00156
00157 /*************************/
00158 /*** PUBLIC PROTOTYPES ***/
00159 /*************************/
00160
00161 error_t CnvCod_ListParameters( cc_par_t * ioParams );
00162 error_t CnvCod_Encoder( const byte_stream_t * inStream, byte_stream_t * outStream, const cc_par_t *
      pParams );
00163 error_t CnvCod_HardDecoder( const byte_stream_t * inStream, byte_stream_t * outStream, const cc_par_t
      * pParams );
00164 error_t CnvCod_SoftDecoder( const float_stream_t * inStream, byte_stream_t * outStream, const cc_par_t
      * pParams );
00165
00166
00167 #endif
```

6.7 C:/Users/ValmoFil/Music/extra/tlc_chain/src/debug.c File Reference

Debug library.

```
#include "debug.h"
```

Functions

- static bool IsSrcLenValid (len_t srcLenBy, const debug_par_t *dbgParams)

  Function for checking source stream length correctness.
- error_t Debug_ListParameters (debug_par_t *ioParams, const cc_par_t *ccParam, const mod_par_t *modParam, const chan_par_t *chanParam)

  Function for retrieving and listing all simulation parameters into dedicated structure.
- error_t Debug_GenerateRandomBytes (byte_stream_t *ioStream, const uint32_t *pSeed)

  Function for filling input buffer with random bytes.
- error_t Debug_PrintByteStream (const byte_stream_t *inStream, print_label_t label, const debug_par_t *pParams)

  Function for printing on terminal a byte buffer content (in hexadecimal format).
- error_t Debug_PrintFloatStream (const float_stream_t *inStream, print_label_t label, const debug_par_t *pParams)

  Function for printing on terminal a float buffer content.
- error_t Debug_PrintComplexStream (const complex_stream_t *inStream, print_label_t label, const debug_par_t *pParams)

  Function for printing on terminal a complex buffer content.
- error_t Debug_PrintParameters (len_t srcLen, const debug_par_t *pParams)

  Function for printing on terminal all simulation parameters.
- error_t Debug_CheckWrongBits (const byte_stream_t *inStreamA, const byte_stream_t *inStreamB, print_label_t label, const debug_par_t *pParams)

  Function for estimating and printing on terminal the number of mismatched bits between two byte streams.
- error_t Debug_WriteByteStreamToCsv (const byte_stream_t *inStream, print_label_t label)

  Function for writing byte stream content into CSV file.
- error_t Debug_WriteComplexStreamToCsv (const complex_stream_t *inStream, print_label_t label)

  Function for writing complex stream content into CSV file.

6.7.1 Detailed Description

Debug library.

Author

Filippo Valmori

Date

26/08/2024

Library containing debug functions.

6.7.2 Function Documentation

6.7.2.1 Debug_CheckWrongBits()

```
error_t Debug_CheckWrongBits (
            const byte_stream_t * inStreamA,
            const byte_stream_t * inStreamB,
            print_label_t label,
            const debug_par_t * pParams)
```

Function for estimating and printing on terminal the number of mismatched bits between two byte streams.

Parameters

in	*inStreamA*	1st input stream
in	*inStreamB*	2nd input stream
in	*label*	label ID
in	*pParams*	pointer to debug parameters structure

Returns

error ID

6.7.2.2 Debug_GenerateRandomBytes()

```
error_t Debug_GenerateRandomBytes (
            byte_stream_t * ioStream,
            const uint32_t * pSeed)
```

Function for filling input buffer with random bytes.

Parameters

out	*ioStream*	i/o stream to be filled
in	*pSeed*	poiter to seed value

Returns

error ID

link random seed to current time

link random seed to provided argument

6.7.2.3 Debug_ListParameters()

```
error_t Debug_ListParameters (
            debug_par_t * ioParams,
            const cc_par_t * ccParam,
            const mod_par_t * modParam,
            const chan_par_t * chanParam)
```

Function for retrieving and listing all simulation parameters into dedicated structure.

Parameters

out	*ioParams*	pointer to i/o parameters structure to be filled
in	*ccParam*	pointer to convolutionan coding parameters structure
in	*modParam*	pointer to modulation coding parameters structure
in	*chanParam*	pointer to channel parameters structure

Returns

error ID

6.7.2.4 Debug_PrintByteStream()

```
error_t Debug_PrintByteStream (
            const byte_stream_t * inStream,
            print_label_t label,
            const debug_par_t * pParams)
```

Function for printing on terminal a byte buffer content (in hexadecimal format).

Parameters

in	*inStream*	input stream
in	*label*	label ID
in	*pParams*	pointer to debug parameters structure

Returns

error ID

6.7.2.5 Debug_PrintComplexStream()

```
error_t Debug_PrintComplexStream (
            const complex_stream_t * inStream,
            print_label_t label,
            const debug_par_t * pParams)
```

Function for printing on terminal a complex buffer content.

Parameters

in	*inStream*	input stream
in	*label*	label ID
in	*pParams*	pointer to debug parameters structure

Returns

error ID

6.7.2.6 Debug_PrintFloatStream()

```
error_t Debug_PrintFloatStream (
            const float_stream_t * inStream,
            print_label_t label,
            const debug_par_t * pParams)
```

Function for printing on terminal a float buffer content.

Parameters

in	*inStream*	input stream
in	*label*	label ID
in	*pParams*	pointer to debug parameters structure

Returns

error ID

6.7.2.7 Debug_PrintParameters()

```
error_t Debug_PrintParameters (
            len_t srcLen,
            const debug_par_t * pParams)
```

Function for printing on terminal all simulation parameters.

Parameters

in	*srcLen*	source buffer length [B]
in	*pParams*	pointer to debug parameters structure

Returns

error ID

6.7.2.8 Debug_WriteByteStreamToCsv()

```
error_t Debug_WriteByteStreamToCsv (
            const byte_stream_t * inStream,
            print_label_t label)
```

Function for writing byte stream content into CSV file.

Parameters

in	*inStream*	input stream
in	*label*	label ID

Returns

error ID

write stream length as 1st element

6.7.2.9 Debug_WriteComplexStreamToCsv()

```
error_t Debug_WriteComplexStreamToCsv (
            const complex_stream_t * inStream,
            print_label_t label)
```

Function for writing complex stream content into CSV file.

Parameters

in	*inStream*	input stream
in	*label*	label ID

Returns

error ID

write stream length as 1st element

6.7.2.10 IsSrcLenValid()

```
static bool IsSrcLenValid (
            len_t srcLenBy,
            const debug_par_t * dbgParams)  [static]
```

Function for checking source stream length correctness.

Parameters

lenBy	: source buffer length [B]

Returns

validity outcome

source bit length shall be positive and divisible by code rate denominator

convolutional punctured bit length shall be a multiple of NUM_BITS_PER_BYTE

convolutional punctured bit length shall be a multiple of MOD_BPS

6.8 C:/Users/ValmoFil/Music/extra/tlc_chain/src/debug.h File Reference

Debug library header.

```
#include "channel.h"
#include "convolutional.h"
#include "error.h"
#include "memory.h"
#include "modulation.h"
#include "system.h"
```

Data Structures

- struct _debug_par_t

Macros

- #define PID_NCOLS_BYTE 35u

  Number of byte columns per row to print before wrapping.
- #define **PID_NCOLS_FLOAT** 15u

  Number of float columns per row to print before wrapping.
- #define **PID_NCOLS_SYMB** 8u

  Number of symbol columns per row to print before wrapping.

Typedefs

- typedef struct _debug_par_t **debug_par_t**

Enumerations

- enum print_label_t {
  PID_TX_SRC = 0 , PID_RX_SRC , PID_TX_CNVCOD , PID_RX_CNVCOD ,
  PID_TX_MAP , PID_RX_MAP , PID_RX_LLR , PID_NUM }

Functions

- error_t Debug_PrintParameters (len_t srcLen, const debug_par_t *pParams)

  Function for printing on terminal all simulation parameters.
- error_t Debug_ListParameters (debug_par_t *ioParams, const cc_par_t *ccParam, const mod_par_t *mod↩
  Param, const chan_par_t *chanParam)

  Function for retrieving and listing all simulation parameters into dedicated structure.
- error_t Debug_GenerateRandomBytes (byte_stream_t *ioStream, const uint32_t *pSeed)

  Function for filling input buffer with random bytes.
- error_t Debug_PrintByteStream (const byte_stream_t *inStream, print_label_t label, const debug_par_t *p↩
  Params)

  Function for printing on terminal a byte buffer content (in hexadecimal format).
- error_t Debug_PrintFloatStream (const float_stream_t *inStream, print_label_t label, const debug_par_t *p↩
  Params)

Function for printing on terminal a float buffer content.

- error_t Debug_PrintComplexStream (const complex_stream_t *inStream, print_label_t label, const debug_par_t *pParams)

  Function for printing on terminal a complex buffer content.

- error_t Debug_CheckWrongBits (const byte_stream_t *inStreamA, const byte_stream_t *inStreamB, print_label_t label, const debug_par_t *pParams)

  Function for estimating and printing on terminal the number of mismatched bits between two byte streams.

- error_t Debug_WriteByteStreamToCsv (const byte_stream_t *inStream, print_label_t label)

  Function for writing byte stream content into CSV file.

- error_t Debug_WriteComplexStreamToCsv (const complex_stream_t *inStream, print_label_t label)

  Function for writing complex stream content into CSV file.

6.8.1 Detailed Description

Debug library header.

Author

Filippo Valmori

Date

26/08/2024

6.8.2 Macro Definition Documentation

6.8.2.1 PID_NCOLS_BYTE

```
#define PID_NCOLS_BYTE 35u
```

Number of byte columns per row to print before wrapping.

- import channel library
- import convolutional library
- import error library
- import memory library
- import modulation library
- import system library

6.8.3 Enumeration Type Documentation

6.8.3.1 print_label_t

```
enum print_label_t
```

Enumerator

PID_RX_SRC	Tx source bytes print-ID
PID_TX_CNVCOD	Rx source bytes print-ID
PID_RX_CNVCOD	Tx convolution coded bytes print-ID
PID_TX_MAP	Rx convolution coded bytes print-ID
PID_RX_MAP	Tx mapped symbols print-ID
PID_RX_LLR	Rx mapped symbols print-ID
PID_NUM	Rx demapped LLRs print-ID

6.8.4 Function Documentation

6.8.4.1 Debug_CheckWrongBits()

```
error_t Debug_CheckWrongBits (
            const byte_stream_t * inStreamA,
            const byte_stream_t * inStreamB,
            print_label_t label,
            const debug_par_t * pParams)
```

Function for estimating and printing on terminal the number of mismatched bits between two byte streams.

Parameters

in	*inStreamA*	1st input stream
in	*inStreamB*	2nd input stream
in	*label*	label ID
in	*pParams*	pointer to debug parameters structure

Returns

error ID

6.8.4.2 Debug_GenerateRandomBytes()

```
error_t Debug_GenerateRandomBytes (
            byte_stream_t * ioStream,
            const uint32_t * pSeed)
```

Function for filling input buffer with random bytes.

Parameters

out	*ioStream*	i/o stream to be filled
in	*pSeed*	poiter to seed value

Returns

error ID

link random seed to current time

link random seed to provided argument

6.8.4.3 Debug_ListParameters()

```
error_t Debug_ListParameters (
            debug_par_t * ioParams,
            const cc_par_t * ccParam,
            const mod_par_t * modParam,
            const chan_par_t * chanParam)
```

Function for retrieving and listing all simulation parameters into dedicated structure.

Parameters

out	*ioParams*	pointer to i/o parameters structure to be filled
in	*ccParam*	pointer to convolutionan coding parameters structure
in	*modParam*	pointer to modulation coding parameters structure
in	*chanParam*	pointer to channel parameters structure

Returns

error ID

6.8.4.4 Debug_PrintByteStream()

```
error_t Debug_PrintByteStream (
            const byte_stream_t * inStream,
            print_label_t label,
            const debug_par_t * pParams)
```

Function for printing on terminal a byte buffer content (in hexadecimal format).

Parameters

in	*inStream*	input stream
in	*label*	label ID
in	*pParams*	pointer to debug parameters structure

Returns

error ID

6.8.4.5 Debug_PrintComplexStream()

```
error_t Debug_PrintComplexStream (
            const complex_stream_t * inStream,
            print_label_t label,
            const debug_par_t * pParams)
```

Function for printing on terminal a complex buffer content.

Parameters

in	*inStream*	input stream
in	*label*	label ID
in	*pParams*	pointer to debug parameters structure

Returns

error ID

6.8.4.6 Debug_PrintFloatStream()

```
error_t Debug_PrintFloatStream (
            const float_stream_t * inStream,
            print_label_t label,
            const debug_par_t * pParams)
```

Function for printing on terminal a float buffer content.

Parameters

in	*inStream*	input stream
in	*label*	label ID
in	*pParams*	pointer to debug parameters structure

Returns

error ID

6.8.4.7 Debug_PrintParameters()

```
error_t Debug_PrintParameters (
            len_t srcLen,
            const debug_par_t * pParams)
```

Function for printing on terminal all simulation parameters.

Parameters

in	*srcLen*	source buffer length [B]
in	*pParams*	pointer to debug parameters structure

Returns

error ID

6.8.4.8 Debug_WriteByteStreamToCsv()

```
error_t Debug_WriteByteStreamToCsv (
            const byte_stream_t * inStream,
            print_label_t label)
```

Function for writing byte stream content into CSV file.

Parameters

in	*inStream*	input stream
in	*label*	label ID

Returns

error ID

write stream length as 1st element

6.8.4.9 Debug_WriteComplexStreamToCsv()

```
error_t Debug_WriteComplexStreamToCsv (
            const complex_stream_t * inStream,
            print_label_t label)
```

Function for writing complex stream content into CSV file.

Parameters

in	*inStream*	input stream
in	*label*	label ID

Returns

error ID

write stream length as 1st element

6.9 debug.h

Go to the documentation of this file.

```
00001
00010 #ifndef DEBUG_H
00011 #define DEBUG_H
00012
00013
00014 /****************/
00015 /*** INCLUDES ***/
00016 /****************/
00017
00018 #include "channel.h"
00019 #include "convolutional.h"
00020 #include "error.h"
00021 #include "memory.h"
00022 #include "modulation.h"
00023 #include "system.h"
00027 /******************/
00028 /*** PARAMETERS ***/
00029 /******************/
00030
00031 #define PID_NCOLS_BYTE      35u
00032 #define PID_NCOLS_FLOAT     15u
00033 #define PID_NCOLS_SYMB      8u
00034
00035
00036
00037 /****************/
```

```
00038 /*** TYPEDEFS ***/
00039 /*****************/
00040
00041 typedef enum
00042 {
00043   PID_TX_SRC = 0,
00044   PID_RX_SRC,
00045   PID_TX_CNVCOD,
00046   PID_RX_CNVCOD,
00047   PID_TX_MAP,
00048   PID_RX_MAP,
00049   PID_RX_LLR,
00050   // keep NUM as final entry
00051   PID_NUM
00052 } print_label_t;
00053
00054
00055 typedef struct _debug_par_t
00056 {
00057   cc_par_t ccPar;
00058   mod_par_t modPar;
00059   chan_par_t chanPar;
00060 } debug_par_t;
00061
00062
00063
00064 /*******************/
00065 /*** PROTOTYPES ***/
00066 /*******************/
00067
00068 error_t Debug_PrintParameters( len_t srcLen, const debug_par_t * pParams );
00069 error_t Debug_ListParameters( debug_par_t * ioParams, const cc_par_t * ccParam, const mod_par_t *
      modParam, const chan_par_t * chanParam );
00070 error_t Debug_GenerateRandomBytes( byte_stream_t * ioStream, const uint32_t * pSeed );
00071 error_t Debug_PrintByteStream( const byte_stream_t * inStream, print_label_t label, const debug_par_t
      * pParams );
00072 error_t Debug_PrintFloatStream( const float_stream_t * inStream, print_label_t label, const
      debug_par_t * pParams );
00073 error_t Debug_PrintComplexStream( const complex_stream_t * inStream, print_label_t label, const
      debug_par_t * pParams );
00074 error_t Debug_CheckWrongBits( const byte_stream_t * inStreamA, const byte_stream_t * inStreamB,
      print_label_t label, const debug_par_t * pParams );
00075 error_t Debug_WriteByteStreamToCsv( const byte_stream_t * inStream, print_label_t label );
00076 error_t Debug_WriteComplexStreamToCsv( const complex_stream_t * inStream, print_label_t label );
00077
00078
00079 #endif
```

6.10 C:/Users/ValmoFil/Music/extra/tlc_chain/src/error.c File Reference

Error library.

`#include "error.h"`

Functions

- error_t Error_HandleErr (error_t inErr)
  Function for handling detected error.

6.10.1 Detailed Description

Error library.

Author

Filippo Valmori

Date

26/08/2024

Library containing error handling functions.

6.10.2 Function Documentation

6.10.2.1 Error_HandleErr()

```
error_t Error_HandleErr (
            error_t inErr)
```

Function for handling detected error.

Parameters

in	*inErr*	detected error

Returns

error ID

6.11 C:/Users/ValmoFil/Music/extra/tlc_chain/src/error.h File Reference

Error handling library header.

```
#include "system.h"
```

Macros

- #define **ALARM_TYPE** ((alarm_t) ALARM_STOP)

Enumerations

- enum error_t {
  ERR_NONE = 0 , ERR_INV_NULL_POINTER , ERR_INV_PRINTID , ERR_INV_CNVCOD_RATE , ERR_INV_CNVCOD_KLEN , ERR_INV_CNVCOD_DECMET , ERR_INV_BUFFER_SIZE , ERR_INV_DYNAMIC_ALLOC , ERR_INV_STREAM_TYPE , ERR_INV_MODULATION_TYPE , ERR_INV_MODULATION_BPS , ERR_INV_CHANNEL_TYPE , ERR_NUM }
- enum alarm_t { **ALARM_NONE** = 0 , ALARM_PRINT , ALARM_STOP , ALARM_NUM }

Functions

- error_t Error_HandleErr (error_t inErr)
  Function for handling detected error.

6.11.1 Detailed Description

Error handling library header.

Author

Filippo Valmori

Date

26/08/2024

6.11.2 Enumeration Type Documentation

6.11.2.1 alarm_t

`enum alarm_t`

Enumerator

ALARM_PRINT	• no alarm ID
ALARM_STOP	• print on terminal alarm ID
ALARM_NUM	• stop execution alarm ID

6.11.2.2 error_t

`enum error_t`

- import system library

Enumerator

ERR_INV_NULL_POINTER	• successful error ID
ERR_INV_PRINTID	• invalid null pointer error ID
ERR_INV_CNVCOD_RATE	• invalid print label provided error ID

Enumerator

ERR_INV_CNVCOD_KLEN	• invalid convolutional coding rate error ID
ERR_INV_CNVCOD_DECMET	• invalid convolutional coding constaint length error ID
ERR_INV_BUFFER_SIZE	• invalid convolutional coding decoding method error ID
ERR_INV_DYNAMIC_ALLOC	• invalid buffer size error ID
ERR_INV_STREAM_TYPE	• invalid dynamic memory allocation error ID
ERR_INV_MODULATION_TYPE	• invalid stream type error ID
ERR_INV_MODULATION_BPS	• invalid modulation type error ID
ERR_INV_CHANNEL_TYPE	• invalid modulation BPS error ID
ERR_NUM	• invalid channel type error ID

6.11.3 Function Documentation

6.11.3.1 Error_HandleErr()

```
error_t Error_HandleErr (
            error_t inErr)
```

Function for handling detected error.

Parameters

in	*inErr*	detected error

Returns

error ID

6.12 error.h

Go to the documentation of this file.

```
00001
00010 #ifndef ERROR_H
00011 #define ERROR_H
00012
00013
00014 /*****************/
00015 /*** INCLUDES ***/
00016 /*****************/
00017
00018 #include "system.h"
00022 /******************/
00023 /*** TYPEDEFS ***/
00024 /******************/
00025
00026 typedef enum
00027 {
00028   ERR_NONE = 0,
00029   ERR_INV_NULL_POINTER,
00030   ERR_INV_PRINTID,
00031   ERR_INV_CNVCOD_RATE,
00032   ERR_INV_CNVCOD_KLEN,
00033   ERR_INV_CNVCOD_DECMET,
00034   ERR_INV_BUFFER_SIZE,
00035   ERR_INV_DYNAMIC_ALLOC,
00036   ERR_INV_STREAM_TYPE,
00037   ERR_INV_MODULATION_TYPE,
00038   ERR_INV_MODULATION_BPS,
00039   ERR_INV_CHANNEL_TYPE,
00040   // keep NUM as final entry
00041   ERR_NUM
00042 } error_t;
00043
00044
00045 typedef enum
00046 {
00047   ALARM_NONE = 0,
00048   ALARM_PRINT,
00049   ALARM_STOP,
00050   // keep NUM as final entry
00051   ALARM_NUM
00052 } alarm_t;
00053
00054
00055
00056 /*******************/
00057 /*** PARAMETERS ***/
00058 /*******************/
00059
00060 #define ALARM_TYPE          ((alarm_t) ALARM_STOP)
00061
00062
00063
00064 /*******************/
00065 /*** PROTOTYPES ***/
00066 /*******************/
00067
00068 error_t Error_HandleErr( error_t inErr );
00069
00070
00071 #endif
```

6.13 C:/Users/ValmoFil/Music/extra/tlc_chain/src/main.c File Reference

Main file.

```
#include "channel.h"
#include "convolutional.h"
#include "debug.h"
#include "error.h"
#include "memory.h"
#include "modulation.h"
```

Macros

- #define **LEN_SRC_BY** 150u

  source info stream length [B] (NB: Max = 1000)
- #define **LEN_CC_UNP_BY** (CC_NBRANCHES∗LEN_SRC_BY)

  unpunctured convolutional coded stream length [B]
- #define LEN_CC_PUN_BY

  punctured convolutional coded stream length [B]
- #define **LEN_CC_PUN_BI** BY2BI_LEN(LEN_CC_PUN_BY)

  punctured convolutional coded stream length [b]
- #define **LEN_MOD_SY** (LEN_CC_PUN_BI/MOD_BPS)

  modulated symbol stream length [Sy]
- #define **LEN_LLR_FL** LEN_CC_PUN_BI
- #define DEF_STREAM_DECLARE(name, type, length)
- #define DEF_STREAM_ALLOCATE(name, type, length)
- #define DEF_STREAM_FREE(name, type, length)
- #define LIST_OF_STREAMS(ENTRY)

Functions

- int main (void)

Variables

- static cc_par_t **ccParams**
- static mod_par_t **modParams**
- static chan_par_t **chanParams**
- static debug_par_t **dgbParams**

6.13.1 Detailed Description

Main file.

Author

Filippo Valmori

Date

26/08/2024

File for running DVB-S simulation containing:

1. random info bytes generation;
2. scrambling;
3. reed-solomon coding;
4. interleaving;
5. convolutional coding;
6. phase modulation (mapper + srrc filtering);
7. channel corruption.

6.13.2 Macro Definition Documentation

6.13.2.1 DEF_STREAM_ALLOCATE

```
#define DEF_STREAM_ALLOCATE(
                name,
                type,
                length)
```

Value:
```
Memory_AllocateStream(&name##Stream,length,name##Stream.id);
```

6.13.2.2 DEF_STREAM_DECLARE

```
#define DEF_STREAM_DECLARE(
                name,
                type,
                length)
```

Value:
```
type##_stream_t name##Stream = {.pBuf = NULL, .len = 0, .id = memory_type_##type};
```

6.13.2.3 DEF_STREAM_FREE

```
#define DEF_STREAM_FREE(
                name,
                type,
                length)
```

Value:
```
Memory_FreeStream(&name##Stream,memory_type_##type);
```

6.13.2.4 LEN_CC_PUN_BY

```
#define LEN_CC_PUN_BY
```

Value:
```
                            (LEN_CC_UNP_BY/CC_NBRANCHES* \
                            (CC_RATE+1)/CC_RATE)
```

punctured convolutional coded stream length [B]

6.13.2.5 LIST_OF_STREAMS

```
#define LIST_OF_STREAMS(
                ENTRY)
```

Value:
```
  ENTRY( txSrc, byte,    LEN_SRC_BY    ) \
  ENTRY( rxSrc, byte,    LEN_SRC_BY    ) \
  ENTRY( txCc,  byte,    LEN_CC_PUN_BY ) \
  ENTRY( rxCc,  byte,    LEN_CC_PUN_BY ) \
  ENTRY( txMod, complex, LEN_MOD_SY    ) \
  ENTRY( rxMod, complex, LEN_MOD_SY    ) \
  ENTRY( rxLLR, float,   LEN_LLR_FL    )
```

6.13.3 Function Documentation

6.13.3.1 main()

```
int main (
             void )
```

- declare all streams
- allocate memory for all streams
- list convolutional coding parameters
- list channel parameters
- list modulation parameters
- fill tx source buffer with random bytes
- convolutional encoder
- apply bsc channel corruption
- convolutional hard-decoder
- modulation mapper
- apply awgn channel corruption
- modulation hard-demapper
- convolutional hard-decoder
- modulation soft-demapper
- convolutional soft-decoder
- check number of corrupted bits at convolutional coding level
- check number of corrupted bits at source level
- free memory for all streams

6.14 C:/Users/ValmoFil/Music/extra/tlc_chain/src/memory.c File Reference

Memory ibrary.

```
#include "memory.h"
```

Functions

- static error_t AllocateByteStream (byte_stream_t *ioStream, len_t len)
  Function for dynamically allocating memory for a byte stream.
- static error_t AllocateFloatStream (float_stream_t *ioStream, len_t len)
  Function for dynamically allocating memory for a float stream.
- static error_t AllocateComplexStream (complex_stream_t *ioStream, len_t len)
  Function for dynamically allocating memory for a complex stream.
- static error_t FreeByteStream (byte_stream_t *ioStream)
  Function for dynamically deallocating memory for a byte stream.
- static error_t FreeFloatStream (float_stream_t *ioStream)
  Function for dynamically deallocating memory for a float stream.
- static error_t FreeComplexStream (complex_stream_t *ioStream)
  Function for dynamically deallocating memory for a complex stream.
- error_t Memory_AllocateStream (void *ioStream, len_t len, memory_type_t type)
  Function for dynamically allocating memory for any type of stream.
- error_t Memory_FreeStream (void *ioStream, memory_type_t type)
  Function for dynamically deallocating memory for any type of stream.

6.14.1 Detailed Description

Memory ibrary.

Author

Filippo Valmori

Date

26/08/2024

Library containing dynamic memory allocation functions.

6.14.2 Function Documentation

6.14.2.1 AllocateByteStream()

```
static error_t AllocateByteStream (
            byte_stream_t * ioStream,
            len_t len)  [static]
```

Function for dynamically allocating memory for a byte stream.

Parameters

in,out	*ioStream*	i/o stream whose buffer has to be allocated
in	*len*	buffer length

Returns

error ID

6.14.2.2 AllocateComplexStream()

```
static error_t AllocateComplexStream (
            complex_stream_t * ioStream,
            len_t len)  [static]
```

Function for dynamically allocating memory for a complex stream.

Parameters

in,out	*ioStream*	i/o stream whose buffer has to be allocated
in	*len*	buffer length

Returns

error ID

6.14.2.3 AllocateFloatStream()

```
static error_t AllocateFloatStream (
            float_stream_t * ioStream,
            len_t len)  [static]
```

Function for dynamically allocating memory for a float stream.

Parameters

in,out	*ioStream*	i/o stream whose buffer has to be allocated
in	*len*	buffer length

Returns

error ID

6.14.2.4 FreeByteStream()

```
static error_t FreeByteStream (
            byte_stream_t * ioStream)  [static]
```

Function for dynamically deallocating memory for a byte stream.

Parameters

ioStream[in,out]	i/o stream whose buffer has to be deallocated

Returns

error ID

6.14.2.5 FreeComplexStream()

```
static error_t FreeComplexStream (
            complex_stream_t * ioStream)  [static]
```

Function for dynamically deallocating memory for a complex stream.

Parameters

ioStream[in,out]	i/o stream whose buffer has to be deallocated

Returns

error ID

6.14.2.6 FreeFloatStream()

```
static error_t FreeFloatStream (
            float_stream_t * ioStream)  [static]
```

Function for dynamically deallocating memory for a float stream.

Parameters

ioStream[in,out]	i/o stream whose buffer has to be deallocated

Returns

error ID

6.14.2.7 Memory_AllocateStream()

```
error_t Memory_AllocateStream (
            void * ioStream,
            len_t len,
            memory_type_t type)
```

Function for dynamically allocating memory for any type of stream.

Parameters

in,out	*ioStream*	i/o stream whose buffer has to be allocated
in	*len*	buffer length
in	*type*	stream type ID

Returns

error ID

6.14.2.8 Memory_FreeStream()

```
error_t Memory_FreeStream (
            void * ioStream,
            memory_type_t type)
```

Function for dynamically deallocating memory for any type of stream.

Parameters

in,out	*ioStream*	i/o stream whose buffer has to be deallocated
in	*type*	stream type ID

Returns

error ID

6.15 memory.h

```
00001
00010 #ifndef MEMORY_H
00011 #define MEMORY_H
00012
00013
00014 /****************/
00015 /*** INCLUDES ***/
00016 /****************/
00017
00018
00019 #include "error.h"
00020 #include "system.h"
00024 /****************/
00025 /*** TYPEDEFS ***/
00026 /****************/
00027
00028 typedef enum
00029 {
00030   memory_type_byte = 0,
00031   memory_type_float,
00032   memory_type_complex
00033 } memory_type_t;
00034
00035
00036 typedef struct _byte_stream_t
00037 {
00038   byte_t * pBuf;
00039   len_t len;
00040   memory_type_t id;
00041 } byte_stream_t;
00042
00043
00044 typedef struct _float_stream_t
00045 {
```

```
00046   float * pBuf;
00047   len_t len;
00048   memory_type_t id;
00049 } float_stream_t;
00050
00051
00052 typedef struct _complex_stream_t
00053 {
00054   complex_t * pBuf;
00055   len_t len;
00056   memory_type_t id;
00057 } complex_stream_t;
00058
00059
00060
00061 /*************************/
00062 /*** PUBLIC PROTOTYPES ***/
00063 /*************************/
00064
00065 error_t Memory_AllocateStream( void * ioStream, len_t len, memory_type_t type );
00066 error_t Memory_FreeStream( void * ioStream, memory_type_t type );
00067
00068
00069 #endif
```

6.16 C:/Users/ValmoFil/Music/extra/tlc_chain/src/modulation.c File Reference

Modulation library.

```
#include "modulation.h"
```

Functions

- static bool IsQamBpsValid (uint8_t bps)

  Function for checking if QAM BPS parameter is valid.
- static error_t GetMappingTable (mod_maptable_t *ioTable, const mod_par_t *pParams)

  Function for retrieving specific mapping table according to modulation.
- static error_t GetPskTable (mod_maptable_t *ioTable, const mod_par_t *pParams)

  Function for computing Gray-coded PSK mapping table.
- static error_t GetQamTable (mod_maptable_t *ioTable, const mod_par_t *pParams)

  Function for computing Gray-coded QAM mapping table.
- static error_t GetGraySequence (byte_t *ioBuffer, const mod_par_t *pParams)

  Function for retrieving Gray coded sequence.
- error_t Modulation_ListParameters (mod_par_t *ioParams)

  Function for retrieving and listing modulation parameters into dedicated structure.
- error_t Modulation_Mapper (const byte_stream_t *inStream, complex_stream_t *outStream, const mod_par_t *pParams)

  Function for byte to complex symbol stream mapping.
- error_t Modulation_HardDemapper (const complex_stream_t *inStream, byte_stream_t *outStream, const mod_par_t *pParams)

  Function for hard-demapping an input symbol stream into corresponding byte stream.
- error_t Modulation_SoftDemapper (const complex_stream_t *inStream, float_stream_t *outStream, const mod_par_t *pParams)

  Function for soft-demapping an input symbol stream into corresponding LLR stream.

6.16.1 Detailed Description

Modulation library.

Author

Filippo Valmori

Date

26/08/2024

Library containing modulation functions.

6.16.2 Function Documentation

6.16.2.1 GetGraySequence()

```
static error_t GetGraySequence (
            byte_t * ioBuffer,
            const mod_par_t * pParams)  [static]
```

Function for retrieving Gray coded sequence.

Parameters

out	*ioBuffer*	i/o buffer to be filled
in	*pParams*	pointer to modulation parameters structure

Returns

error ID

- clear buffer content
- number of bits per block at i-th iteration
- bit shift value at i-th iteration
- counter within each single block
- starting value of the writing index

6.16.2.2 GetMappingTable()

```
static error_t GetMappingTable (
            mod_maptable_t * ioTable,
            const mod_par_t * pParams)  [static]
```

Function for retrieving specific mapping table according to modulation.

Parameters

out	*ioTable*	i/o table
in	*pParams*	pointer to modulation parameters structure

Returns

error ID

6.16.2.3 GetPskTable()

```
static error_t GetPskTable (
            mod_maptable_t * ioTable,
            const mod_par_t * pParams)  [static]
```

Function for computing Gray-coded PSK mapping table.

Parameters

out	*ioTable*	i/o table
in	*pParams*	pointer to modulation parameters structure

Returns

error ID

6.16.2.4 GetQamTable()

```
static error_t GetQamTable (
            mod_maptable_t * ioTable,
            const mod_par_t * pParams)  [static]
```

Function for computing Gray-coded QAM mapping table.

Parameters

out	*ioTable*	i/o table
in	*pParams*	pointer to modulation parameters structure

Returns

error ID

6.16.2.5 IsQamBpsValid()

```
static bool IsQamBpsValid (
            uint8_t bps)  [static]
```

Function for checking if QAM BPS parameter is valid.

Parameters

in	*bps*	modulation bits-per-symbol value

Returns

validity outcome

6.16.2.6 Modulation_HardDemapper()

```
error_t Modulation_HardDemapper (
            const complex_stream_t * inStream,
            byte_stream_t * outStream,
            const mod_par_t * pParams)
```

Function for hard-demapping an input symbol stream into corresponding byte stream.

Parameters

in	*inStream*	input stream
out	*outStream*	output stream
in	*pParams*	pointer to modulation parameters structure

Returns

error ID

- bit counter for output stream writing
- retrieve mapping table
- clear output buffer

6.16.2.7 Modulation_ListParameters()

```
error_t Modulation_ListParameters (
            mod_par_t * ioParams)
```

Function for retrieving and listing modulation parameters into dedicated structure.

Parameters

out	*ioParams*	pointer to i/o parameters structure to be filled

Returns

error ID

6.16.2.8 Modulation_Mapper()

```
error_t Modulation_Mapper (
            const byte_stream_t * inStream,
            complex_stream_t * outStream,
            const mod_par_t * pParams)
```

Function for byte to complex symbol stream mapping.

Parameters

in	*inStream*	input stream
out	*outStream*	output stream
in	*pParams*	pointer to modulation parameters structure

Returns

error ID

6.16.2.9 Modulation_SoftDemapper()

```
error_t Modulation_SoftDemapper (
            const complex_stream_t * inStream,
            float_stream_t * outStream,
            const mod_par_t * pParams)
```

Function for soft-demapping an input symbol stream into corresponding LLR stream.

Parameters

in	*inStream*	input stream
out	*outStream*	output stream
in	*pParams*	pointer to modulation parameters structure

Returns

error ID

6.17 C:/Users/ValmoFil/Music/extra/tlc_chain/src/modulation.h File Reference

Modulation library header.

```
#include "error.h"
#include "memory.h"
#include "system.h"
```

Data Structures

- struct _mod_par_t
- struct _mod_maptable_t

Macros

- #define **MOD_TYPE** ((modulation_t) MOD_PSK)

  modulation type
- #define **MOD_BPS** 2u

  modulation number of bits per symbol [b/Sy]
- #define **MOD_SD_N0** ((float) 4.0) /∗∗ Assumed linear noise variance for soft demapper (e.g. 4 equals 6 dB) ∗/
- #define **MOD_BPS_MIN** 1u

  Minimum BPS value allowed.
- #define **MOD_BPS_MAX** 6u

  Maximum BPS value allowed.
- #define **MOD_ORDER** (0x01<<MOD_BPS)

  modulation order (NB: Min. = 2 | Max = 64)
- #define **MOD_BINARY** 2u

  Binary modulation order.
- #define MOD_TYPE_STR(x)

  macro to convert modulation type value into string

Typedefs

- typedef struct _mod_par_t **mod_par_t**
- typedef struct _mod_maptable_t **mod_maptable_t**

Enumerations

- enum modulation_t { **MOD_PSK** = 0 , MOD_QAM , MOD_NUM }

Functions

- error_t Modulation_ListParameters (mod_par_t ∗ioParams)

  Function for retrieving and listing modulation parameters into dedicated structure.
- error_t Modulation_Mapper (const byte_stream_t ∗inStream, complex_stream_t ∗outStream, const mod_par_t ∗pParams)

  Function for byte to complex symbol stream mapping.
- error_t Modulation_HardDemapper (const complex_stream_t ∗inStream, byte_stream_t ∗outStream, const mod_par_t ∗pParams)

  Function for hard-demapping an input symbol stream into corresponding byte stream.
- error_t Modulation_SoftDemapper (const complex_stream_t ∗inStream, float_stream_t ∗outStream, const mod_par_t ∗pParams)

  Function for soft-demapping an input symbol stream into corresponding LLR stream.

6.17.1 Detailed Description

Modulation library header.

Author

Filippo Valmori

Date

26/08/2024

6.17.2 Macro Definition Documentation

6.17.2.1 MOD_TYPE_STR

```
#define MOD_TYPE_STR(
            x)
```

Value:

```
((x == MOD_PSK) ? "PSK" : \
(x == MOD_QAM) ? "QAM" : \
"N/A")
```

macro to convert modulation type value into string

6.17.3 Enumeration Type Documentation

6.17.3.1 modulation_t

```
enum modulation_t
```

- import error library
- import memory library
- import system library

Enumerator

MOD_QAM	• PSK modulation ID
MOD_NUM	• QAM modulation ID

6.17.4 Function Documentation

6.17.4.1 Modulation_HardDemapper()

```
error_t Modulation_HardDemapper (
            const complex_stream_t * inStream,
            byte_stream_t * outStream,
            const mod_par_t * pParams)
```

Function for hard-demapping an input symbol stream into corresponding byte stream.

Parameters

in	*inStream*	input stream
out	*outStream*	output stream
in	*pParams*	pointer to modulation parameters structure

Returns

error ID

- bit counter for output stream writing
- retrieve mapping table
- clear output buffer

6.17.4.2 Modulation_ListParameters()

```
error_t Modulation_ListParameters (
            mod_par_t * ioParams)
```

Function for retrieving and listing modulation parameters into dedicated structure.

Parameters

out	*ioParams*	pointer to i/o parameters structure to be filled

Returns

error ID

6.17.4.3 Modulation_Mapper()

```
error_t Modulation_Mapper (
            const byte_stream_t * inStream,
            complex_stream_t * outStream,
            const mod_par_t * pParams)
```

Function for byte to complex symbol stream mapping.

Parameters

in	*inStream*	input stream
out	*outStream*	output stream
in	*pParams*	pointer to modulation parameters structure

Returns

error ID

6.17.4.4 Modulation_SoftDemapper()

```
error_t Modulation_SoftDemapper (
            const complex_stream_t * inStream,
            float_stream_t * outStream,
            const mod_par_t * pParams)
```

Function for soft-demapping an input symbol stream into corresponding LLR stream.

Parameters

in	*inStream*	input stream
out	*outStream*	output stream
in	*pParams*	pointer to modulation parameters structure

Returns

error ID

6.18 modulation.h

Go to the documentation of this file.

```
00001
00010 #ifndef MODULATION_H
00011 #define MODULATION_H
00012
00013
00014 /*****************/
00015 /*** INCLUDES ***/
00016 /*****************/
00017
00018 #include "error.h"
00019 #include "memory.h"
00020 #include "system.h"
00024 /*************/
00025 /*** ENUMS ***/
00026 /*************/
00027
00028 typedef enum
00029 {
00030   MOD_PSK = 0,
00031   MOD_QAM,
00032   // keep NUM as final entry
00033   MOD_NUM
00034 } modulation_t;
00035
00036
00037
00038 /*******************/
00039 /*** PARAMETERS ***/
00040 /*******************/
00041
00042 #define MOD_TYPE          ((modulation_t) MOD_PSK)
00043 #define MOD_BPS           2u
00044
00045
00046
00047 /*****************/
00048 /*** CONSTANTS ***/
00049 /*****************/
00050
00051 #define MOD_SD_N0         ((float) 4.0)
00053 #define MOD_BPS_MIN       1u
00054 #define MOD_BPS_MAX       6u
00055
00056 #define MOD_ORDER         (0x01<<MOD_BPS)
00057 #define MOD_BINARY        2u
00058
00059 #define MOD_TYPE_STR(x)   ((x == MOD_PSK) ? "PSK" : \
00060                            (x == MOD_QAM) ? "QAM" : \
00061                            "N/A")
00062
00063 #if (MOD_BPS < MOD_BPS_MIN)
00064 #error MOD_BPS shall be positve!
00065 #endif
00066
00067
00068
00069 /*****************/
00070 /*** TYPEDEFS ***/
00071 /*****************/
00072
00073 typedef struct _mod_par_t
00074 {
```

```
00075   modulation_t type;
00076   uint8_t order;
00077   uint8_t bps;
00078   float phOfst;
00079 } mod_par_t;
00080
00081
00082 typedef struct _mod_maptable_t
00083 {
00084   byte_t bits[MOD_ORDER];
00085   complex_t symbs[MOD_ORDER];
00086 } mod_maptable_t;
00087
00088
00089
00090 /*************************/
00091 /*** PUBLIC PROTOTYPES ***/
00092 /*************************/
00093
00094 error_t Modulation_ListParameters( mod_par_t * ioParams );
00095 error_t Modulation_Mapper( const byte_stream_t * inStream, complex_stream_t * outStream, const
      mod_par_t * pParams );
00096 error_t Modulation_HardDemapper( const complex_stream_t * inStream, byte_stream_t * outStream, const
      mod_par_t * pParams );
00097 error_t Modulation_SoftDemapper( const complex_stream_t * inStream, float_stream_t * outStream, const
      mod_par_t * pParams );
00098
00099
00100 #endif
```

6.19 C:/Users/ValmoFil/Music/extra/tlc_chain/src/system.h File Reference

Dynamic memory allocation library header.

```
#include <stdio.h>
#include <stdbool.h>
#include <stdint.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
```

Data Structures

- struct _complex_t

Macros

- #define BY2BI_SHIFT 3u

  number of bit shifts to convert byte into bit value
- #define BY2BI_LEN(x)
- #define BI2BY_LEN(x)
- #define **LSBIT_MASK** ((uint8_t) 0x01)
- #define **LSBYTE_MASK** ((uint32_t) 0x0007)
- #define **NUM_BITS_PER_BYTE** 8u
- #define **BITIDX_1LAST** (NUM_BITS_PER_BYTE-1)
- #define **BITIDX_2LAST** (BITIDX_1LAST-1)
- #define **MATH_PI** 3.14159f
- #define **len_t** uint32_t

  bit/byte buffer length type
- #define **byte_t** uint8_t

  byte type

Typedefs

- typedef struct _complex_t **complex_t**

6.19.1 Detailed Description

Dynamic memory allocation library header.

System header.

Author

Filippo Valmori

Date

26/08/2024

Author

Filippo Valmori

Date

26/08/2024

Header file containing project common libraries and definitions.

6.19.2 Macro Definition Documentation

6.19.2.1 BI2BY_LEN

```
#define BI2BY_LEN(
                x)
```

Value:
```
((x)»BY2BI_SHIFT)
```

6.19.2.2 BY2BI_LEN

```
#define BY2BI_LEN(
                x)
```

Value:
```
((x)«BY2BI_SHIFT)
```

6.19.2.3 BY2BI_SHIFT

```
#define BY2BI_SHIFT 3u
```

number of bit shifts to convert byte into bit value

- import standard i/o library
- import boolean type library
- import integer types library
- import string library (e.g. to use "memcpy" and "memset" functions)
- import random generation library
- import time library (e.g. to link random seed to actual time)
- import mathematical library (e.g. to use "sin/cos" and "log/exp" functions)

6.20 system.h

Go to the documentation of this file.

```
00001
00012 #ifndef SYSTEM_H
00013 #define SYSTEM_H
00014
00015
00016 /*****************/
00017 /*** INCLUDES ***/
00018 /*****************/
00019
00020 #include <stdio.h>
00021 #include <stdbool.h>
00022 #include <stdint.h>
00023 #include <string.h>
00024 #include <stdlib.h>
00025 #include <time.h>
00026 #include <math.h>
00030 /******************/
00031 /*** CONSTANTS ***/
00032 /******************/
00033
00034 #define BY2BI_SHIFT             3u
00035 #define BY2BI_LEN(x)            ((x)«BY2BI_SHIFT)
00036 #define BI2BY_LEN(x)            ((x)»BY2BI_SHIFT)
00037 #define LSBIT_MASK              ((uint8_t) 0x01)
00038 #define LSBYTE_MASK             ((uint32_t) 0x0007)
00039 #define NUM_BITS_PER_BYTE       8u
00040 #define BITIDX_1LAST            (NUM_BITS_PER_BYTE-1)
00041 #define BITIDX_2LAST            (BITIDX_1LAST-1)
00042 #define MATH_PI                 3.14159f
00043
00044
00045
00046 /*****************/
00047 /*** TYPEDEFS ***/
00048 /*****************/
00049
00050 #define len_t                   uint32_t
00051 #define byte_t                  uint8_t
00052
00053 typedef struct _complex_t
00054 {
00055   float re;
00056   float im;
00057 } complex_t;
00058
00059
00060 #endif
```