FlexRAN Software Development Kit (SDK)

User Guide and API Reference

March 2021

v21.03

Intel Confidential

Document Number: 572002

Contents

Chapter 1

FlexRAN SDK

1.1 Revision history

Table 1.1 SDK revision history

Date	Version	Description
Mar 2021	21.03	FlexRAN 21.03 quarterly release. Addition of FlexRAN programmers guide.
Nov 2020	20.11	FlexRAN 20.11 quarterly release. Addition of FlexRAN programmers guide.
Aug 2020	20.08	FlexRAN 20.08 quarterly release. Addition of FlexRAN programmers guide.
Apr 2020	20.04	FlexRAN 20.04 quarterly release. Addition of FlexRAN programmers guide.
Feb 2020	20.02	FlexRAN 20.02 quarterly release. Addition of FlexRAN programmers guide.
Oct 2019	19.10	FlexRAN 19.10 quarterly release. Addition of FlexRAN programmers guide.
Jul 2019	19.06	FlexRAN 19.06 quarterly release. Addition of FlexRAN programmers guide.
Apr 2019	19.04-ea	FlexRAN 19.04-ea quarterly release. Addition of FlexRAN programmers guide.
Apr 2019	19.03	FlexRAN 19.03 quarterly release. Addition of FlexRAN programmers guide.
Dec 2018	18.12	FlexRAN 18.12 quarterly release. Addition of FlexRAN programmers guide.
Oct 2018	18.09	FlexRAN 18.09 quarterly release. Change of release naming convention
Jun 2018	1.6.0	FlexRAN Q2 2018 release
Mar 2018	1.5.0	FlexRAN Q1 2018 release

See FlexRAN Reference Solution Software Release Notes for release content detail.

1.2 Introduction

The FlexRAN Software Development Kit (SDK) provides a set of low-level wireless signal processing modules optimized for use on Intel® Architecture platforms. The modules cover both 4G and 5G wireless standards, and typically offer multiple optimised implementations to target specific platform ISAs.

Each module has its own individual source code folder, unit tests, and library.

This document is broken into several sections :

- FlexRAN SDK User manual Instructions for installing and configuring the SDK, running unit tests and checking results.
- FlexRAN SDK Programmers Guide Provides software architecture information, developer and contributor information, and optimization guidelines.
- API Documentation The documentation of the API for each of the modules in the File Documentation section.

1.3 Related documents

Table 1.2 SDK related documents

Document	Description	Document No./Location
FlexRAN Reference Solution Software Release Notes	Provides instructions for obtaining the software, the features of the software, known issues, and FAQ.	575822

Chapter 2

FlexRAN SDK User manual

2.1 Introduction

This section contains information for getting started with the FlexRAN SDK, including building the libraries, running the unit tests and performance tests and integrating with an application.

2.2 Installation instructions

2.2.1 System requirements and dependencies

The SDK can be built on any Linux∗ OS compatible with ICC. However, for optimum performance, it is recommended to use a platform configured according to the instructions from the FlexRAN Reference Solution Software Release Notes.

In addition this SDK release has the following dependencies:

Table 2.1 SDK software dependencies

Item	Description	Revision	Notes
ICC	Intel® compiler	19.0.3.206	ICC 2019 update 3
CMake	Cmake	3.9.2	Minimum 2.8.12
gtest	Google Test	1.7.0	Required to run the verification and compute performance tests
IPP	Integrated Performance Primitives	18.0	Required by some functions in SDK
mkl	Math Kernel Library	18.0	Required by some functions in SDK

2.2.2 Installation instructions

2.2.2.1 Google∗ test installation

Download googletest from https://github.com/google/googletest/releases
Untar google test. The recommended installation folder is /opt/gtest/

Example build and installation commands:

```
sudo tar -xvf googletest-release-1.7.0.tar.gz
sudo mv googletest-release-1.7.0 gtest-1.7.0
export GTEST_DIR=/opt/gtest/gtest-1.7.0

cd ${GTEST_DIR}
sudo g++ -isystem ${GTEST_DIR}/include -I${GTEST_DIR} -pthread -c ${GTEST_DIR}/src/gtest-all.cc
sudo ar -rv libgtest.a gtest-all.o
cd ${GTEST_DIR}/build-aux
sudo cmake ${GTEST_DIR}
sudo make
cd ${GTEST_DIR}
sudo ln -s build-aux/libgtest_main.a libgtest_main.a
```

2.3 Build instructions

The first step to build the SDK is the configuration of the environment variables required to configure the build.

2.3.1 SDK environment variables

2.3.1.1 WIRELESS_SDK_STANDARD

The WIRELESS_SDK_STANDARD is an OPTIONAL environment variable that specifies the wireless standard being targeted and therefore the SDK modules that will be built.

Valid values for WIRELESS_SDK_STANDARD are: lte, 5gnr, and all.

If WIRELESS_SDK_STANDARD is set to lte or 5gnr then only the SDK modules that support the selected wireless standard will be built.

If WIRELESS_SDK_STANDARD is set to all or is not set then all SDK modules will be built.

For example, to select only 5gnr SDK modules to be built use the following command:

```
export WIRELESS_SDK_STANDARD=5gnr
```

2.3.1.2 WIRELESS_SDK_TARGET_ISA

The WIRELESS_SDK_TARGET_ISA is an OPTIONAL environment variable that specifies the ISA that the SDK will be built for. If WIRELESS_SDK_TARGET_ISA is not set, it will auto-detect the maximum supported ISA of the build machine.

Valid values for WIRELESS_SDK_TARGET_ISA are: avx2 and avx512. For example to specify the SDK target ISA as AVX512 use the following command:

```
export WIRELESS_SDK_TARGET_ISA=avx512
```

2.3.1.3 CMAKE_BUILD_TYPE

The CMAKE_BUILD_TYPE is an OPTIONAL environment variable that defines the compiler flags used in the generated Makefiles.

Valid values for CMAKE_BUILD_TYPE are: Release, Debug RelWithDebInfo and MinSizeRel. For example o specify CMAKE_BUILD_TYPE Release use the following command:

```
export CMAKE_BUILD_TYPE=Release
```

The CMAKE_BUILD_TYPE options set the following compiler flags. These values can be updated in the file cmake/intel-compile-options.cmake

Table 2.2 CMAKE_BUILD_TYPE options

Option	Compiler flags
Release	-O3 -DNDEBUG
Debug	-O1 -g
RelWithDebInfo	-O2 -g -DNDEBUG
MinSizeRel	-Os -DNDEBUG

If CMAKE_BUILD_TYPE is not set, it will default to Release.

2.3.1.4 GTEST_ROOT

The GTEST_ROOT environment variable defines where gtest has been installed on the build system. If GTEST↩
_ROOT is not set, or is set to an invalid path, the unit tests will be omitted from the build.

To set GTEST_ROOT to the recommend gtest install location use the following command:

```
export GTEST_ROOT=/opt/gtest/gtest-1.7.0/
```

2.3.2 Makefiles generation

Once all the required environment variables have been set the makefiles can be created by executing the create-makefiles-linux.sh script.

```
./create-makefiles-linux.sh
```

The create-makefiles-linux.sh script checks the build environment and if correct will creates a build folder containing the makefiles to build the SDK. The name of the build folder is dependent on the ISA that is being targeted and is echoed to the screen.

The screen shot below shows example output on an AVX512 build system.

```
2018-03-20 07:31:15,441: GUEST [DEBUG]: : ./create-makefiles-linux.sh
INFO:  Environment variable GTEST_ROOT=/opt/gtest/gtest-1.7.0/
INFO:  Environment variable CMAKE_BUILD_TYPE not found, defaulting to Release
INFO:  Environment variable WIRELESS_SDK_TOOLCHAIN not found, defaulting to icc
INFO:  Environment variable WIRELESS_SDK_TARGET_ISA not found, auto-detecting avx512
INFO:  Environment variable RTE_SDK=/opt/dpdk-17.11/
       Note that the verification tests assume that huge pages of 1G have been set.
       Smaller huge page configurations will lead to page faults and degraded performance.
INFO:  Environment variable RTE_TARGET not found, defaulting to x86_64-native-linuxapp-icc
-- The CXX compiler identification is Intel 18.0.1.20171018
-- The C compiler identification is Intel 18.0.1.20171018
-- Check for working CXX compiler: /opt/intel/compilers_and_libraries_2018.1.163/linux/bin/intel64/icc
-- Check for working CXX compiler: /opt/intel/compilers_and_libraries_2018.1.163/linux/bin/intel64/icc -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Check for working C compiler: /opt/intel/compilers_and_libraries_2018.1.163/linux/bin/intel64/icc
-- Check for working C compiler: /opt/intel/compilers_and_libraries_2018.1.163/linux/bin/intel64/icc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
icc: remark #10397: optimization reports are generated in *.optrpt files in the output location
CMAKE_CXX_FLAGS_RELEASE is -O3 -DNDEBUG
-- Found GTest: /opt/gtest/gtest-1.7.0/libgtest.a
-- Looking for pthread.h
-- Looking for pthread.h - found
-- Looking for pthread_create
-- Looking for pthread_create - not found
-- Looking for pthread_create in pthreads
-- Looking for pthread_create in pthreads - not found
-- Looking for pthread_create in pthread
-- Looking for pthread_create in pthread - found
-- Found Threads: TRUE
-- Configuring done
-- Generating done
-- Build files have been written to: /home/jenkins/workspace/flexRAN/isg_cid-wireless_sdk/build-avx512-icc
```

2.3.3 SDK build

Once the Makefiles have been generated the SDK and unit tests can be built.

The build is performed from within the build folder that was generated by the create-makefiles-linux.sh script. The name of the build folder depends on the targeted ISA and follows the following format:

```
build-$WIRELESS_SDK_TARGET_ISA-icc
```

The commands below assume WIRELESS_SDK_TARGET_ISA=avx512. To build all kernels and unit tests, execute the following commands:

```
cd build-avx512-icc
make && make install
```

The kernel libraries and headers will be installed to the following locations:

```
---- build-avx512-icc
  |       |---- install
  |              |---- lib_crc
  |                   |---- libcrc.a
  |                   |---- phy_crc.h
  |              |---- lib_deinterleave
  |                   |---- libdeinterleave.a
  |                   |---- phy_deinterleave.h
                 etc
```

The unit test executables will be built in the following locations:

```
---- build-avx512-icc
  |       |---- test
  |              |---- phy
  |                   |---- test_crc
  |                   |---- test_deinterleave
                       etc
```

Alternatively the build can be limited to specific libraries and / or unit test executables. To do so, specify the names of the libraries (with the lib prefix) and tests (with the test_ prefix) to build.

For example the following commands will build only the sample_kernel library, the crc unit tests binary, and their dependencies (the common library and the crc library):

```
cd build-avx512-icc
make libsample_kernel test_crc && make install
```

2.4 Running the unit tests

Once the SDK has been built the unit tests can be run.

The DIR_WIRELESS_SDK environment variable is required at run-time by the SDK and unit tests. The DIR_W↩ IRELESS_SDK environment variable should be set to the full path of the build directory that was created by the create-makefiles-linux.sh script, for example (example shown for avx512):

```
export DIR_WIRELESS_SDK=/install_path/sdk/build-avx512-icc
```

The build creates a single unit test binary for each kernel containing all the tests for that kernel. The unit tests are separated into "Check" (verification tests, checking functionality) and "Perf" (Compute performance tests, measuring cycle counts) tests. The following text describes how to run the unit tests for the lib_llr_demapping kernel. However, the instructions are applicable to any kernel.

First go to the test directory for the kernel that is being tested:

```
cd build-avx512-icc/test/phy/test_llr_demapping
```

The test folder should contain a test binary named unittests and a test_vectors folder To view all the available tests supported execute the following command:

```
./unittests --gtest_list_tests
```

2.4.1 Running all tests

To run all the tests simply execute the unittests binary with no arguments:

```
./unittests
```

The tests will echo information to the screen as they run, for example:

```
[==========] Running 90 tests from 2 test cases.
[----------] Global test environment set-up.
[----------] 42 tests from LlrDemappingCheck
[ RUN      ] LlrDemappingCheck.QPSK_C_FP32_Case1
[       OK ] LlrDemappingCheck.QPSK_C_FP32_Case1 (8 ms)
[ RUN      ] LlrDemappingCheck.QAM16_C_FP32_Case2
[       OK ] LlrDemappingCheck.QAM16_C_FP32_Case2 (6 ms)
etc...
```

When all the tests have completed the results of the test will be shown:

```
[----------] Global test environment tear-down
[==========] 90 tests from 2 test cases ran. (32407 ms total)
[  PASSED  ] 90 tests.
```

To run only the consistency tests use the following command:

```
./unittests --gtest_filter=*Check*
```

To run only the performance tests use the following command:

```
./unittests --gtest_filter=*Perf*
```

The test results are written to a JUnit compatible xml file in kernels test directory:

```
test_results.xml
```

NOTE: Alternatively the tests can be run in the Intel® Software Developmemt Emulator (SDE), for example:

```
sde64 -- ./unittests --gtest_filter=*Check*
```

2.4.2 Analysing performance with SDE mix and IACA

As part of the SDK optimisation flow, a python script was developed to take the output of running a test bench using Intel SDE with the mix option, and analyse the most time consuming code segments with the Intel® Architecture Code Analyzer (IACA) tool.

To use the mix_to_iaca.py script:

Ensure that SDE is installed and on the PATH.

Ensure that IACA is installed and point the IACA_PATH environment variable to the IACA binary PATH.

```
sde64 -version
iaca -v
```

Compile an SDK test bench.

Run the SDK testbench using the sde mix framework, for example:

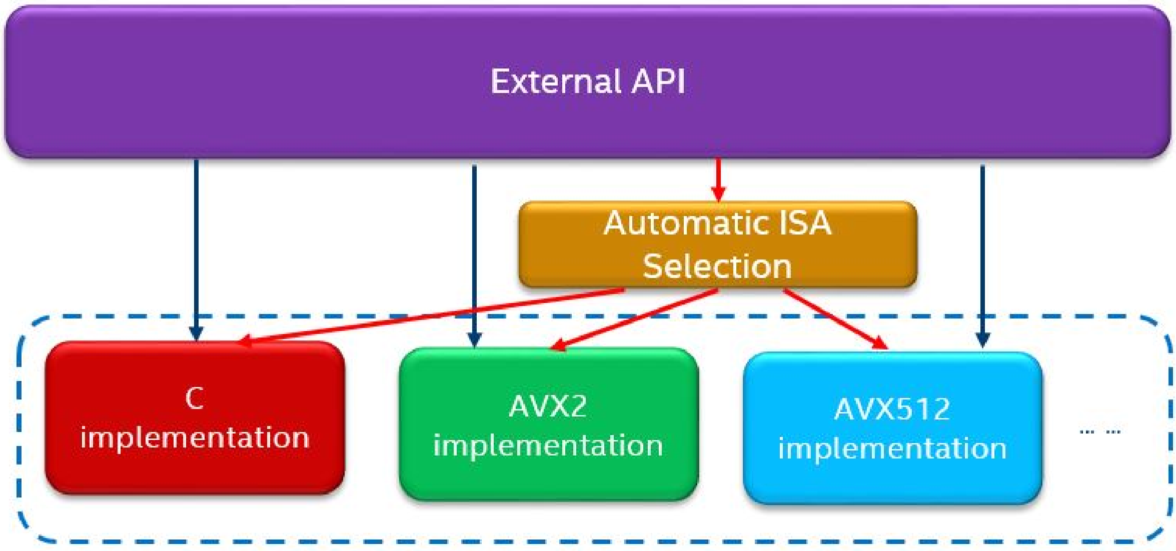
```
sde64 -mix - ./unittests --nb_loops=1000 --gtest_filter=*Perf*
```

Pipe the mix file to IACA and inspect output

```
$DIR_WIRELESS_SDK/test/scripts/mix_to_iaca.py sde-mix-out.txt
vi sde-mix-out.iaca
```

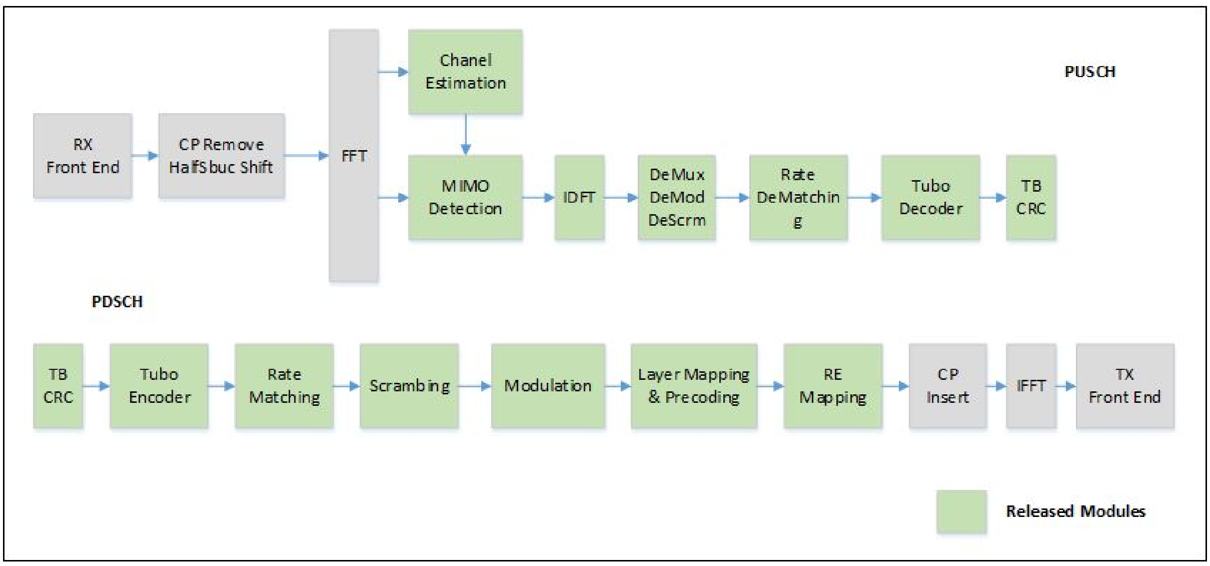
2.5 Using the SDK library in an application

The SDK modules typically offer multiple optimised implementations to target specific platform ISAs. The default API for each module calls the version built for the target ISA that was specified when generating the makefiles. Each module also contains ISA specific APIs so that a user can choose to override the target ISA and specify manually which version to call.

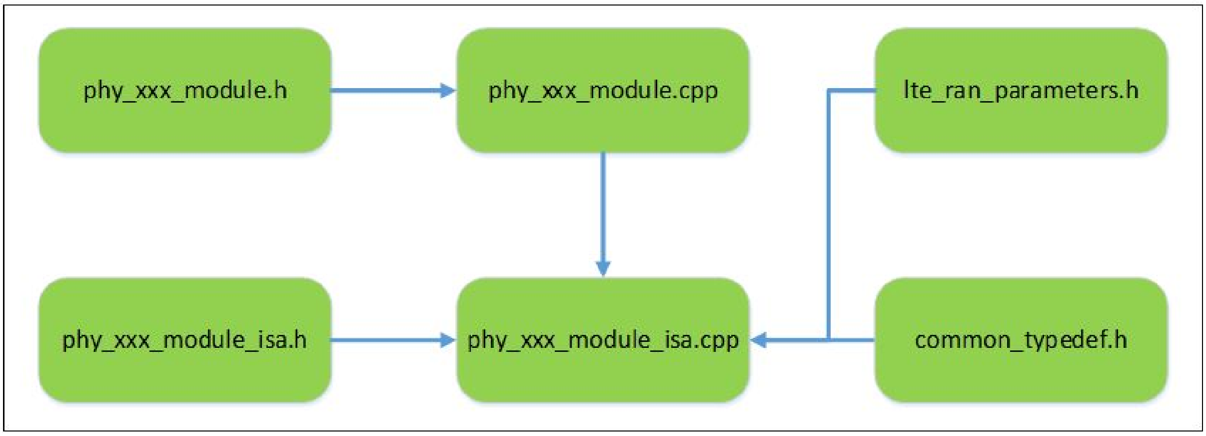


Each SDK module uses a common API style, with a single 'request' input structure and a single 'response' output structure. All data memory/buffers in a module's request/response structures must be allocated by the calling function.

A diagram of an example LTE wireless protocol stack system built using SDK modules is shown:



where the following diagram shows the relationship between the source files of a typical SDK module:



- The file phy_xxx_module.h contains the public API of the module defining the request/response structures and functions calls supported. The default implementation is defined by the target ISA.
- The file phy_xxx_module.cpp contains the entry functions of the module library. Here is where the specific implementation for the target ISA is called.
- The files phy_xxx_module_isa.h /phy_xxx_module_isa.cpp define the ISA specific implementations of the module. There may be multiple implementations depending on the ISA's supported by the module.

Chapter 3

FlexRAN SDK Programmers Guide

3.1 Introduction

The purpose of this section is to introduce standards and conventions that have to be followed by any person contributing to FlexRAN SDK. It consist of description of certain BKMs (Best known Methods) and links to resources that are essential (or useful) in the development process.

3.2 Coding Style

The API and header files exposing the API shall conform to a C standard implementation and shall follow the DPDK coding style — Data Plane Development Kit documentation with exceptions. http://doc.dpdk.org/guides/contributing/coding_style.html

There are two exceptions where DPDK coding style should not be followed:

- 4 spaces are preferred over tabs
- Comments should follow doxygen format as described in the section 5.

Internal Implementations of the library functions shall also follow the same coding style, however they shall not be limited to C code but also support C++ 11 std.

3.3 Naming Conventions

The following naming conventions should be followed at all times.

3.3.1 Module Naming Convention

Item	Convention	Examples
<kernel name>	Lowercase, underscore separated If 5GTF PHY specific must end with "_5gtf" If 5GNR PHY specific must end with with "_5gnr"	rate_matching rate_matching_5gtf rate_matching_5gnr
kernel src dir	lib_<kernelname>	source/phy/lib_rate_matching
kernel library name	lib<kernelname>.a	librate_matching.a
kernel public header	phy_<kernelname>.h	phy_rate_matching.h
kernel private header	phy_<kernelname>*XXXXX.h*	*phy_rate_matching_internal.h*
kernel src files	*phy*<kernelname>.cpp phy_<kernelname>*XXXXX.cpp*	*phy_rate_matching.cpp* *phy_rate_matching_avx512.cpp*
kernel test dir	*test*<kernelname>	test/phy/test_rate_matching
kernel test vector dir	test_<kernelname>/test_vectors	test/phy/test_rate_matching/test_vectors
kernel test table dir	test_<kernelname>/test_tables	test/phy/test_rate_matching/test_tables
kernel conf file	test_<kernelname>/test.conf	test/phy/test_rate_matching/test.conf

3.3.2 Test Bench Naming Convention

Item	Conventions	Example
performance test only file	<kernel_name>*performance.cc*	*modulation_performance.cc*
functional test only file	*<kernel_name>_functional.cc*	*modulation_functional.cc*
performance and functional tests class	*<KernelName>Test*	*ModulationTest*
performance tests class	*<KernelName>Perf*	*ModulationPerf*
functional tests class	*<KernelName>Check*	*ModulationCheck*
module_name	*<kernel_name>(-<implementation>)(-<fxp|flp|...>)*	*llr_demapping-impl_a-fxp*
Performance test case	<*case*>(any_other_parameter)	QPSK or QPSK_FLOAT32
Functional test case	<case>_(any_other_parameter)	QPSK

3.4 API Standards

Only headers files that are used in the declaration of the API should be included.

For example if one of the functions declared in the header file returns value that is uint8_t type then stdint header should be included, but not any extra headers. API declaration must be wrapped in the extern "C" clause:

```
#ifdef __cplusplus
extern "C" {
#endif

// Code goes here

#ifdef __cplusplus
}
#endif
```

Only function and structures that are exposed to client should be placed in the external header file.

All the API function should follow request/response convention where each function takes two arguments: first is the pointer to the const structure with all input parameters; and second one is the pointer to the structure with the function output(s). It is advisable to include length of the output in the response structure even if input and output length are the same. It allows response to be self contained. For example for scramble kernel request/response structures and a function declaration are as followed:

```
struct bblib_scramble_request {
    uint8_t* data_in;
    uint32_t c_init;
    uint32_t len;
};

struct bblib_scramble_response {
    uint8_t* data_out;
    int32_t len;
};

int32_t
bblib_scramble(const struct bblib_scramble_request *request, struct
      bblib_scramble_response *response);
```

Structures and functions names should be prefixed with bblib_.

For each kernel functions generic and ISA specific functions should be declared.

Generic function when called will be used the best available ISA, whereas ISA specific functions call specific implementation directly.

```
int bblib_kernel()          //generic version, implementation chosen based on build
int bblib_kernel_c()        //C compliant implementation, run on any x86 CPU type
int bblib_kernel_sse()      //SSE implementation, CPU must support sse4.2 instruction set
int bblib_kernel_avx()      //AVX2 implementation, CPU must support avx2 instruction set
int bblib_kernel_avx512()   //AVX512 implementation, CPU must support avx512 instruction set
```

3.5 Module Initialization

Any initialization required by the function shall be performed through a constructor function and not require any initialization setup by the application.

Constructors are implemented in C via structures defined in the code. An example of the constructor implementation is outline below. Note for kernels that contain multiple functions or implementations it is important to define the initialization constructor in the same file as the API's are implemented.

This way only the functions used by an application are initialized.

```
struct init_kernelname_function
{
    init_kernelname_function()
    {
       // Do initialization here

       // Print Kernel Version in constructor
       bblib_kernename_print_version();
    }
};

// Constructor defined in same file as function implementation
int bblib_kernelname_function() {
     // Do something
}

init_kernelname_function do_constructor; //define struct here to do initialization
```

3.6 Doxygen Comments

Qt C++ Doxygen comments style has been employed in headers files. This page: https://www.stack.nl/~dimitri/doxygen/manual/docblocks.html is a good source of the information and can be refereed in the case something is not covered in this chapter.

At the top of the file following comments should be included, where file name is the name of the header file and brief description is a short description of the module:

```
// COPYRIGHT_TAG

/*!
 *    \file   file_name
 *    \brief  Brief description.
 *\/
```

In any case where brief description is included detailed description can be included. It should be placed after the brief description with the empty line separating these two types of descriptions.

Enum should be followed by the following comment, where name of enum is the name of the enum type:

```
/*!
 *    \enum name_of_enum
 *    \brief Brief description.
 *\/
```

Each element of the enum should be followed by the comment with description of the element:

```
/*!< Description. *\/
```

Structures should use following comment, where structure name is the name of the structure:

```
/*!
 *    \struct name_of_the_structure
 *    \brief Brief description.
 *\/
```

Each structure element should be followed by the comment explaining this element. The same type of comment as for enum should be used.

Each function comment should contain brief description of the function (detailed one is optional), input and output parameters, optional notes and return value (in case of the void function return comment can be skipped). Parameter comment consists of 4 parts: the parameter tag, [in], [out] or [inout] indicator, name of the parameter and brief description. Following example shows how function can be documented:

```
//  \param [in] input_param Brief description .
//  \param [out] output_param Brief description.
//  \note Optional note.
//  \return Description of return values.
```

Each generic and ISA specific functions providing the same functionality should be wrapped in the group comment and only generic function should be documented. A generic function should be the first function in the group. Following comments are used to group functions:

```
    //! @{
    // All functions go here
    //! @}
```

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

File Index

5.1 File List

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

Chapter 6

Data Structure Documentation

6.1 bblib_beamforming_dl_expand_request Struct Reference

```
#include <phy_beamforming_dl_expand.h>
```

Data Fields

- int16_t * pUlCh [BBLIB_BF_MAX_RX_ANT_NUM][BBLIB_BF_MAX_UL_LAYER_NUM]
- int16_t nLayer
- int16_t nRxAnt
- int16_t nStart
- int16_t nLen

6.1.1 Detailed Description

Request struct of DL expanding for beamforming.

6.1.2 Field Documentation

6.1.2.1 nLayer

```
int16_t bblib_beamforming_dl_expand_request::nLayer
```

Number of Layers - valid values 4 for 32 nRxAnt or 8/4/2/1 for 64 nRxAnt

6.1.2.2 nLen

```
int16_t bblib_beamforming_dl_expand_request::nLen
```

Number of matrices

6.1.2.3 nRxAnt

```
int16_t bblib_beamforming_dl_expand_request::nRxAnt
```

Number of Rx antennas - valid values 32 or 64

6.1.2.4 nStart

```
int16_t bblib_beamforming_dl_expand_request::nStart
```

Start point

6.1.2.5 pUlCh

```
int16_t* bblib_beamforming_dl_expand_request::pUlCh[BBLIB_BF_MAX_RX_ANT_NUM][BBLIB_BF_MAX_UL_LAYER_NUM]
```

Data pointer points to UL channel

6.2 bblib_beamforming_dl_expand_response Struct Reference

```
#include <phy_beamforming_dl_expand.h>
```

Data Fields

- int16_t * pDlCh [BBLIB_BF_MAX_DL_LAYER_NUM][BBLIB_BF_MAX_RX_ANT_NUM]

6.2.1 Detailed Description

Response struct of DL expanding for beamforming.

6.2.2 Field Documentation

6.2.2.1 pDlCh

```
int16_t* bblib_beamforming_dl_expand_response::pDlCh[BBLIB_BF_MAX_DL_LAYER_NUM][BBLIB_BF_MAX_RX_ANT_NUM]
```

Data pointer points to DL channel

6.3 bblib_cestimate_pucch_part1_request Struct Reference

```
#include <phy_cestimate_pucch.h>
```

Data Fields

- int8_t pucch_format
- int32_t num_rx_ants
- int16_t input_offset
- int16_t num_ul_symb
- int16_t df
- uint32_t Fs
- int16_t num_ul_rb
- int16_t num_used_e
- int16_t pucch_shortened_flag
- uint16_t rb_start
- int16_t catm_enable
- int16_t pucch_rf_tuning_symbols
- int16_t num_pilots_slot
- int16_t ∗ num_orth_cover
- int16_t ∗ sdescramb
- void ∗ ah_est [MAX_SYM_PER_SUBFRAME_SDK][MAX_NUM_ANT_CEST_PUCCH]
- void ∗ r_alpha_uv [MAX_SYM_PER_SUBFRAME_SDK]
- float ∗ err_avg
- void ∗ chan_est_ptr [MAX_NUM_SLOTS][MAX_NUM_ANT_CEST_PUCCH]

6.3.1 Detailed Description

Request structure for PUCCH chan estimator, part 1.

6.3.2 Field Documentation

6.3.2.1 ah_est

```
void* bblib_cestimate_pucch_part1_request::ah_est[MAX_SYM_PER_SUBFRAME_SDK][MAX_NUM_ANT_CEST_PUCCH]
```

Array of pointers to int16_t. Points to an array of up to (MAX_NUM_SUBCARRIERS) complex pairs, each complex pair comprising two int16_t.

6.3.2.2 catm_enable

```
int16_t bblib_cestimate_pucch_part1_request::catm_enable
```

catm enable flag.

6.3.2.3 chan_est_ptr

```
void* bblib_cestimate_pucch_part1_request::chan_est_ptr[MAX_NUM_SLOTS][MAX_NUM_ANT_CEST_PUCCH]
```

2D array of pointers to 16 element arrays of complex float pair.

6.3.2.4 df

`int16_t bblib_cestimate_pucch_part1_request::df`

df

6.3.2.5 err_avg

`float* bblib_cestimate_pucch_part1_request::err_avg`

Error average. Pointer to 2x float complex pair.

6.3.2.6 Fs

`uint32_t bblib_cestimate_pucch_part1_request::Fs`

Fs

6.3.2.7 input_offset

`int16_t bblib_cestimate_pucch_part1_request::input_offset`

row_a_fft_size - K

6.3.2.8 num_orth_cover

`int16_t* bblib_cestimate_pucch_part1_request::num_orth_cover`

Pointer to array of k_num_slots orthogonal covers, format 1 only.

6.3.2.9 num_pilots_slot

`int16_t bblib_cestimate_pucch_part1_request::num_pilots_slot`

Number of pilots

6.3.2.10 num_rx_ants

`int32_t bblib_cestimate_pucch_part1_request::num_rx_ants`

Number of RX antennas.

6.3.2.11 num_ul_rb

`int16_t bblib_cestimate_pucch_part1_request::num_ul_rb`

Number of UL Resource Blocks.

6.3.2.12 num_ul_symb

```
int16_t bblib_cestimate_pucch_part1_request::num_ul_symb
```

Number of PUCCH symbols per slot.

6.3.2.13 num_used_e

```
int16_t bblib_cestimate_pucch_part1_request::num_used_e
```

num_used_e

6.3.2.14 pucch_format

```
int8_t bblib_cestimate_pucch_part1_request::pucch_format
```

Values in range of enum cestimate_pucch_phy_formats.

6.3.2.15 pucch_rf_tuning_symbols

```
int16_t bblib_cestimate_pucch_part1_request::pucch_rf_tuning_symbols
```

PUCCH RF tuning symbols.

6.3.2.16 pucch_shortened_flag

```
int16_t bblib_cestimate_pucch_part1_request::pucch_shortened_flag
```

PUCCH shortening flag.

6.3.2.17 r_alpha_uv

```
void* bblib_cestimate_pucch_part1_request::r_alpha_uv[MAX_SYM_PER_SUBFRAME_SDK]
```

Pointers to arrays of NRB_SC complex int16_t pairs, One per symbol.

6.3.2.18 rb_start

```
uint16_t bblib_cestimate_pucch_part1_request::rb_start
```

RB_start

6.3.2.19 sdescramb

```
int16_t* bblib_cestimate_pucch_part1_request::sdescramb
```

Pointer to array of k_num_slots sdescramb, format 1 only.

6.4 bblib_cestimate_pucch_part1_response Struct Reference

```
#include <phy_cestimate_pucch.h>
```

Data Fields

- float all_pucch_pwr_avg_rb
- float rx_in_pwr_avg
- float pucch_pwr_avg
- int8_t timing_advance_pucch
- void * ch_est_pucch
- void * d_est_ack
- void * d_est_cqi

6.4.1 Detailed Description

Response structure for PUCCH chan estimator, part 1.

6.4.2 Field Documentation

6.4.2.1 all_pucch_pwr_avg_rb

```
float bblib_cestimate_pucch_part1_response::all_pucch_pwr_avg_rb
```

All PUCCH power average

6.4.2.2 ch_est_pucch

```
void* bblib_cestimate_pucch_part1_response::ch_est_pucch
```

Channel Estimate PUCCH

6.4.2.3 d_est_ack

```
void* bblib_cestimate_pucch_part1_response::d_est_ack
```

Channel Estimate ACK

6.4.2.4 d_est_cqi

```
void* bblib_cestimate_pucch_part1_response::d_est_cqi
```

Channel Estimate CQI. Format 2 only

6.4.2.5 pucch_pwr_avg

```
float bblib_cestimate_pucch_part1_response::pucch_pwr_avg
```

PUCCH power average

6.4.2.6 rx_in_pwr_avg

```
float bblib_cestimate_pucch_part1_response::rx_in_pwr_avg
```

RX IN power average

6.4.2.7 timing_advance_pucch

```
int8_t bblib_cestimate_pucch_part1_response::timing_advance_pucch
```

PUCCH timing advance

6.5 bblib_cestimate_pucch_pilot_mul_request Struct Reference

```
#include <phy_cestimate_pucch.h>
```

Data Fields

- int16_t num_pilots_slot
- int8_t multi_rb_check_rb_start
- int16_t ue_err_expo
- int32_t ue_err_avg_ch [2]
- void ∗ pucch_pilot [NUM_SLOTS_PER_SUBF_CEST_PUCCH][MAX_NUM_PILOT_SYM_PER_SLOT]
- void ∗ ah_est [MAX_NUM_ANT_CEST_PUCCH][NUM_SLOTS_PER_SUBF_CEST_PUCCH][MAX_NUM_PILOT_SYM_PER_
- int32_t num_antennas

6.5.1 Detailed Description

Request structure for the pilots multiplication function in PUCCH channel estimation.

6.5.2 Field Documentation

6.5.2.1 ah_est

```
void* bblib_cestimate_pucch_pilot_mul_request::ah_est[MAX_NUM_ANT_CEST_PUCCH][NUM_SLOTS_PER_SUBF_CEST_PUCCH][M
```

Pointer to fft output buffer for all symbols in slot and rx antennas. Each pointer points to buffer for 16 complex numbers (16 chosen for vectorization), that contains NRB_SC = 12 valid complex numbers. For fixed point mode complex numbers are in fixed point Q16s12 format, type complex_int16_t For floating point mode complex numbers are in floating point format, type std::complex<float>

6.5.2.2 multi_rb_check_rb_start

```
int8_t bblib_cestimate_pucch_pilot_mul_request::multi_rb_check_rb_start
```

Multi RB check for RB start. Set to 1 if there will be PUCCH residing on the same resource block.

6.5.2.3 num_antennas

```
int32_t bblib_cestimate_pucch_pilot_mul_request::num_antennas
```

The number of antennas.

6.5.2.4 num_pilots_slot

```
int16_t bblib_cestimate_pucch_pilot_mul_request::num_pilots_slot
```

Number of pilots per slot (MAX_NUM_PILOT_SYM_PER_SLOT = 3). If doing simSR with format 2s have to be changed number of pilot symbols to be format 1s to decode correctly.

6.5.2.5 pucch_pilot

```
void* bblib_cestimate_pucch_pilot_mul_request::pucch_pilot[NUM_SLOTS_PER_SUBF_CEST_PUCCH][MAX_NUM_PILOT_SYM_PE
```

Pointer to locally generated pilot symbols for the PUCCH. Points to buffer length NRB_SC = 12 of valid complex numbers. For fixed point mode complex numbers are in fixed point Q16s12 format, type complex_int16_t For floating point mode complex numbers are in floating point format, type std::complex<float>

6.5.2.6 ue_err_avg_ch

```
int32_t bblib_cestimate_pucch_pilot_mul_request::ue_err_avg_ch[2]
```

UE Variable for PUCCH (integer value [Re Im])

6.5.2.7 ue_err_expo

```
int16_t bblib_cestimate_pucch_pilot_mul_request::ue_err_expo
```

UE Variable for PUCCH

6.6 bblib_cestimate_pucch_pilot_mul_response Struct Reference

```
#include <phy_cestimate_pucch.h>
```

Data Fields

- void * chan_est [MAX_NUM_ANT_CEST_PUCCH][NUM_SLOTS_PER_SUBF_CEST_PUCCH][MAX_NUM_PILOT_SYM_PE
- complex_float avg_err

6.6.1 Detailed Description

Response structure for the pilots multiplication function in PUCCH channel estimation.

6.6.2 Field Documentation

6.6.2.1 avg_err

```
complex_float bblib_cestimate_pucch_pilot_mul_response::avg_err
```

Average error as floating point complex number.

6.6.2.2 chan_est

```
void* bblib_cestimate_pucch_pilot_mul_response::chan_est[MAX_NUM_ANT_CEST_PUCCH][NUM_SLOTS_PER_SUBF_CEST_PUCCH
```

Array of pointers to channel estimation as complex numbers. Each pointer points to buffer for 16 complex numbers (16 chosen for vectorization), that contains NRB_SC = 12 valid complex numbers. For fixed point mode complex numbers are in fixed point Q16s12 format, type complex_int16_t For floating point mode complex numbers are in floating point format, type std::complex<float>

6.7 bblib_cestmate_request Struct Reference

```
#include <phy_cestimate.h>
```

Data Fields

- PLTE_BS_RX_COMMON_PARAMS pRxCommonParams
- PLTE_BS_RX_DATA_PARAMS pRxPuschInputParams
- PLTE_BS_RX_DATA_FEC_PARAMS pUlRxFecPa
- PLTE_BS_RX_DATA_PTRS pRxPuschDemodFlPtrs

6.7.1 Detailed Description

Request structure for cestmate.

6.7.2 Field Documentation

6.7.2.1 pRxCommonParams

`PLTE_BS_RX_COMMON_PARAMS bblib_cestmate_request::pRxCommonParams`

common parameters

6.7.2.2 pRxPuschDemodFlPtrs

`PLTE_BS_RX_DATA_PTRS bblib_cestmate_request::pRxPuschDemodFlPtrs`

receive data

6.7.2.3 pRxPuschInputParams

`PLTE_BS_RX_DATA_PARAMS bblib_cestmate_request::pRxPuschInputParams`

pusch parmeters

6.7.2.4 pUlRxFecPa

`PLTE_BS_RX_DATA_FEC_PARAMS bblib_cestmate_request::pUlRxFecPa`

fec parameters

6.8 bblib_cestmate_response Struct Reference

`#include <phy_cestimate.h>`

Data Fields

- float * ahEstPtr [MAX_SYM_PER_SUBFRAME][MAX_NUM_ANT]

6.8.1 Detailed Description

Response structure for cestmate.

6.8.2 Field Documentation

6.8.2.1 ahEstPtr

```
float* bblib_cestmate_response::ahEstPtr[MAX_SYM_PER_SUBFRAME][MAX_NUM_ANT]
```

the FFT's output data

6.9 bblib_channel_estimation_5gnr_request Struct Reference

```
#include <phy_cestimate_5gnr.h>
```

Data Fields

- int16_t n_dmrs_config_type
- int16_t n_rxant
- int16_t n_layer
- int16_t n_layer_in_group
- int16_t n_start_prb
- int16_t n_prb
- int16_t n_mu
- int16_t n_fft_size
- int16_t n_fl_dmrs_symb
- int16_t n_dmrs_symb
- uint16_t n_dmrs_symb_idx [CE_MAX_DMRS_SYMBOL]
- int16_t n_data_symb
- float f_time_interp_coeff [CE_MAX_DMRS_SYMBOL][CE_N_SYMB_PER_SLOT]
- int16_t n_dmrs_port [CE_MAX_TX_LAYER_NUM]
- float f_boost_linear [CE_MAX_TX_LAYER_NUM]
- int16_t n_enable_ta_est
- int16_t n_enable_ta_comp
- int16_t n_enable_doppler_est
- int16_t n_interp_method
- int16_t n_delay_spread_index
- int16_t * p_dmrs_base_seq [CE_MAX_TX_LAYER_NUM][CE_MAX_DMRS_SYMBOL]
- int16_t * p_dmrs_seq [CE_MAX_TX_LAYER_NUM][CE_MAX_DMRS_SYMBOL]
- int16_t * p_ce_in [CE_MAX_RX_ANT_NUM][CE_MAX_DMRS_SYMBOL]
- int16_t * p_ce_dct_buff
- int16_t n_priori_ta
- int16_t n_enable_fo_comp

6.9.1 Detailed Description

Structure defines the Channel Estimation and TA function request interface in 5GNR.

6.9.2 Field Documentation

6.9.2.1 f_boost_linear

```
float bblib_channel_estimation_5gnr_request::f_boost_linear[CE_MAX_TX_LAYER_NUM]
```

DMRS boosting linear value

6.9.2.2 f_time_interp_coeff

```
float bblib_channel_estimation_5gnr_request::f_time_interp_coeff[CE_MAX_DMRS_SYMBOL][CE_N_SY←↩
MB_PER_SLOT]
```

Time domain interpolation coefficient

6.9.2.3 n_data_symb

```
int16_t bblib_channel_estimation_5gnr_request::n_data_symb
```

Number of sheduled data symbols

6.9.2.4 n_delay_spread_index

```
int16_t bblib_channel_estimation_5gnr_request::n_delay_spread_index
```

delay spread index to indicate different interpolation weight matrix

6.9.2.5 n_dmrs_config_type

```
int16_t bblib_channel_estimation_5gnr_request::n_dmrs_config_type
```

DMRS configuration type: 1 or 2

6.9.2.6 n_dmrs_port

```
int16_t bblib_channel_estimation_5gnr_request::n_dmrs_port[CE_MAX_TX_LAYER_NUM]
```

DMRS port, type1: 0∼7; type2: 0∼11

6.9.2.7 n_dmrs_symb

```
int16_t bblib_channel_estimation_5gnr_request::n_dmrs_symb
```

Number of DMRS symbols

6.9.2.8 n_dmrs_symb_idx

```
uint16_t bblib_channel_estimation_5gnr_request::n_dmrs_symb_idx[CE_MAX_DMRS_SYMBOL]
```

Dmrs Symbol index

6.9.2.9 n_enable_doppler_est

```
int16_t bblib_channel_estimation_5gnr_request::n_enable_doppler_est
```

flag shows whether enable doppler estimation

6.9.2.10 n_enable_fo_comp

```
int16_t bblib_channel_estimation_5gnr_request::n_enable_fo_comp
```

FO compensation flag

6.9.2.11 n_enable_ta_comp

```
int16_t bblib_channel_estimation_5gnr_request::n_enable_ta_comp
```

bitmap flag shows whether enable TA compensation bit0: TA pre-compensation; bit 1: TA post-compensation

6.9.2.12 n_enable_ta_est

```
int16_t bblib_channel_estimation_5gnr_request::n_enable_ta_est
```

flag shows whether enable TA estimation

6.9.2.13 n_fft_size

```
int16_t bblib_channel_estimation_5gnr_request::n_fft_size
```

FFT size

6.9.2.14 n_fl_dmrs_symb

```
int16_t bblib_channel_estimation_5gnr_request::n_fl_dmrs_symb
```

Number of front loaded DMRS symbols

6.9.2.15 n_interp_method

```
int16_t bblib_channel_estimation_5gnr_request::n_interp_method
```

flag shows which interpolation method used, 0∼3

6.9.2.16 n_layer

`int16_t bblib_channel_estimation_5gnr_request::n_layer`

Number of Layers to process

6.9.2.17 n_layer_in_group

`int16_t bblib_channel_estimation_5gnr_request::n_layer_in_group`

Number of Layers in MU pair

6.9.2.18 n_mu

`int16_t bblib_channel_estimation_5gnr_request::n_mu`

Numerology, determine sub carrier spacing, Value: 0->4

6.9.2.19 n_prb

`int16_t bblib_channel_estimation_5gnr_request::n_prb`

Number of PRBs

6.9.2.20 n_priori_ta

`int16_t bblib_channel_estimation_5gnr_request::n_priori_ta`

priori information for TA pre-compensation

6.9.2.21 n_rxant

`int16_t bblib_channel_estimation_5gnr_request::n_rxant`

Number of Rx antennas

6.9.2.22 n_start_prb

`int16_t bblib_channel_estimation_5gnr_request::n_start_prb`

Starting PRB number

6.9.2.23 p_ce_dct_buff

`int16_t* bblib_channel_estimation_5gnr_request::p_ce_dct_buff`

CE internal buffer, length is n_prb*768 byte

6.9.2.24 p_ce_in

```
int16_t* bblib_channel_estimation_5gnr_request::p_ce_in[CE_MAX_RX_ANT_NUM][CE_MAX_DMRS_SYMBOL]
```

Data pointer points to nRx∗nSymbol CE input buffer, format 16S13

6.9.2.25 p_dmrs_base_seq

```
int16_t* bblib_channel_estimation_5gnr_request::p_dmrs_base_seq[CE_MAX_TX_LAYER_NUM][CE_MAX_D↩
MRS_SYMBOL]
```

Data pointer points to DMRS base sequence, format 16S14

6.9.2.26 p_dmrs_seq

```
int16_t* bblib_channel_estimation_5gnr_request::p_dmrs_seq[CE_MAX_TX_LAYER_NUM][CE_MAX_DMRS_S↩
YMBOL]
```

Data pointer points to DMRS sequence for each tx layers, format 16S14

6.10 bblib_channel_estimation_5gnr_response Struct Reference

```
#include <phy_cestimate_5gnr.h>
```

Data Fields

- int16_t ∗ p_ce_ls [CE_MAX_TX_LAYER_NUM][CE_MAX_RX_ANT_NUM]
- int16_t ∗ p_ce_out [CE_MAX_RX_ANT_NUM][CE_MAX_TX_LAYER_NUM][CE_N_SYMB_PER_SLOT]
- int16_t ∗ p_irc_IpN [CE_MAX_RX_ANT_NUM][CE_MAX_DMRS_SYMBOL]
- int16_t ∗ p_ce_ta_comp [CE_MAX_TX_LAYER_NUM]
- float f_est_sigma2 [CE_MAX_DMRS_SYMBOL]
- float f_power_bits [CE_MAX_RX_ANT_NUM][CE_MAX_TX_LAYER_NUM][CE_MAX_DMRS_SYMBOL]
- float f_est_snr [CE_MAX_DMRS_SYMBOL][CE_MAX_TX_LAYER_NUM][CE_MAX_RX_ANT_NUM]
- float f_est_doppler_shift [CE_MAX_TX_LAYER_NUM]
- float f_est_cfo [CE_MAX_TX_LAYER_NUM]
- int16_t n_est_ta [CE_MAX_TX_LAYER_NUM][CE_MAX_DMRS_SYMBOL]

6.10.1 Detailed Description

Structure defines the Channel Estimation and TA function response interface in 5GNR.

6.10.2 Field Documentation

6.10.2.1 f_est_cfo

```
float bblib_channel_estimation_5gnr_response::f_est_cfo[CE_MAX_TX_LAYER_NUM]
```

Estimated frequency offset

6.10.2.2 f_est_doppler_shift

```
float bblib_channel_estimation_5gnr_response::f_est_doppler_shift[CE_MAX_TX_LAYER_NUM]
```

Estimated doppler shift

6.10.2.3 f_est_sigma2

```
float bblib_channel_estimation_5gnr_response::f_est_sigma2[CE_MAX_DMRS_SYMBOL]
```

Estimated noise power

6.10.2.4 f_est_snr

```
float bblib_channel_estimation_5gnr_response::f_est_snr[CE_MAX_DMRS_SYMBOL][CE_MAX_TX_LAYER_←
NUM][CE_MAX_RX_ANT_NUM]
```

Estimated SNR after CE

6.10.2.5 f_power_bits

```
float bblib_channel_estimation_5gnr_response::f_power_bits[CE_MAX_RX_ANT_NUM][CE_MAX_TX_LAYE←
R_NUM][CE_MAX_DMRS_SYMBOL]
```

power bits of received DMRS signal

6.10.2.6 n_est_ta

```
int16_t bblib_channel_estimation_5gnr_response::n_est_ta[CE_MAX_TX_LAYER_NUM][CE_MAX_DMRS_SY←
MBOL]
```

Estimated TA value

6.10.2.7 p_ce_ls

```
int16_t* bblib_channel_estimation_5gnr_response::p_ce_ls[CE_MAX_TX_LAYER_NUM][CE_MAX_RX_ANT_N←
UM]
```

Data pointer points to nLayer∗nRx CE LS result buffer, format 16S13

6.10.2.8 p_ce_out

```
int16_t* bblib_channel_estimation_5gnr_response::p_ce_out[CE_MAX_RX_ANT_NUM][CE_MAX_TX_LAYER_←
NUM][CE_N_SYMB_PER_SLOT]
```

Data pointer points to nRx∗nLayer∗nSymbol CE output buffer, format 16S13

6.10.2.9 p_ce_ta_comp

```
int16_t* bblib_channel_estimation_5gnr_response::p_ce_ta_comp[CE_MAX_TX_LAYER_NUM]
```

Data pointer points to nLayer CE TA value buffer. Dummy, not used right now

6.10.2.10 p_irc_IpN

```
int16_t* bblib_channel_estimation_5gnr_response::p_irc_IpN[CE_MAX_RX_ANT_NUM][CE_MAX_DMRS_SYM←
BOL]
```

Data pointer points to nRx∗n_DMRS IpN buffer, format 16S13

6.11 bblib_compress_request Struct Reference

```
#include <phy_companding.h>
```

Data Fields

- int16_t ∗ data_in
- int32_t len

6.11.1 Detailed Description

Request structure containing pointer to data and its length.

6.11.2 Field Documentation

6.11.2.1 data_in

```
int16_t* bblib_compress_request::data_in
```

Pointer to data to compress.

6.11.2.2 len

```
int32_t bblib_compress_request::len
```

Length of input data.

6.12 bblib_compress_response Struct Reference

```
#include <phy_companding.h>
```

Data Fields

- int8_t * data_out
- int32_t len

6.12.1 Detailed Description

Response structure containing pointer to data and its length.

6.12.2 Field Documentation

6.12.2.1 data_out

```
int8_t* bblib_compress_response::data_out
```

Pointer to data after compression.

6.12.2.2 len

```
int32_t bblib_compress_response::len
```

Length of output data.

6.13 bblib_crc_request Struct Reference

```
#include <phy_crc.h>
```

Data Fields

- uint8_t * data
- uint32_t len

6.13.1 Detailed Description

Request structure containing pointer to input data sequence and its length (in bits).

6.13.2 Field Documentation

6.13.2.1 data

```
uint8_t* bblib_crc_request::data
```

Pointer to input data sequence for the CRC generator or CRC validate functions. Input data should be byte aligned in memory

6.13.2.2 len

```
uint32_t bblib_crc_request::len
```

The length of input data, in bits.

6.14 bblib_crc_response Struct Reference

```
#include <phy_crc.h>
```

Data Fields

- uint8_t * data
- uint32_t len
- uint32_t crc_value
- bool check_passed

6.14.1 Detailed Description

Response structure containing pointer to input data with appended CRC, new data length, CRC value and result of a CRC validate function.

6.14.2 Field Documentation

6.14.2.1 check_passed

```
bool bblib_crc_response::check_passed
```

Result of CRC Check Function, where true = passed, false = failed.

6.14.2.2 crc_value

```
uint32_t bblib_crc_response::crc_value
```

The calculated CRC value rounded to whole bytes,by appending zeros. Example: CRC11 bit binary value of 10010110001 (0x962) would be rounded to the nearest byte therefore becomes 010010110001 (0x4b1)

6.14.2.3 data

```
uint8_t* bblib_crc_response::data
```

Pointer to output data comprising of the input data with appended CRC value.

6.14.2.4 len

```
uint32_t bblib_crc_response::len
```

The length of output data including the CRC value, in bits.

6.15 bblib_decompress_request Struct Reference

```
#include <phy_companding.h>
```

Data Fields

- int8_t * data_in
- int32_t len

6.15.1 Detailed Description

Request structure containing pointer to data and its length.

6.15.2 Field Documentation

6.15.2.1 data_in

```
int8_t* bblib_decompress_request::data_in
```

Pointer to data to decompress.

6.15.2.2 len

```
int32_t bblib_decompress_request::len
```

Length of input data.

6.16 bblib_decompress_response Struct Reference

```
#include <phy_companding.h>
```

Data Fields

- int16_t * data_out
- int32_t len

6.16.1 Detailed Description

Response structure containing pointer to data and its length.

6.16.2 Field Documentation

6.16.2.1 data_out

```
int16_t* bblib_decompress_response::data_out
```

Pointer to data after decompression.

6.16.2.2 len

```
int32_t bblib_decompress_response::len
```

Length of output data.

6.17 bblib_deinterleave_request Struct Reference

```
#include <phy_deinterleave.h>
```

Data Fields

- modulation_type modType
- cp_type CPType
- ack_type bundType
- int32_t Rmux
- int32_t Cmux
- int32_t LLR
- int32_t LLRRI
- int32_t LLRACK
- int32_t LLRCQI
- int32_t nACK
- int32_t nRI
- int16_t * pIn

6.17.1 Detailed Description

Request structure providing the inputs and configuration to the deinterleaver.

Note

this module process de-interleave after the demodulation with LLR.

6.17.2 Field Documentation

6.17.2.1 bundType

```
ack_type bblib_deinterleave_request::bundType
```

Tdd bundling mode

6.17.2.2 Cmux

```
int32_t bblib_deinterleave_request::Cmux
```

Number of columns for channel de-interleaver.the value is 11 or 12

6.17.2.3 CPType

```
cp_type bblib_deinterleave_request::CPType
```

Cyclic prefix type

6.17.2.4 LLR

`int32_t bblib_deinterleave_request::LLR`

Number of pusch modulated symbols ∗ Qm

6.17.2.5 LLRACK

`int32_t bblib_deinterleave_request::LLRACK`

Number of bits for ACK output(harq ack symbols ∗ Qm, Qack value from from TS36.212 5.2.2.6)

6.17.2.6 LLRCQI

`int32_t bblib_deinterleave_request::LLRCQI`

Number of bits for CQI output.(cqi ack symbols ∗ Qm)

6.17.2.7 LLRRI

`int32_t bblib_deinterleave_request::LLRRI`

Number of bits for RI output(ri symbols ∗ Qm, Qri value from TS36.212 5.2.2.6)

6.17.2.8 modType

`modulation_type bblib_deinterleave_request::modType`

The modulation type

6.17.2.9 nACK

`int32_t bblib_deinterleave_request::nACK`

Number of final decoded ack/nack bits. O value from TS36.212 5.2.2.6 Not used with bblib_deinterleave_data_only

6.17.2.10 nRI

`int32_t bblib_deinterleave_request::nRI`

Number of final decoded ri bits. O value from TS36.212 5.2.2.6 Not used with bblib_deinterleave_data_only

6.17.2.11 pIn

```
int16_t* bblib_deinterleave_request::pIn
```

The input data, align with 512 bits. the input data Length is Rmux ∗ Cmux ∗ModType(bits), for 64QAM,onle 6 bits is used in a byte

6.17.2.12 Rmux

```
int32_t bblib_deinterleave_request::Rmux
```

Number of rows for channel de-interleaver. correspoding to sub-carriers

6.18 bblib_deinterleave_response Struct Reference

```
#include <phy_deinterleave.h>
```

Data Fields

- int8_t ∗ pOutData
- int8_t ∗ pOutACK
- int8_t ∗ pOutRI
- int8_t ∗ pOutCQI

6.18.1 Detailed Description

Response structure bundling the outputs from the deinterleaver.

6.18.2 Field Documentation

6.18.2.1 pOutACK

```
int8_t* bblib_deinterleave_response::pOutACK
```

Soft decisions for coded channel ACK bits.(Qack ∗ Qm,value from ts36.212 5.2.2.6) Not used with bblib_deinterleave_data_only

6.18.2.2 pOutCQI

```
int8_t* bblib_deinterleave_response::pOutCQI
```

Soft decisions for coded channel CQI bits(value from ts36.212 5.2.2.6) Not used with bblib_deinterleave_data_only

6.18.2.3 pOutData

`int8_t* bblib_deinterleave_response::pOutData`

Soft decisions for multiplexed coded transport block

6.18.2.4 pOutRI

`int8_t* bblib_deinterleave_response::pOutRI`

Soft decisions for coded channel RI bits(Qri ∗ Qm,value from ts36.212 5.2.2.6) Not used with bblib_deinterleave_data_only

6.19 bblib_deinterleave_ul_request Struct Reference

```
#include <phy_rate_match.h>
```

Data Fields

- uint8_t ∗ pharqbuffer
- int32_t ncb
- enum circular_buffer_format circ_buffer

6.19.1 Detailed Description

Request structure for parameters in API of deinterleaver (rate dematching) for LTE.

Note

pharqbuffer and pinteleavebuffer need to be aligned with 256 bits

6.19.2 Field Documentation

6.19.2.1 circ_buffer

`enum circular_buffer_format bblib_deinterleave_ul_request::circ_buffer`

Defines input of sub-block deinterleaver format: circular buffer without padding or full circular buffer with dummy padding bits.

6.19.2.2 ncb

`int32_t bblib_deinterleave_ul_request::ncb`

cyclic buffer length

6.19.2.3 pharqbuffer

`uint8_t* bblib_deinterleave_ul_request::pharqbuffer`

output of HARQ combine, and input of sub-block deinterleaver

6.20 bblib_deinterleave_ul_response Struct Reference

`#include <phy_rate_match.h>`

Data Fields

- uint8_t * pinteleavebuffer

6.20.1 Detailed Description

Response structure for parameters in API of deinterleaver (rate dematching) for LTE.

6.20.2 Field Documentation

6.20.2.1 pinteleavebuffer

`uint8_t* bblib_deinterleave_ul_response::pinteleavebuffer`

output of sub-block deinterleaver, and input of turbo decoder adapter

6.21 bblib_demod_pucch_request Struct Reference

`#include <phy_demodulation_pucch.h>`

Data Fields

- enum bblib_demod_pucch_format format
- int16_t numAnt
- int16_t Qm
- int16_t RfTuningSymb
- int16_t RepNumInst
- int16_t MaxRepNum
- int16_t catmEnable
- int16_t shortenFlag
- float noisePwr
- void * dEstAck
- void * chEst
- void * dEstCqi
- float noisePwrAvg
- float chanPow
- void * bDetSoft

6.21.1 Detailed Description

Structure defining the input interface for the kernel.

6.21.2 Field Documentation

6.21.2.1 bDetSoft

```
void* bblib_demod_pucch_request::bDetSoft
```

Cat-M soft decision state for Repetition for PUCCH. Buffers must be 64-byte aligned and a single IQ input in either floating point or in fixed point Q16s11 format.

6.21.2.2 catmEnable

```
int16_t bblib_demod_pucch_request::catmEnable
```

Cat-M enable flag.

6.21.2.3 chanPow

```
float bblib_demod_pucch_request::chanPow
```

Cat-M channel power state for Repetition for PUCCH.

6.21.2.4 chEst

```
void* bblib_demod_pucch_request::chEst
```

Pointer to input buffer containing the averaged channel estimate for each slot and each RX antenna. Buffers must be 64-byte aligned and must be of size num_slot∗num_antenna. Numbers are stored as interleaved IQ values in either floating point or in fixed point Q16s11 format.

6.21.2.5 dEstAck

```
void* bblib_demod_pucch_request::dEstAck
```

Pointer to input buffer containing the ack/nack correlation for each slot and each RX antenna. Buffers must be 64-byte aligned and must be of size num_slot∗num_antenna. Numbers are stored as interleaved IQ values in either floating point or in fixed point Q16s11 format.

6.21.2.6 dEstCqi

```
void* bblib_demod_pucch_request::dEstCqi
```

Pointer to input buffer containing the cqi correlation for each slot and each RX antenna. Buffers must be 64-byte aligned and must be of size num_slot∗num_antenna∗num_cqi_per_subframe. Numbers are stored as interleaved IQ values in either floating point or in fixed point Q16s11 format.

6.21.2.7 format

```
enum bblib_demod_pucch_format bblib_demod_pucch_request::format
```

PUCCH format type.

6.21.2.8 MaxRepNum

```
int16_t bblib_demod_pucch_request::MaxRepNum
```

Maximum Cat-M Repetition.

6.21.2.9 noisePwr

```
float bblib_demod_pucch_request::noisePwr
```

Noise variance seen.

6.21.2.10 noisePwrAvg

```
float bblib_demod_pucch_request::noisePwrAvg
```

Cat-M noise power state for Repetition for PUCCH.

6.21.2.11 numAnt

`int16_t bblib_demod_pucch_request::numAnt`

Number of RX antenna in the system. Valid values - 1, 2, 4 and 8

6.21.2.12 Qm

`int16_t bblib_demod_pucch_request::Qm`

qam factor of PUCCH. Determines if using BPSK or QPSK for PUCCH format 2, 2a and 2b demodulation. Valid values - 1 (BPSK), 2 (QPSK)

6.21.2.13 RepNumInst

`int16_t bblib_demod_pucch_request::RepNumInst`

Cat-M Repetition Number. Used when Cat-M enable flag is set to combine the soft decision state, channel power and noise power calculated by the algorithm with the Cat-M input values from the request structure to accumulate the results. Value must be less than MaxRepNum to allow values to be combined. Used for PUCCH format 1, 1a and 1b demodulation.

6.21.2.14 RfTuningSymb

`int16_t bblib_demod_pucch_request::RfTuningSymb`

Rf tuning symbol for PUCCH.

6.21.2.15 shortenFlag

`int16_t bblib_demod_pucch_request::shortenFlag`

Flag for shortened PUCCH.

6.22 bblib_demod_pucch_response Struct Reference

```
#include <phy_demodulation_pucch.h>
```

Data Fields

- void * bDetSoftOut

6.22.1 Detailed Description

Structure defining the output interface of the kernel.

6.22.2 Field Documentation

6.22.2.1 bDetSoftOut

```
void* bblib_demod_pucch_response::bDetSoftOut
```

Pointer to output buffer containing the output soft decisions after demapping. Buffers must be 64-byte aligned. Buffer size should be single IQ input in either floating point or in fixed point Q16s11 format for format 1s. For format 2s, buffer size should be num_cqi_per_subframe+1 IQ inputs in either floating point or in fixed point Q16s11 format.

6.23 bblib_demodulation_request Struct Reference

```
#include <phy_demodulation.h>
```

Data Fields

- int16_t * input0
- int16_t * input1
- int16_t * input2
- uint32_t shift
- enum bblib_modulation_order mod_order
- uint32_t num_carriers
- uint8_t threshold
- int8_t const2
- int8_t const4
- int8_t const8
- enum bblib_demod_llr_polarity llr_polarity

6.23.1 Detailed Description

Request structure for demodulation function.

Note

Demodulation has three inputs as there are three output from the IDFT, as it is called three times processing 4 symbols in parallel.
QPSK doesn't use const parameters, 16QAM uses const2, 64QAM uses const2 and const4 and 256QAM uses all const parameters.
256QAM doesn't use shift parameter as it assumes iDFT output is not shifted. The reason for that is this is a legacy parameter that is set to 0 in the refPHY, but is required in unittests for lower orders of the modulation.

6.23.2 Field Documentation

6.23.2.1 const2

```
int8_t bblib_demodulation_request::const2
```

Used to adjust output based on noise and channel power.

6.23.2.2 const4

```
int8_t bblib_demodulation_request::const4
```

Used to adjust output based on noise and channel power.

6.23.2.3 const8

```
int8_t bblib_demodulation_request::const8
```

Used to adjust output based on noise and channel power.

6.23.2.4 input0

```
int16_t* bblib_demodulation_request::input0
```

Input symbol data align with 64 bytes.

6.23.2.5 input1

```
int16_t* bblib_demodulation_request::input1
```

Input symbol data align with 64 bytes.

6.23.2.6 input2

```
int16_t* bblib_demodulation_request::input2
```

Input symbol data align with 64 bytes.

6.23.2.7 llr_polarity

```
enum bblib_demod_llr_polarity bblib_demodulation_request::llr_polarity
```

Used to adjust the sign of the llr outputs

6.23.2.8 mod_order

```
enum bblib_modulation_order bblib_demodulation_request::mod_order
```

Supported values are 2 (QPSK), 4 (16QAM), 6 (64QAM), 8 (256QAM).

6.23.2.9 num_carriers

```
uint32_t bblib_demodulation_request::num_carriers
```

The number of carriers. Must be divisible by 4.

6.23.2.10 shift

```
uint32_t bblib_demodulation_request::shift
```

Scale the IDFT output.

6.23.2.11 threshold

```
uint8_t bblib_demodulation_request::threshold
```

Limit data to within the range set by threshold.

6.24 bblib_demodulation_response Struct Reference

```
#include <phy_demodulation.h>
```

Data Fields

- int16_t ∗ output
- int32_t num_output

6.24.1 Detailed Description

Response structure for demodulation function.

6.24.2 Field Documentation

6.24.2.1 num_output

```
int32_t bblib_demodulation_response::num_output
```

Number of output soft-bits.

6.24.2.2 output

`int16_t* bblib_demodulation_response::output`

Pointer to the output soft-bit after demodulation. Buffer has to be 64 bytes aligned.

6.25 bblib_descramble_request Struct Reference

`#include <phy_scramble.h>`

Data Fields

- int16_t * data_in
- bool is_discontiguous
- uint32_t len
- uint32_t c_init

6.25.1 Detailed Description

Request structure for descrambler.

6.25.2 Field Documentation

6.25.2.1 c_init

`uint32_t bblib_descramble_request::c_init`

The value of init.

6.25.2.2 data_in

`int16_t* bblib_descramble_request::data_in`

The input data before descramble, must be aligned to 64bits. This data is 8 bit LLR.

6.25.2.3 is_discontiguous

`bool bblib_descramble_request::is_discontiguous`

Selected on the basis of different input data structures, 0/1, 0 is contiguous memory, 1 is discontiguous memory for both input and output. Discontiguous memory uses byte locations 0..5 only in each 8 bytes

6.25.2.4 len

```
uint32_t bblib_descramble_request::len
```

The length of input data in number of 8 bit LLR

6.26 bblib_descramble_response Struct Reference

```
#include <phy_scramble.h>
```

Data Fields

- int16_t * data_out
- int32_t len

6.26.1 Detailed Description

Response structure for scrambler.

6.26.2 Field Documentation

6.26.2.1 data_out

```
int16_t* bblib_descramble_response::data_out
```

The output data after descramble, must be aligned to 64bits. This data is 8 bit LLR.

6.26.2.2 len

```
int32_t bblib_descramble_response::len
```

The length of output data in number of 8 bit LLR

6.27 bblib_despread_compensate_pucch_f1_request Struct Reference

```
#include <phy_pucch_5gnr.h>
```

Data Fields

- int16_t n_shift_right
- uint16_t n_sym_num
- int16_t n_freq_hopping
- int16_t ∗ p_data [BBLIB_PUCCH_SYMB_PER_SLOT]
- int16_t ∗ p_seq [BBLIB_PUCCH_SYMB_PER_SLOT]

6.27.1 Detailed Description

Request structure for PUCCH format 1 despreading and compensation.

6.27.2 Field Documentation

6.27.2.1 n_freq_hopping

`int16_t bblib_despread_compensate_pucch_f1_request::n_freq_hopping`

enable or disable intra slot freq hopping

6.27.2.2 n_shift_right

`int16_t bblib_despread_compensate_pucch_f1_request::n_shift_right`

shift right bits after multiplication

6.27.2.3 n_sym_num

`uint16_t bblib_despread_compensate_pucch_f1_request::n_sym_num`

Number of symbols in PUCCH format 1

6.27.2.4 p_data

`int16_t* bblib_despread_compensate_pucch_f1_request::p_data[BBLIB_PUCCH_SYMB_PER_SLOT]`

Pointer to received symbol, Q16S14 for the I/Q data

6.27.2.5 p_seq

`int16_t* bblib_despread_compensate_pucch_f1_request::p_seq[BBLIB_PUCCH_SYMB_PER_SLOT]`

Pointer to reference symbol, with low PAPR sequence∗Omega, Q16S14 for the I/Q data

6.28 bblib_despread_compensate_pucch_f1_response Struct Reference

```
#include <phy_pucch_5gnr.h>
```

Data Fields

- int16_t * p_constell_out

6.28.1 Detailed Description

Response structure for PUCCH format 1 despreading and compensation.

6.28.2 Field Documentation

6.28.2.1 p_constell_out

```
int16_t* bblib_despread_compensate_pucch_f1_response::p_constell_out
```

Pointer to output constellation, Q16S14 for the I/Q data

6.29 bblib_dft_burst_request Struct Reference

```
#include <phy_dft_idft.h>
```

Data Fields

- void ** data_in
- int16_t dft_points
- dft_idft_flag_type dft_flag
- int16_t num_input_buffers

6.29.1 Detailed Description

Request structure for the dft function using burst mode.

6.29.2 Field Documentation

6.29.2.1 data_in

```
void** bblib_dft_burst_request::data_in
```

Pointer to array of pointers to the input buffers for dft. Each input buffer contains the input symbols for a single dft input. Each symbol is organized as two 16-bit fixed point integers, which represent the real part and the imaginary part respectively. Array size should equal num_input_buffers.

6.29.2.2 dft_flag

```
dft_idft_flag_type bblib_dft_burst_request::dft_flag
```

Should be set to BBLIB_DFT_TYPE.

6.29.2.3 dft_points

```
int16_t bblib_dft_burst_request::dft_points
```

The points which need to be transfermed for dft, the valid sample points are 12 to 1200, and is limited to products of the integers 2, 3, 5 and multiple of 12.

6.29.2.4 num_input_buffers

```
int16_t bblib_dft_burst_request::num_input_buffers
```

Number of input buffers passed to the dft. Valid values are 1 and 4. In case single symbol DFT/IDFT value is set to 1. In case four symbols DFT/IDFT value is 4.

6.30 bblib_dft_burst_response Struct Reference

```
#include <phy_dft_idft.h>
```

Data Fields

- void ** data_out
- int16_t num_output_buffers

6.30.1 Detailed Description

\ Response structure containing the output dft data in burst form.

6.30.2 Field Documentation

6.30.2.1 data_out

`void** bblib_dft_burst_response::data_out`

Pointer to array of pointers to the output buffers for dft. Each output buffer contains the output symbols for a single dft output. Each symbol is organized as two 16-bit fixed point integers, which represent the real part and the imaginary part respectively. Array size should equal num_output_buffers.

6.30.2.2 num_output_buffers

`int16_t bblib_dft_burst_response::num_output_buffers`

Number of output buffers passed to the dft. Valid values are 1 and 4. Must match num_input_buffer.

6.31 bblib_dft_request Struct Reference

`#include <phy_dft_idft.h>`

Data Fields

- void * data_in
- void ** data_in_4way
- int16_t dft_idft_points
- dft_idft_flag_type dft_idft_flag
- int16_t num_input_buffers

6.31.1 Detailed Description

Request structure for dft/idft function.

6.31.2 Field Documentation

6.31.2.1 data_in

`void* bblib_dft_request::data_in`

Input buffer for dft/idft. data_in is uesed for interleaved 4 way input and non-interleaved 1 way input Each symbol is organized as two 16-bit fixed-pointer integers, which represent the real part and the imaginary part respectively.

Symb0–Symb1–Symb2–Symb3 : Point 0 Symb0–Symb1–Symb2–Symb3 : Point 1 ... ... Symb0–Symb1–Symb2–Symb3 : Point N

6.31.2.2 data_in_4way

`void** bblib_dft_request::data_in_4way`

Input buffer for dft/idft.data_in_4way is used only for non-interleaved 4 way input Each symbol is organized as two 16-bit fixed-pointer integers, which represent the real part and the imaginary part respectively.

Symb0 : Point 0 ... Point N; Symb1 : Point 0 ... Point N; Symb2 : Point 0 ... Point N; Symb3 : Point 0 ... Point N;

6.31.2.3 dft_idft_flag

`dft_idft_flag_type bblib_dft_request::dft_idft_flag`

Table 6.1 DFT/IDFT TYPE DEFINE

Number	DFT/IDFT type
0	normal DFT transformation
1	normal IDFT transformation

6.31.2.4 dft_idft_points

```
int16_t bblib_dft_request::dft_idft_points
```

The points which need to be transfermed for dft/idft, the valid sample points are 12 to 1200, and is limited to products of the integers 2, 3, 5 and multiple of 12.

6.31.2.5 num_input_buffers

```
int16_t bblib_dft_request::num_input_buffers
```

Number of input buffers passed to the dft. Valid values are 1 and 4. In case single symbol DFT/IDFT value is set to 1. In case four symbols DFT/IDFT value is 4.

6.32 bblib_dft_response Struct Reference

```
#include <phy_dft_idft.h>
```

Data Fields

- void * data_out
- void ** data_out_4way
- uint8_t scale_out
- uint8_t scale_out_list [DFT_4_WAY]

6.32.1 Detailed Description

Response structure containing output data.

6.32.2 Field Documentation

6.32.2.1 data_out

```
void* bblib_dft_response::data_out
```

Output buffer for dft/idft.data_out is used for interleaved 4 way input and non-interleaved 1 way output Each symbol is organized as two 16-bit fixed-pointer integers, which represent the real part and the imaginary part respectively.

Symb0–Symb1–Symb2–Symb3 : Point 0 Symb0–Symb1–Symb2–Symb3 : Point 1 ... ... Symb0–Symb1–Symb2–Symb3 : Point N

6.32.2.2 data_out_4way

```
void** bblib_dft_response::data_out_4way
```

Output buffer for dft/idft.data_out_4way is used only for non-interleaved output Each symbol is organized as two 16-bit fixed-pointer integers, which represent the real part and the imaginary part respectively.

Symb0 : Point 0 ... Point N; Symb1 : Point 0 ... Point N; Symb2 : Point 0 ... Point N; Symb3 : Point 0 ... Point N;

6.32.2.3 scale_out

```
uint8_t bblib_dft_response::scale_out
```

Output scaling of dft computing. For example, if input fxp is 16sX, then output will be 16s(X-scale_out)

6.32.2.4 scale_out_list

```
uint8_t bblib_dft_response::scale_out_list[DFT_4_WAY]
```

Output scaling of dft computing, has individula value for each way. For example, if input fxp is 16sX, then output will be 16s(X-scale_out[m]) for m'th may

6.33 bblib_dftcodebook_weightgen_request Struct Reference

```
#include <phy_dftcodebook_weightgen.h>
```

Data Fields

- int16_t ∗ pChState [BBLIB_DFTCODEBOOK_MAX_RX_ANT_NUM][BBLIB_DFTCODEBOOK_MAX_RX_ANT_NUM]
- int16_t nLayer
- int16_t nStream
- int16_t nAnt
- int16_t nAntVertical
- int16_t nAntHorizontal
- int16_t nPolarization
- int16_t nStart
- int16_t nLen
- int16_t nGranularity

6.33.1 Detailed Description

Request struct of dft codebook based weight matrix generation.

6.33.2 Field Documentation

6.33.2.1 nAnt

`int16_t bblib_dftcodebook_weightgen_request::nAnt`

Number of Rx Antennas

6.33.2.2 nAntHorizontal

`int16_t bblib_dftcodebook_weightgen_request::nAntHorizontal`

Number of Rx antennas Horizontal elements

6.33.2.3 nAntVertical

`int16_t bblib_dftcodebook_weightgen_request::nAntVertical`

Number of Rx antennas vertical elements

6.33.2.4 nGranularity

`int16_t bblib_dftcodebook_weightgen_request::nGranularity`

RB selection granularity of the beam forming weight matrix gen calculation

6.33.2.5 nLayer

`int16_t bblib_dftcodebook_weightgen_request::nLayer`

Number of Tx Layers

6.33.2.6 nLen

`int16_t bblib_dftcodebook_weightgen_request::nLen`

Number of input RB

6.33.2.7 nPolarization

```
int16_t bblib_dftcodebook_weightgen_request::nPolarization
```

Rx Antenna polarization indication, 1 for no polarized antenna, 2 for dual-polarized antenna

6.33.2.8 nStart

```
int16_t bblib_dftcodebook_weightgen_request::nStart
```

Start point

6.33.2.9 nStream

```
int16_t bblib_dftcodebook_weightgen_request::nStream
```

Number of the aimed output streams - valid maximum values 32 or 64

6.33.2.10 pChState

```
int16_t* bblib_dftcodebook_weightgen_request::pChState[BBLIB_DFTCODEBOOK_MAX_RX_ANT_NUM][BBLIB_DFTCODEBOOK_MAX
```

Data pointer points to channel

6.34 bblib_dftcodebook_weightgen_response Struct Reference

```
#include <phy_dftcodebook_weightgen.h>
```

Data Fields

- int16_t * pWeightMatrixBufs [BBLIB_DFTCODEBOOK_MAX_STREAM_NUM]

6.34.1 Detailed Description

Response struct of dft codebook based weight matrix generation.

6.34.2 Field Documentation

6.34.2.1 pWeightMatrixBufs

```
int16_t* bblib_dftcodebook_weightgen_response::pWeightMatrixBufs[BBLIB_DFTCODEBOOK_MAX_STREAM_NUM]
```

Data pointer points to weight matrix buffer

6.35 bblib_eigen_beamforming_request Struct Reference

```
#include <phy_eigen_beamforming.h>
```

Data Fields

- complex_float * m_chan_est
- float m_sigma_sq
- size_t m_num_users
- size_t m_num_antennas
- size_t m_num_matrices

6.35.1 Detailed Description

Request structure for eigen beamforming.

6.35.2 Field Documentation

6.35.2.1 m_chan_est

```
complex_float* bblib_eigen_beamforming_request::m_chan_est
```

Input matrix of channel estimate Max size = max_antennas * max_num_users * max_num_matrices (16 by 32 by 16) 64 byte aligned

6.35.2.2 m_num_antennas

```
size_t bblib_eigen_beamforming_request::m_num_antennas
```

The number of columns in the channel estimates. Fixed for a given base-station. Max 32

6.35.2.3 m_num_matrices

```
size_t bblib_eigen_beamforming_request::m_num_matrices
```

The number of matrices to work on. Multiples of 8 or 16 will be executed in SIMD. Max 16

6.35.2.4 m_num_users

```
size_t bblib_eigen_beamforming_request::m_num_users
```

The number of rows in the channel estimates. Can vary across TTIs. Max 16

6.35.2.5 m_sigma_sq

```
float bblib_eigen_beamforming_request::m_sigma_sq
```

Sigma squared

6.36 bblib_eigen_beamforming_response Struct Reference

```
#include <phy_eigen_beamforming.h>
```

Data Fields

- complex_float * m_precoding
- uint32_t * m_num_layers
- size_t m_num_antennas
- size_t m_max_num_layers
- size_t m_num_matrices

6.36.1 Detailed Description

The response structure for DL Eigen beamforming. The structures inside the response should be pre-allocated to the expected sizes. Note that the response can return multiple matrix objects. They will all have the same number of rows, but can have different numbers of columns. Each matrix has storage of size [numAntennas * maxNumLayers], but only the first m_numLayers[i] columns of matrix 'i' will contain valid data.

6.36.2 Field Documentation

6.36.2.1 m_max_num_layers

```
size_t bblib_eigen_beamforming_response::m_max_num_layers
```

The number of columns in the precoding matrix.

6.36.2.2 m_num_antennas

```
size_t bblib_eigen_beamforming_response::m_num_antennas
```

The number of rows in the precoding matrix.

6.36.2.3 m_num_layers

`uint32_t* bblib_eigen_beamforming_response::m_num_layers`

The number of layers in each matrix. Size = num_matrices, 64 byte aligned

6.36.2.4 m_num_matrices

`size_t bblib_eigen_beamforming_response::m_num_matrices`

The number of matrices to work on. Multiples of 8 will be executed in SIMD.

6.36.2.5 m_precoding

`complex_float* bblib_eigen_beamforming_response::m_precoding`

The return matrix objects, size k_maxNumAntennas * k_maxLayers * k_maxMatrices 64 byte aligned

6.37 bblib_fd_correlation_request Struct Reference

`#include <phy_fd_correlation.h>`

Data Fields

- complex_int16_t * in0
- complex_int16_t * in1
- uint16_t len

6.37.1 Detailed Description

Request structure for frequency domain correlation function.

Note

fd_correlation has three inputs - Frequency domain input sequence 0 (this input is conjugated internally) Frequency domain input sequence 1 Length of the input/output sequences in complex symbols

the data pattern of the in0 and in1 should be in0 = [real(a0), image(a0), real(a1), image(a1), ...]; in1 = [real(b0), image(b0), real(b1), image(b1), ...];

6.37.2 Field Documentation

6.37.2.1 in0

`complex_int16_t* bblib_fd_correlation_request::in0`

Frequency domain data aligned with 64 bytes.

6.37.2.2 in1

`complex_int16_t* bblib_fd_correlation_request::in1`

Frequency domain data aligned with 64 bytes.

6.37.2.3 len

`uint16_t bblib_fd_correlation_request::len`

Input/output symbol length in complex symbols

6.38 bblib_fd_correlation_response Struct Reference

`#include <phy_fd_correlation.h>`

Data Fields

- complex_int16_t * out

6.38.1 Detailed Description

Response structure for frequency domain correlation function.

Note

the data pattern of out should be out = [real(c0), image(c0), real(c1), image(c1), ...];

6.38.2 Field Documentation

6.38.2.1 out

`complex_int16_t* bblib_fd_correlation_response::out`

Pointer to complex output symbols, 64 bytes aligned.

6.39 bblib_fec_enc_byte_concat_soft_request Struct Reference

```
#include <phy_fec_enc_byte_concat_soft.h>
```

Data Fields

- uint32_t * E
- int32_t hLen
- int32_t tBitTotal
- uint8_t ** pIn

6.39.1 Detailed Description

Request structure for bblib_fec_enc_byte_concat_soft.

6.39.2 Field Documentation

6.39.2.1 E

```
uint32_t* bblib_fec_enc_byte_concat_soft_request::E
```

The pointer to length of each code block.

6.39.2.2 hLen

```
int32_t bblib_fec_enc_byte_concat_soft_request::hLen
```

The number of code blocks.

6.39.2.3 pIn

```
uint8_t** bblib_fec_enc_byte_concat_soft_request::pIn
```

The pointer to input code blocks.

6.39.2.4 tBitTotal

```
int32_t bblib_fec_enc_byte_concat_soft_request::tBitTotal
```

The sum of lengths of all code blocks rate matching output in bits.

6.40 bblib_fec_enc_byte_concat_soft_response Struct Reference

`#include <phy_fec_enc_byte_concat_soft.h>`

Data Fields

- uint8_t * pOut

6.40.1 Detailed Description

Response structure for bblib_fec_enc_byte_concat_soft.

6.40.2 Field Documentation

6.40.2.1 pOut

`uint8_t* bblib_fec_enc_byte_concat_soft_response::pOut`

The pointer to output buffer after concatenation.

6.41 bblib_fft_request Struct Reference

`#include <phy_fft_ifft.h>`

Data Fields

- complex_int16_t * in
- uint32_t size

6.41.1 Detailed Description

Request structure for the FFT function contaning pointer to the input data and the size (number of points) of the FFT.

Note

The input buffer has to be aligned to 64 bytes.

6.41.2 Field Documentation

6.41.2.1 in

`complex_int16_t* bblib_fft_request::in`

Pointer to the buffer with the input data. The size of the buffer has to be at least size $*$ 4 bytes. The fixed point data format has to be Q16s15. The input data has to be in the natural order. The reordering is done inside the function.

6.41.2.2 size

`uint32_t bblib_fft_request::size`

Number of complex samples in the buffer/number of points in FFT. It coresponds to N in the DFT formula.

6.42 bblib_fft_response Struct Reference

`#include <phy_fft_ifft.h>`

Data Fields

- complex_int16_t $*$ out
- uint32_t size

6.42.1 Detailed Description

Response structure for the FFT function contaning pointer to the output data and the size (number of points) of the FFT.

Note

The output buffer has to be aligned to 64 bytes.

6.42.2 Field Documentation

6.42.2.1 out

`complex_int16_t* bblib_fft_response::out`

Pointer to the buffer with the output data. The size of the buffer has to be at least size $*$ 4 bytes. The output format is Q16s15.

6.42.2.2 size

```
uint32_t bblib_fft_response::size
```

Number of complex samples in the buffer

6.43 bblib_harq_combine_ul_request Struct Reference

```
#include <phy_rate_match.h>
```

Data Fields

- uint8_t * pdmout
- int32_t k0withoutnull
- int32_t ncb
- int32_t e
- int32_t isretx

6.43.1 Detailed Description

Request structure for parameters in API of HARQ for LTE.

Note

pdmout and pharqbuffer need to be aligned with 256 bits

6.43.2 Field Documentation

6.43.2.1 e

```
int32_t bblib_harq_combine_ul_request::e
```

HARQ combine input length in bytes

6.43.2.2 isretx

```
int32_t bblib_harq_combine_ul_request::isretx
```

flag of retransmission, 0: no retransmission, 1: retransmission

6.43.2.3 k0withoutnull

```
int32_t bblib_harq_combine_ul_request::k0withoutnull
```

K0 without NULL based on RV, position of this input HARQ sequence in ring buffer

6.43.2.4 ncb

```
int32_t bblib_harq_combine_ul_request::ncb
```

cyclic buffer length

6.43.2.5 pdmout

```
uint8_t* bblib_harq_combine_ul_request::pdmout
```

demodulation output, and input of HARQ

6.44 bblib_harq_combine_ul_response Struct Reference

```
#include <phy_rate_match.h>
```

Data Fields

- uint8_t * pharqbuffer

6.44.1 Detailed Description

Response structure for parameters in API of HARQ for LTE.

6.44.2 Field Documentation

6.44.2.1 pharqbuffer

```
uint8_t* bblib_harq_combine_ul_response::pharqbuffer
```

output of HARQ combine, and input of sub-block deinterleaver

6.45 bblib_idft_burst_request Struct Reference

```
#include <phy_dft_idft.h>
```

Data Fields

- void ** data_in
- int16_t idft_points
- dft_idft_flag_type idft_flag
- int16_t num_input_buffers

6.45.1 Detailed Description

Request structure for the idft function using burst mode.

6.45.2 Field Documentation

6.45.2.1 data_in

`void** bblib_idft_burst_request::data_in`

Pointer to array of pointers to the input buffers for idft. Each input buffer contains the input symbols for a single idft input. Each symbol is organized as two 16-bit fixed point integers, which represent the real part and the imaginary part respectively. Array size should equal num_input_buffers.

6.45.2.2 idft_flag

`dft_idft_flag_type bblib_idft_burst_request::idft_flag`

Should be set to BBLIB_IDFT_TYPE.

6.45.2.3 idft_points

`int16_t bblib_idft_burst_request::idft_points`

The points which need to be transfermed for idft, the valid sample points are 12 to 1200, and is limited to products of the integers 2, 3, 5 and multiple of 12.

6.45.2.4 num_input_buffers

`int16_t bblib_idft_burst_request::num_input_buffers`

Number of input buffers passed to the idft. Valid values are 1 and 4. In case single symbol DFT/IDFT value is set to 1. In case four symbols DFT/IDFT value is 4.

6.46 bblib_idft_burst_response Struct Reference

`#include <phy_dft_idft.h>`

Data Fields

- void ∗∗ data_out
- int16_t num_output_buffers

6.46.1 Detailed Description

\ Response structure containing the output idft data in burst form.

6.46.2 Field Documentation

6.46.2.1 data_out

```
void** bblib_idft_burst_response::data_out
```

Pointer to array of pointers to the output buffers for idft. Each output buffer contains the output symbols for a single idft output. Each symbol is organized as two 16-bit fixed point integers, which represent the real part and the imaginary part respectively. Array size should equal num_output_buffers.

6.46.2.2 num_output_buffers

```
int16_t bblib_idft_burst_response::num_output_buffers
```

Number of output buffers passed to the idft. Valid values are 1 and 4. Must match num_input_buffer.

6.47 bblib_irc_rnn_calculation_5gnr_request Struct Reference

```
#include <phy_irc_rnn_calculation_5gnr.h>
```

Data Fields

- int16_t n_dmrs_config_type
- int16_t n_rxant
- int16_t n_layer
- int16_t n_start_prb
- int16_t n_prb
- int16_t n_dmrs_symb
- int16_t n_data_symb
- int16_t ∗ p_irc_IpN [CE_MAX_RX_ANT_NUM][CE_MAX_DMRS_SYMBOL]

6.47.1 Detailed Description

Structure defines the irc_rnn_calculation function request interface in 5GNR.

6.47.2 Field Documentation

6.47.2.1 n_data_symb

`int16_t bblib_irc_rnn_calculation_5gnr_request::n_data_symb`

Number of sheduled data symbols

6.47.2.2 n_dmrs_config_type

`int16_t bblib_irc_rnn_calculation_5gnr_request::n_dmrs_config_type`

DMRS configuration type: 1 or 2

6.47.2.3 n_dmrs_symb

`int16_t bblib_irc_rnn_calculation_5gnr_request::n_dmrs_symb`

Number of DMRS symbols

6.47.2.4 n_layer

`int16_t bblib_irc_rnn_calculation_5gnr_request::n_layer`

Number of Layers

6.47.2.5 n_prb

`int16_t bblib_irc_rnn_calculation_5gnr_request::n_prb`

Number of PRBs

6.47.2.6 n_rxant

`int16_t bblib_irc_rnn_calculation_5gnr_request::n_rxant`

Number of Rx antennas

6.47.2.7 n_start_prb

`int16_t bblib_irc_rnn_calculation_5gnr_request::n_start_prb`

Starting PRB number

6.47.2.8 p_irc_IpN

```
int16_t* bblib_irc_rnn_calculation_5gnr_request::p_irc_IpN[CE_MAX_RX_ANT_NUM][CE_MAX_DMRS_SYM↩
BOL]
```

Data pointer points to nRx∗n_DMRS IpN buffer

6.48 bblib_irc_rnn_calculation_5gnr_response Struct Reference

```
#include <phy_irc_rnn_calculation_5gnr.h>
```

Data Fields

- int32_t ∗ p_irc_Rnn_dmrs [CE_MAX_DMRS_SYMBOL][CE_MAX_RX_ANT_NUM][CE_MAX_RX_ANT_N↩ UM]
- int32_t ∗ p_irc_Rnn_out_re [CE_MAX_DMRS_SYMBOL][CE_MAX_RX_ANT_NUM][CE_MAX_RX_ANT_↩ NUM]
- int32_t ∗ p_irc_Rnn_out_im [CE_MAX_DMRS_SYMBOL][CE_MAX_RX_ANT_NUM][CE_MAX_RX_ANT_↩ NUM]

6.48.1 Detailed Description

Structure defines the irc_rnn_calculation response interface in 5GNR.

6.48.2 Field Documentation

6.48.2.1 p_irc_Rnn_dmrs

```
int32_t* bblib_irc_rnn_calculation_5gnr_response::p_irc_Rnn_dmrs[CE_MAX_DMRS_SYMBOL][CE_MAX_R↩
X_ANT_NUM][CE_MAX_RX_ANT_NUM]
```

Data pointer points to nRx∗nRx∗n_DMRS Rnn buffer for dmrs

6.48.2.2 p_irc_Rnn_out_im

```
int32_t* bblib_irc_rnn_calculation_5gnr_response::p_irc_Rnn_out_im[CE_MAX_DMRS_SYMBOL][CE_MAX↩
_RX_ANT_NUM][CE_MAX_RX_ANT_NUM]
```

Data pointer points to nRx∗nRx∗n_DMRS imag part of Rnn buffer

6.48.2.3 p_irc_Rnn_out_re

```
int32_t* bblib_irc_rnn_calculation_5gnr_response::p_irc_Rnn_out_re[CE_MAX_DMRS_SYMBOL][CE_MAX↩
_RX_ANT_NUM][CE_MAX_RX_ANT_NUM]
```

Data pointer points to nRx∗nRx∗n_DMRS real part of Rnn buffer

6.49 bblib_layer_llr_demap_request Struct Reference

```
#include <phy_pusch_symbol_processing_5gnr.h>
```

Data Fields

- uint16_t nLayer
- uint16_t nLen
- uint8_t nLlrFxpPoints
- enum bblib_modulation_order eModOrder
- int16_t ∗ pEqualOut [BBLIB_MAX_TX_LAYER_NUM]
- float ∗ pMmseGain [BBLIB_MAX_TX_LAYER_NUM]
- float ∗ pPostSINR [BBLIB_MAX_TX_LAYER_NUM]

6.49.1 Detailed Description

Request struct of layer demapping and LLR demapping processing.

6.49.2 Field Documentation

6.49.2.1 eModOrder

```
enum bblib_modulation_order bblib_layer_llr_demap_request::eModOrder
```

Supported Modulation Values are: 2 (QPSK), 4 (16QAM), 6 (64QAM)

6.49.2.2 nLayer

```
uint16_t bblib_layer_llr_demap_request::nLayer
```

Number of Layers per UE, Only Support Equal Layers Number Now - valid values 1, 2

6.49.2.3 nLen

```
uint16_t bblib_layer_llr_demap_request::nLen
```

Number of Continguous Subcarrier Number

6.49.2.4 nLlrFxpPoints

```
uint8_t bblib_layer_llr_demap_request::nLlrFxpPoints
```

Indicate the Decimal Digits of Llr Output Fixed Point Value. Right now need to be 0∼7

6.49.2.5 pEqualOut

```
int16_t* bblib_layer_llr_demap_request::pEqualOut[BBLIB_MAX_TX_LAYER_NUM]
```

Data Pointer Points to Equalization Output Data, Requires Memory Continguous, format 16S13

6.49.2.6 pMmseGain

```
float* bblib_layer_llr_demap_request::pMmseGain[BBLIB_MAX_TX_LAYER_NUM]
```

Pointer Points to nTx∗1 Estimated MMSE Gain, floating number

6.49.2.7 pPostSINR

```
float* bblib_layer_llr_demap_request::pPostSINR[BBLIB_MAX_TX_LAYER_NUM]
```

Pointer Points to nTx∗1 pPostSINR, floating number

6.50 bblib_layer_llr_demap_response Struct Reference

```
#include <phy_pusch_symbol_processing_5gnr.h>
```

Data Fields

- int8_t ∗ pLlr

6.50.1 Detailed Description

Response struct of layer demapping and LR demapping processing.

6.50.2 Field Documentation

6.50.2.1 pLlr

```
int8_t* bblib_layer_llr_demap_response::pLlr
```

Pointer to Output Buffer of LLRs, buffer should be 64 byte aligned, output format 8S(nLlrFxpPoints)

6.51 bblib_layerdemapping_5gnr_request Struct Reference

```
#include <phy_layerdemapping_5gnr.h>
```

Data Fields

- int32_t * p_in [bblib_layerdemapper_max_layers]
- int32_t n_len_start
- int32_t n_in_len
- int8_t n_layer_per_ue

6.51.1 Detailed Description

Request structure containing pointer to data and its length.

6.51.2 Field Documentation

6.51.2.1 n_in_len

```
int32_t bblib_layerdemapping_5gnr_request::n_in_len
```

The layer data length.

6.51.2.2 n_layer_per_ue

```
int8_t bblib_layerdemapping_5gnr_request::n_layer_per_ue
```

The number of layer, can support 2/4 layers for 1 codeword.

6.51.2.3 n_len_start

```
int32_t bblib_layerdemapping_5gnr_request::n_len_start
```

The layer data start in symbol. It start with 0.

6.51.2.4 p_in

```
int32_t* bblib_layerdemapping_5gnr_request::p_in[bblib_layerdemapper_max_layers]
```

The layer demapping input data with multiple layer and int32_t for each sample. It should be aligned on a 64byte boundary

6.52 bblib_layerdemapping_5gnr_response Struct Reference

```
#include <phy_layerdemapping_5gnr.h>
```

Data Fields

- int32_t * p_out

6.52.1 Detailed Description

reponse structure containing pointer to output data.

6.52.2 Field Documentation

6.52.2.1 p_out

```
int32_t* bblib_layerdemapping_5gnr_response::p_out
```

The layer mapping output data array and each sample with int32_t. It should be aligned on a 64byte boundary

6.53 bblib_layermapping_5gnr_layers Struct Reference

```
#include <phy_layermapping_5gnr.h>
```

Data Fields

- void * data_output
- uint32_t data_len_layer

6.53.1 Detailed Description

reponse structure containing pointer to data and its length.

6.53.2 Field Documentation

6.53.2.1 data_len_layer

```
uint32_t bblib_layermapping_5gnr_layers::data_len_layer
```

The number of IQ values.

6.53.2.2 data_output

```
void* bblib_layermapping_5gnr_layers::data_output
```

The data output per layer. the width of data_output is base on input data with complex_int16_t or complex_float.

6.54 bblib_layermapping_5gnr_request Struct Reference

```
#include <phy_layermapping_5gnr.h>
```

Data Fields

- void * data_in
- uint32_t data_len_symbol
- uint32_t num_layer

6.54.1 Detailed Description

Request structure containing pointer to data and its length.

6.54.2 Field Documentation

6.54.2.1 data_in

```
void* bblib_layermapping_5gnr_request::data_in
```

The layer mapping input data for one codeword. where the input data can be complex_float or complex_int16_t. The data is stored in I0/Q0/I1/Q1......

6.54.2.2 data_len_symbol

```
uint32_t bblib_layermapping_5gnr_request::data_len_symbol
```

The codeword length in symbol.

6.54.2.3 num_layer

```
uint32_t bblib_layermapping_5gnr_request::num_layer
```

The number of layer, can support 2/3/4 layers for 1 codeword.

6.55 bblib_layermapping_5gnr_response Struct Reference

```
#include <phy_layermapping_5gnr.h>
```

Data Fields

- struct bblib_layermapping_5gnr_layers ** data_layer
- uint32_t num_layer

6.55.1 Detailed Description

reponse structure containing pointer to data and its length.

6.55.2 Field Documentation

6.55.2.1 data_layer

```
struct bblib_layermapping_5gnr_layers** bblib_layermapping_5gnr_response::data_layer
```

The layer mapping output data array.

6.55.2.2 num_layer

```
uint32_t bblib_layermapping_5gnr_response::num_layer
```

The layer number.

6.56 bblib_ldpc_decoder_5gnr_request Struct Reference

`#include <phy_ldpc_decoder_5gnr.h>`

Data Fields

- uint16_t Zc
- int32_t baseGraph
- int32_t nRows
- int8_t * varNodes
- int16_t numChannelLlrs
- int16_t numFillerBits
- int16_t maxIterations
- bool enableEarlyTermination

6.56.1 Detailed Description

Structure for input parameters in API of LDPC Decoder for 5GNR.

Note

...

6.56.2 Field Documentation

6.56.2.1 baseGraph

`int32_t bblib_ldpc_decoder_5gnr_request::baseGraph`

LDPC Base graph, which can be 1 or 2 as defined in TS38212-5.2.1.

6.56.2.2 enableEarlyTermination

`bool bblib_ldpc_decoder_5gnr_request::enableEarlyTermination`

When true, the decoder is allowed to terminate before maxIterations if the parity-check equations all pass

6.56.2.3 maxIterations

`int16_t bblib_ldpc_decoder_5gnr_request::maxIterations`

The maximum number of iterations that the decoder will perform before it is forced to terminate

6.56.2.4 nRows

`int32_t bblib_ldpc_decoder_5gnr_request::nRows`

Number Rows in the LDPC being used for the encoding native code rate - Minimum 4

6.56.2.5 numChannelLlrs

`int16_t bblib_ldpc_decoder_5gnr_request::numChannelLlrs`

The number of post rate-matched output LLR values

6.56.2.6 numFillerBits

`int16_t bblib_ldpc_decoder_5gnr_request::numFillerBits`

The number of filler bits used in the encoding

6.56.2.7 varNodes

`int8_t* bblib_ldpc_decoder_5gnr_request::varNodes`

Pointer to the buffer used to store the code word 8-bit integer LLRs into the top-level of the decoder to each code block This corresponds to the bit sequence d_k as defined in TS38.212-5.3.2. This should be z∗22 + z∗nRows - z∗2 - numFillerBits in length for BG1 This should be z∗10 + z∗nRows - z∗2 - numFillerBits in length for BG2 The filler bits NULL are not included

6.56.2.8 Zc

`uint16_t bblib_ldpc_decoder_5gnr_request::Zc`

Lifting factor Zc as defined in TS38212-5.2.1.

6.57 bblib_ldpc_decoder_5gnr_response Struct Reference

```
#include <phy_ldpc_decoder_5gnr.h>
```

Data Fields

- int16_t ∗ varNodes
- int numMsgBits
- uint8_t ∗ compactedMessageBytes
- int iterationAtTermination
- bool parityPassedAtTermination

6.57.1 Detailed Description

structure for outputs of LDPC decoder for 5GNR.

Note

6.57.2 Field Documentation

6.57.2.1 compactedMessageBytes

```
uint8_t* bblib_ldpc_decoder_5gnr_response::compactedMessageBytes
```

Decoded Message Data

6.57.2.2 iterationAtTermination

```
int bblib_ldpc_decoder_5gnr_response::iterationAtTermination
```

The number of iterations executed before termination.

6.57.2.3 numMsgBits

```
int bblib_ldpc_decoder_5gnr_response::numMsgBits
```

Output message stored as individual bits in a pre-allocated buffer. Number of bytes allocation *must* be $>=$ ceil((z$*$22 - numFillerBits)/8) for BG1 Number of bytes allocation *must* be $>=$ ceil((z$*$10 - numFillerBits)/8) for BG2

6.57.2.4 parityPassedAtTermination

```
bool bblib_ldpc_decoder_5gnr_response::parityPassedAtTermination
```

True if the parity checks all had passed at termination (Always true if response.iterationAtTermination $<$ request.↩ maxIterations).

6.57.2.5 varNodes

```
int16_t* bblib_ldpc_decoder_5gnr_response::varNodes
```

Pointer to the buffer used to store the code word 16-bit LLR outputs Space allocation *must* be $>=$ z$*$22 + z$*$nRows for BG1 Space allocation *must* be $>=$ z$*$10 + z$*$nRows for BG2

6.58 bblib_ldpc_encoder_5gnr_request Struct Reference

```
#include <phy_ldpc_encoder_5gnr.h>
```

Data Fields

- uint16_t Zc
- int32_t baseGraph
- int32_t nRows
- int8_t numberCodeblocks
- int8_t * input [MAX_CB_BLOCK]

6.58.1 Detailed Description

Structure for input parameters in API of LDPC Encoder for 5GNR.

Note

...

6.58.2 Field Documentation

6.58.2.1 baseGraph

```
int32_t bblib_ldpc_encoder_5gnr_request::baseGraph
```

LDPC Base graph, which can be 1 or 2 as defined in TS38212-5.2.1.

6.58.2.2 input

```
int8_t* bblib_ldpc_encoder_5gnr_request::input[MAX_CB_BLOCK]
```

Pointer to input stream related to each code block This corresponds to the bit sequence c_k as defined in TS38.↩ 212-5.3.2. This includes therefore the filler bits set as 0.

6.58.2.3 nRows

```
int32_t bblib_ldpc_encoder_5gnr_request::nRows
```

Number Rows being used for the encoding native code rate - Minimum 4

6.58.2.4 numberCodeblocks

`int8_t bblib_ldpc_encoder_5gnr_request::numberCodeblocks`

Used to run several code blocks in one operation notably for low expansion factor All code blocks must use the same parameters above. numberCodeblocks ∗ Zc must not exceed 512

6.58.2.5 Zc

`uint16_t bblib_ldpc_encoder_5gnr_request::Zc`

Lifting factor Zc as defined in TS38212-5.2.1.

6.59 bblib_ldpc_encoder_5gnr_response Struct Reference

`#include <phy_ldpc_encoder_5gnr.h>`

Data Fields

- int8_t ∗ output [MAX_CB_BLOCK]

6.59.1 Detailed Description

structure for outputs of LDPC encoder for 5GNR.

Note

6.59.2 Field Documentation

6.59.2.1 output

`int8_t* bblib_ldpc_encoder_5gnr_response::output[MAX_CB_BLOCK]`

Output buffer for data stream after LDPC Encoding for each CodeBlocks This corresponds to the parity bit sequence w_k as defined in TS38.212-5.3.2 The actual length is limited to the number of rows requested.

6.60 bblib_LDPC_ratematch_5gnr_request Struct Reference

`#include <phy_LDPC_ratematch_5gnr.h>`

Data Fields

- int32_t Ncb
- int32_t Zc
- int32_t E
- int32_t Qm
- int32_t rvidx
- int32_t baseGraph
- int32_t nullIndex
- int32_t nLen
- uint8_t * input

6.60.1 Detailed Description

Structure for input parameters in API of rate matching for 5GNR.

Note

input data alignment depends on modulation type.
For BPSK, no alignment requirement.
For QPSK, input should be aligned with 2 BITS.
For 16QAM, input should be aligned with 4 BITS.
For 64QAM, input should be aligned with 6 BITS.
For 256QAM, input should be aligned with 1 Byte.

6.60.2 Field Documentation

6.60.2.1 baseGraph

`int32_t bblib_LDPC_ratematch_5gnr_request::baseGraph`

Base graph, which can be 1/2.

6.60.2.2 E

`int32_t bblib_LDPC_ratematch_5gnr_request::E`

Length of the output buffer in bits. Currently limited to 8448*8 bits

6.60.2.3 input

`uint8_t* bblib_LDPC_ratematch_5gnr_request::input`

pointer to input stream. alignment depends on modulation type

6.60.2.4 Ncb

`int32_t bblib_LDPC_ratematch_5gnr_request::Ncb`

Length of the circular buffer in bits.

6.60.2.5 nLen

`int32_t bblib_LDPC_ratematch_5gnr_request::nLen`

Length of null bits. 0 if no null bit

6.60.2.6 nullIndex

`int32_t bblib_LDPC_ratematch_5gnr_request::nullIndex`

Position of starting null bits. -1 if no null bit

6.60.2.7 Qm

`int32_t bblib_LDPC_ratematch_5gnr_request::Qm`

Modulation type, which can be 1/2/4/6/8.

6.60.2.8 rvidx

`int32_t bblib_LDPC_ratematch_5gnr_request::rvidx`

Redundancy version, which can be 0/1/2/3.

6.60.2.9 Zc

`int32_t bblib_LDPC_ratematch_5gnr_request::Zc`

Parameter defined in TS 38211-5.2.1.

6.61 bblib_LDPC_ratematch_5gnr_response Struct Reference

```
#include <phy_LDPC_ratematch_5gnr.h>
```

Data Fields

- uint8_t * output

6.61.1 Detailed Description

structure for outputs of rate matching for 5GNR.

Note

output data alignment depends on modulation type. For BPSK, no alignment requirement For QPSK, input should be aligned with 2 BITS For 16QAM, input should be aligned with 4 BITS For 64QAM, input should be aligned with 6 BITS For 256QAM, input should be aligned with 1 Byte

6.61.2 Field Documentation

6.61.2.1 output

```
uint8_t* bblib_LDPC_ratematch_5gnr_response::output
```

Output buffer for data stream after rate matching. alignment depends on modulation type

6.62 bblib_llr_demapping_5gnr_request Struct Reference

```
#include <phy_llr_demapping.h>
```

Data Fields

- void ∗ rx
- void ∗ magnitude
- void ∗ reciprocal_noise_var
- enum bblib_modulation_order modulation
- int num_symbols
- enum bblib_llr_demapping_5gnr_io_format io_format

6.62.1 Detailed Description

Structure defining the LLR Demapper input interface for 5G NR.

6.62.2 Field Documentation

6.62.2.1 io_format

```
enum bblib_llr_demapping_5gnr_io_format bblib_llr_demapping_5gnr_request::io_format
```

Input and output data format enum.

6.62.2.2 magnitude

```
void* bblib_llr_demapping_5gnr_request::magnitude
```

Pointer to buffer of magnitudes per symbol - buffer must be 64 byte aligned. Data type can be float or uint16_t for fixed point format. Fixed point data must be Q16u16 format with range 0 to 0.9999.

6.62.2.3 modulation

```
enum bblib_modulation_order bblib_llr_demapping_5gnr_request::modulation
```

Supported modulation values are: 2 (QPSK), 4 (16QAM), 6 (64QAM), 8 (256QAM).

6.62.2.4 num_symbols

```
int bblib_llr_demapping_5gnr_request::num_symbols
```

Number of input symbols - the number of complex elements in input rx buffer.

6.62.2.5 reciprocal_noise_var

```
void* bblib_llr_demapping_5gnr_request::reciprocal_noise_var
```

Pointer to buffer of reciprocal of the noise variance per symbol, buffer must be 64 byte aligned. Data type can be float or uint16_t for fixed point format. Fixed point data must be Q16u6 format with range 0.5 to 629.

6.62.2.6 rx

```
void* bblib_llr_demapping_5gnr_request::rx
```

Pointer to buffer of the input symbols - buffer must be 64 byte aligned. Data type can be float or int16_t for fixed point format. Fixed point data must be Q16s13 format with range +/-1.14.

6.63 bblib_llr_demapping_5gnr_response Struct Reference

```
#include <phy_llr_demapping.h>
```

Data Fields

- void * llrs
- int num_llrs

6.63.1 Detailed Description

Structure defining the LLR Demapper output interface for 5G NR.

6.63.2 Field Documentation

6.63.2.1 llrs

```
void* bblib_llr_demapping_5gnr_response::llrs
```

Pointer to output buffer of LLRs - buffer should be 64 byte aligned. Data type can be float or int16_t for fixed point format. Fixed point data is Q8s3 format with range +/-16.

6.63.2.2 num_llrs

```
int bblib_llr_demapping_5gnr_response::num_llrs
```

Total number of output LLR - for each input symbol the number of outputs is: QPSK : 2, 16QAM : 4, 64QAM : 6, 256QAM : 8.

6.64 bblib_llr_demapping_nn_5gnr_request Struct Reference

```
#include <phy_llr_demapping.h>
```

Data Fields

- int16_t * rx
- int16_t shift
- enum bblib_modulation_order modulation
- int num_symbols

6.64.1 Detailed Description

Structure defining the LLR Demapper input interface for 5G NR.

Overview: The NN demapper performs symbol to LLR demapping using nearest neighbour method and supports QPSK, 16QAM, 64QAM and 256QAM modulation order.

Note: The implementation has a single scale value used to calculate the LLR thresholds

The NN demapper supports 16bit fixed point. The input and output buffer allocation should be rounded up to a multiple of the register width.

Warning

EXPERIMENTAL: Further optimization is possible, API may change without prior notice.

6.64.2 Field Documentation

6.64.2.1 modulation

`enum bblib_modulation_order bblib_llr_demapping_nn_5gnr_request::modulation`

Supported modulation values are: 2 (QPSK), 4 (16QAM), 6 (64QAM), 8 (256QAM).

6.64.2.2 num_symbols

`int bblib_llr_demapping_nn_5gnr_request::num_symbols`

Number of complex input symbols.

6.64.2.3 rx

`int16_t* bblib_llr_demapping_nn_5gnr_request::rx`

Pointer to buffer of the input symbols - buffer must be 64 byte aligned. Format 16Sx.

6.64.2.4 shift

`int16_t bblib_llr_demapping_nn_5gnr_request::shift`

16bit shift value used to scale the input samples.

6.65 bblib_llr_demapping_nn_5gnr_response Struct Reference

`#include <phy_llr_demapping.h>`

Data Fields

- int8_t * llrs

6.65.1 Detailed Description

Structure defining the Simple Demapper output interface for 5G NR.

6.65.2 Field Documentation

6.65.2.1 llrs

`int8_t* bblib_llr_demapping_nn_5gnr_response::llrs`

Pointer to output buffer of LLRs - buffer should be 64 byte aligned. Format 8S(x-shift)

6.66 bblib_llr_demapping_request Struct Reference

`#include <phy_llr_demapping.h>`

Data Fields

- complex_float * rx
- float magnitude
- float noise_var
- enum bblib_modulation_order modulation
- int num_symbols
- enum bblib_llr_demapping_5gnr_io_format io_format

6.66.1 Detailed Description

Structure defining the LLR Demapper input interface.

6.66.2 Field Documentation

6.66.2.1 io_format

`enum bblib_llr_demapping_5gnr_io_format bblib_llr_demapping_request::io_format`

Input and output data format enum, LTE supports floating point input with fixed or float output

6.66.2.2 magnitude

`float bblib_llr_demapping_request::magnitude`

The amplitude of the +1 constellation point.

6.66.2.3 modulation

`enum bblib_modulation_order bblib_llr_demapping_request::modulation`

Supported values are: 2 (QPSK) 4 (16QAM) 6 (64QAM) 8 (256QAM).

6.66.2.4 noise_var

```
float bblib_llr_demapping_request::noise_var
```

The noise variance.

6.66.2.5 num_symbols

```
int bblib_llr_demapping_request::num_symbols
```

Number of input symbols - the number of complex elements in input rx buffer

6.66.2.6 rx

```
complex_float* bblib_llr_demapping_request::rx
```

Pointer to buffer of the input symbols, buffer must be 64 byte aligned.

6.67 bblib_llr_demapping_response Struct Reference

```
#include <phy_llr_demapping.h>
```

Data Fields

- void * llrs
- int num_llrs

6.67.1 Detailed Description

Structure defining the LLR Demapper output interface.

6.67.2 Field Documentation

6.67.2.1 llrs

```
void* bblib_llr_demapping_response::llrs
```

Pointer to output buffer of LLRs - buffer should be 64 byte aligned. Data type can be float or int16_t for fixed point format. Fixed point data is Q8s3 format with range +/-16.

6.67.2.2 num_llrs

```
int bblib_llr_demapping_response::num_llrs
```

Total number of output LLR - for each input input symbol number of outputs is: QPSK : 2, 16QAM : 4, 64QAM : 6, 256QAM : 8.

6.68 bblib_lte_mu_mimo_equalize_request Struct Reference

```
#include <phy_lte_mu_mimo_equalize.h>
```

Data Fields

- int16_t Nrb_sc
- int16_t RBStart
- int32_t Nrx_antennas
- int16_t numSubCarrier
- mu_mimo_puschshorten_flag PuschShortenFlag
- int16_t inputOffset
- void ∗ mimochEst_usr0_fl [NUM_SLOTS_PER_SUBF][MAX_NUM_ANT]
- void ∗ mimochEst_usr1_fl [NUM_SLOTS_PER_SUBF][MAX_NUM_ANT]
- void ∗ mimoChEst_slope_usr0_fl [MAX_NUM_ANT]
- void ∗ mimoChEst_slope_usr1_fl [MAX_NUM_ANT]
- void ∗ timeDerotation_usr0_fl
- void ∗ timeDerotation_usr1_fl
- void ∗ ahEstPtr [MAX_SYM_PER_SUBFRAME][MAX_NUM_ANT]

6.68.1 Detailed Description

Request structure for mu_mimo channel equalization.

6.68.2 Field Documentation

6.68.2.1 ahEstPtr

```
void* bblib_lte_mu_mimo_equalize_request::ahEstPtr[MAX_SYM_PER_SUBFRAME][MAX_NUM_ANT]
```

FFT output data for all UL symbols and rx antennas, memory should be 64 byte aligned

6.68.2.2 inputOffset

```
int16_t bblib_lte_mu_mimo_equalize_request::inputOffset
```

Offset size of channle user date

6.68.2.3 mimoChEst_slope_usr0_fl

`void* bblib_lte_mu_mimo_equalize_request::mimoChEst_slope_usr0_fl[MAX_NUM_ANT]`

channel slope estimate for time domain interpolation data for user0, memory should be 64 byte aligned

6.68.2.4 mimoChEst_slope_usr1_fl

`void* bblib_lte_mu_mimo_equalize_request::mimoChEst_slope_usr1_fl[MAX_NUM_ANT]`

channel slope estimate for time domain interpolation data for user1, memory should be 64 byte aligned

6.68.2.5 mimochEst_usr0_fl

`void* bblib_lte_mu_mimo_equalize_request::mimochEst_usr0_fl[NUM_SLOTS_PER_SUBF][MAX_NUM_ANT]`

channel estimate data for user0, memory should be 64 byte aligned

6.68.2.6 mimochEst_usr1_fl

`void* bblib_lte_mu_mimo_equalize_request::mimochEst_usr1_fl[NUM_SLOTS_PER_SUBF][MAX_NUM_ANT]`

channel estimate data for user1, memory should be 64 byte aligned

6.68.2.7 Nrb_sc

`int16_t bblib_lte_mu_mimo_equalize_request::Nrb_sc`

Number of subcarriers per resource block

6.68.2.8 Nrx_antennas

`int32_t bblib_lte_mu_mimo_equalize_request::Nrx_antennas`

Number of receiving antennas

6.68.2.9 numSubCarrier

`int16_t bblib_lte_mu_mimo_equalize_request::numSubCarrier`

Number of subcarriers

6.68.2.10 PuschShortenFlag

`mu_mimo_puschshorten_flag bblib_lte_mu_mimo_equalize_request::PuschShortenFlag`

PUSCH Shorten Flag

6.68.2.11 RBStart

`int16_t bblib_lte_mu_mimo_equalize_request::RBStart`

Index of start resource block

6.68.2.12 timeDerotation_usr0_fl

`void* bblib_lte_mu_mimo_equalize_request::timeDerotation_usr0_fl`

time derotation data, the buffer size equal to number of subcarriers for user0, memory should be 64 byte aligned

6.68.2.13 timeDerotation_usr1_fl

`void* bblib_lte_mu_mimo_equalize_request::timeDerotation_usr1_fl`

time derotation data, the buffer size equal to number of subcarriers for user1, memory should be 64 byte aligned

6.69 bblib_lte_mu_mimo_equalize_response Struct Reference

`#include <phy_lte_mu_mimo_equalize.h>`

Data Fields

- void * zEstUsr0_fl
- void * zEstUsr1_fl

6.69.1 Detailed Description

Reponse structure for mu_mimo channel equalization.

6.69.2 Field Documentation

6.69.2.1 zEstUsr0_fl

`void* bblib_lte_mu_mimo_equalize_response::zEstUsr0_fl`

the equalize output data for user0, memory should be 64 byte aligned

6.69.2.2 zEstUsr1_fl

```
void* bblib_lte_mu_mimo_equalize_response::zEstUsr1_fl
```

the equalize output data for user1, memory should be 64 byte aligned

6.70 bblib_lte_su_mimo_equalize_request Struct Reference

```
#include <phy_lte_su_mimo_equalize.h>
```

Data Fields

- puschshorten_flag PuschShortenFlag
- int32_t numAnt
- int16_t numSubCarrier
- enum bblib_modulation_order Qm
- void ∗ chEst_usr_fl [NUM_SLOTS_PER_SUBF][MAX_NUM_ANT]
- void ∗ timeDerotation_usr_fl
- void ∗ chSlope_usr_fl [MAX_NUM_ANT]
- void ∗ noiseVarEst_chan_fl
- void ∗ dataSubc_usr_fl [MAX_PUSCH_DATASYMB_PER_SUBF][MAX_NUM_ANT]

6.70.1 Detailed Description

Request structure for su_mimo channel equalization.

6.70.2 Field Documentation

6.70.2.1 chEst_usr_fl

```
void* bblib_lte_su_mimo_equalize_request::chEst_usr_fl[NUM_SLOTS_PER_SUBF][MAX_NUM_ANT]
```

channel estimate data, memory should be 64 byte aligned

6.70.2.2 chSlope_usr_fl

```
void* bblib_lte_su_mimo_equalize_request::chSlope_usr_fl[MAX_NUM_ANT]
```

channel slope estimate for time domain interpolation data, memory should be 64 byte aligned

6.70.2.3 dataSubc_usr_fl

`void* bblib_lte_su_mimo_equalize_request::dataSubc_usr_fl[MAX_PUSCH_DATASYMB_PER_SUBF][MAX_NUM_ANT]`

rx resource grid FFT output data for PUSCH, memory should be 64 byte aligned

6.70.2.4 noiseVarEst_chan_fl

`void* bblib_lte_su_mimo_equalize_request::noiseVarEst_chan_fl`

noise variance estimate data, memory should be 64 byte aligned

6.70.2.5 numAnt

`int32_t bblib_lte_su_mimo_equalize_request::numAnt`

Number of receiving antennas

6.70.2.6 numSubCarrier

`int16_t bblib_lte_su_mimo_equalize_request::numSubCarrier`

Number of subcarriers

6.70.2.7 PuschShortenFlag

`puschshorten_flag bblib_lte_su_mimo_equalize_request::PuschShortenFlag`

PUSCH shorten flag

6.70.2.8 Qm

`enum bblib_modulation_order bblib_lte_su_mimo_equalize_request::Qm`

Number of modulation order

6.70.2.9 timeDerotation_usr_fl

`void* bblib_lte_su_mimo_equalize_request::timeDerotation_usr_fl`

time derotation data, the buffer size equal to number of subcarriers, memory should be 64 byte aligned

6.71 bblib_lte_su_mimo_equalize_response Struct Reference

`#include <phy_lte_su_mimo_equalize.h>`

Data Fields

- void * zEst_fl

6.71.1 Detailed Description

Reponse structure for su_mimo channel equalization.

6.71.2 Field Documentation

6.71.2.1 zEst_fl

```
void* bblib_lte_su_mimo_equalize_response::zEst_fl
```

the equalize output data, memory should be 64 byte aligned

6.72 bblib_lte_viterbi_decoder_request Struct Reference

```
#include <phy_viterbi_decoder.h>
```

Data Fields

- int16_t k
- int8_t ** llr

6.72.1 Detailed Description

Request structure for viterbi algorithm.

6.72.2 Field Documentation

6.72.2.1 k

```
int16_t bblib_lte_viterbi_decoder_request::k
```

Number of output bits including the CRC bits.

6.72.2.2 llr

```
int8_t** bblib_lte_viterbi_decoder_request::llr
```

Input soft decisions.

6.73 bblib_lte_viterbi_decoder_response Struct Reference

```
#include <phy_viterbi_decoder.h>
```

Data Fields

- int8_t ∗ output

6.73.1 Detailed Description

Response structure for viterbi algorithm.

6.73.2 Field Documentation

6.73.2.1 output

```
int8_t* bblib_lte_viterbi_decoder_response::output
```

Viterbi decoder output buffer. Each byte will contain 1 bit of information.

6.74 bblib_matrix_inv_interleave_request Struct Reference

```
#include <phy_matrix_inversion.h>
```

Data Fields

- complex_float ∗ input
- enum bblib_matrix_inv_type op
- int num_inputs_received

6.74.1 Detailed Description

The request structure used to setup the matrix inverse operation using interleaved inputs. Matrix size should be one of 2x2, 4x4, 6x6, 8x8 or 16x16.

6.74.2 Field Documentation

6.74.2.1 input

`complex_float* bblib_matrix_inv_interleave_request::input`

Pointer to the array containing the floating point complex inputs. The matrices in the array are stored in the following order, starting from the top left entry in each matrix:

Ma0Mb0Mc0Ma1Mb1Mc1Ma2Mb2Mc2Ma3Mb3Mc3

Array must be 64byte aligned.

6.74.2.2 num_inputs_received

`int bblib_matrix_inv_interleave_request::num_inputs_received`

Integer value representing the size of the input array, and hence number of matrix operations to perform. Valid range between 1 - 64.

6.74.2.3 op

`enum bblib_matrix_inv_type bblib_matrix_inv_interleave_request::op`

Enum describing the type of operation to perform.

6.75 bblib_matrix_inv_interleave_response Struct Reference

```
#include <phy_matrix_inversion.h>
```

Data Fields

- complex_float * output
- int num_outputs_proc

6.75.1 Detailed Description

Common matrix response structure returned containing the result of an operation using interleaved inputs.

6.75.2 Field Documentation

6.75.2.1 num_outputs_proc

`int bblib_matrix_inv_interleave_response::num_outputs_proc`

Integer value representing the number of entries in the output array. This should match num_inputs_received.

6.75.2.2 output

`complex_float* bblib_matrix_inv_interleave_response::output`

Pointer to the array containing the floating point complex outputs. The matrices in the array are stored in the following order, starting from the top left entry in each matrix:

Ma0Mb0Mc0Ma1Mb1Mc1Ma2Mb2Mc2Ma3Mb3Mc3

The size of this array must match the size of the input array. Array must be 64byte aligned.

6.76 bblib_matrix_inv_request Struct Reference

`#include <phy_matrix_inversion.h>`

Data Fields

- complex_float ** input
- enum bblib_matrix_inv_type op
- int num_inputs_received

6.76.1 Detailed Description

The request structure used to setup the matrix inverse operation. Matrix size should be one of 2x2, 4x4, 6x6, 8x8 or 16x16.

6.76.2 Field Documentation

6.76.2.1 input

`complex_float** bblib_matrix_inv_request::input`

Pointer to an array of pointers to 64byte aligned buffers containing the floating point complex inputs.

6.76.2.2 num_inputs_received

```
int bblib_matrix_inv_request::num_inputs_received
```

Integer value representing the size of the input array, and hence number of matrix operations to perform. Valid range between 1 - 64.

6.76.2.3 op

```
enum bblib_matrix_inv_type bblib_matrix_inv_request::op
```

Enum describing the type of operation to perform.

6.77 bblib_matrix_inv_response Struct Reference

```
#include <phy_matrix_inversion.h>
```

Data Fields

- complex_float ** output
- int num_outputs_proc

6.77.1 Detailed Description

Common matrix response structure returned containing the result of an operation.

6.77.2 Field Documentation

6.77.2.1 num_outputs_proc

```
int bblib_matrix_inv_response::num_outputs_proc
```

Integer value representing the number of entries in the output array. This should match num_inputs_received.

6.77.2.2 output

```
complex_float** bblib_matrix_inv_response::output
```

Pointer to an array of pointers to 64byte aligned output buffers that will contain the output of the operation. The size of this array must match the size of the input array.

6.78 bblib_mimo_mmse_5gqrd_request Struct Reference

```
#include <phy_mmse_mimo_qrd_5gtf.h>
```

Data Fields

- void * ch_state [MAX_BS_ANT_RX][MAX_UE_ANT_TX]
- int32_t sigma2
- int16_t start_prb
- int16_t n_subcarrier

6.78.1 Detailed Description

Request structure for 5G QRD mimo.

6.78.2 Field Documentation

6.78.2.1 ch_state

```
void* bblib_mimo_mmse_5gqrd_request::ch_state[MAX_BS_ANT_RX][MAX_UE_ANT_TX]
```

Points to nRx*nTx channel.

6.78.2.2 n_subcarrier

```
int16_t bblib_mimo_mmse_5gqrd_request::n_subcarrier
```

Number of SCs granted for one UE.

6.78.2.3 sigma2

```
int32_t bblib_mimo_mmse_5gqrd_request::sigma2
```

Noise power.

6.78.2.4 start_prb

```
int16_t bblib_mimo_mmse_5gqrd_request::start_prb
```

Start PRB number of current UE.

6.79 bblib_mimo_mmse_5gqrd_response Struct Reference

`#include <phy_mmse_mimo_qrd_5gtf.h>`

Data Fields

- void * weight [MAX_BS_ANT_RX][MAX_UE_ANT_TX]

6.79.1 Detailed Description

Response structure for 5G QRD mimo.

6.79.2 Field Documentation

6.79.2.1 weight

`void* bblib_mimo_mmse_5gqrd_response::weight[MAX_BS_ANT_RX][MAX_UE_ANT_TX]`

Output param.

6.80 bblib_mmse_irc_mimo_5gnr_request Struct Reference

`#include <phy_mmse_irc_mimo_5gnr.h>`

Data Fields

- int32_t nLayer
- int32_t nRxAnt
- void * pChState [BBLIB_MAX_RX_ANT_NUM][BBLIB_MAX_TX_LAYER_NUM]
- void * pRxSignal [BBLIB_MAX_RX_ANT_NUM][BBLIB_N_SYMB_PER_SF]
- void * pRnn_Re [BBLIB_N_SYMB_PER_SF][BBLIB_MAX_RX_ANT_NUM][BBLIB_MAX_RX_ANT_NUM]
- void * pRnn_Im [BBLIB_N_SYMB_PER_SF][BBLIB_MAX_RX_ANT_NUM][BBLIB_MAX_RX_ANT_NUM]
- int16_t nStartSC
- int16_t nSubCarrier
- int16_t nTotalAlignedSubCarrier
- int16_t nChSymb
- int16_t nSymb
- int16_t nSymbPerDmrs [BBLIB_N_SYMB_PER_SF]
- int16_t * pSymbIndex
- int16_t nLinInterpEnable
- int16_t nDmrsChSymb
- int16_t nMappingType
- int16_t nGranularity
- float fEstCfo [BBLIB_MAX_TX_LAYER_NUM]
- int16_t nFftSize
- int16_t nNumerology
- int16_t nEnableFoComp

6.80.1 Detailed Description

Structure defines the mmse_irc_mimo function request interface in 5GNR.

6.80.2 Field Documentation

6.80.2.1 fEstCfo

`float bblib_mmse_irc_mimo_5gnr_request::fEstCfo[BBLIB_MAX_TX_LAYER_NUM]`

Normalized frequency offset estimates for each layer

6.80.2.2 nChSymb

`int16_t bblib_mmse_irc_mimo_5gnr_request::nChSymb`

Number of Channel Symbols - valid range [1-(N_SYMB_PER_SF-1)]

6.80.2.3 nDmrsChSymb

`int16_t bblib_mmse_irc_mimo_5gnr_request::nDmrsChSymb`

Number of Dmrs Symbols - valid range [1-4]. Defines how many CEs stored on input at this function call

6.80.2.4 nEnableFoComp

`int16_t bblib_mmse_irc_mimo_5gnr_request::nEnableFoComp`

Flag to enable frequency offset compensation

6.80.2.5 nFftSize

`int16_t bblib_mmse_irc_mimo_5gnr_request::nFftSize`

FFT Size

6.80.2.6 nGranularity

`int16_t bblib_mmse_irc_mimo_5gnr_request::nGranularity`

Defines the number of adjacent symbols sharing the same CE inside the slot

6.80.2.7 nLayer

`int32_t bblib_mmse_irc_mimo_5gnr_request::nLayer`

Number of Layers - valid values 1, 2, 4, 8, 16

6.80.2.8 nLinInterpEnable

`int16_t bblib_mmse_irc_mimo_5gnr_request::nLinInterpEnable`

0 - stored DMRS CE is used directly (nearest neighbor), 1 - time linear interpolation is used

6.80.2.9 nMappingType

`int16_t bblib_mmse_irc_mimo_5gnr_request::nMappingType`

Dmrs Type - A is 0 and B is 1

6.80.2.10 nNumerology

`int16_t bblib_mmse_irc_mimo_5gnr_request::nNumerology`

Numerology

6.80.2.11 nRxAnt

`int32_t bblib_mmse_irc_mimo_5gnr_request::nRxAnt`

Number of Rx antennas - valid values 1, 2, 4, 8, 16

6.80.2.12 nStartSC

`int16_t bblib_mmse_irc_mimo_5gnr_request::nStartSC`

Start Subcarrier

6.80.2.13 nSubCarrier

`int16_t bblib_mmse_irc_mimo_5gnr_request::nSubCarrier`

Number of granted subcarriers - valid range [1-3276]

6.80.2.14 nSymb

`int16_t bblib_mmse_irc_mimo_5gnr_request::nSymb`

Number of granted Symbols - valid range [1-(N_SYMB_PER_SF-1)]

6.80.2.15 nSymbPerDmrs

`int16_t bblib_mmse_irc_mimo_5gnr_request::nSymbPerDmrs[BBLIB_N_SYMB_PER_SF]`

Number of granted Symbols Per Dmrs - valid range [1-(N_SYMB_PER_SF-1)]

6.80.2.16 nTotalAlignedSubCarrier

`int16_t bblib_mmse_irc_mimo_5gnr_request::nTotalAlignedSubCarrier`

Number of subcarriers aligned - valid range [1-3280]

6.80.2.17 pChState

`void* bblib_mmse_irc_mimo_5gnr_request::pChState[BBLIB_MAX_RX_ANT_NUM][BBLIB_MAX_TX_LAYER_NUM]`

Data pointer points to nRxAnt∗nTxLayer channel

6.80.2.18 pRnn_Im

`void* bblib_mmse_irc_mimo_5gnr_request::pRnn_Im[BBLIB_N_SYMB_PER_SF][BBLIB_MAX_RX_ANT_NUM][BBLIB_MAX_RX_ANT_NU`

Data pointer points to nRxAnt∗nRxAnt∗nSymbol Rnn data

6.80.2.19 pRnn_Re

`void* bblib_mmse_irc_mimo_5gnr_request::pRnn_Re[BBLIB_N_SYMB_PER_SF][BBLIB_MAX_RX_ANT_NUM][BBLIB_MAX_RX_ANT_NU`

Data pointer points to nRxAnt∗nRxAnt∗nSymbol Rnn data

6.80.2.20 pRxSignal

`void* bblib_mmse_irc_mimo_5gnr_request::pRxSignal[BBLIB_MAX_RX_ANT_NUM][BBLIB_N_SYMB_PER_SF]`

Data pointer points to nRxAnt∗nSymbol received data

6.80.2.21 pSymbIndex

`int16_t* bblib_mmse_irc_mimo_5gnr_request::pSymbIndex`

Pointer for Data Symbol index

6.81 bblib_mmse_irc_mimo_5gnr_response Struct Reference

```
#include <phy_mmse_irc_mimo_5gnr.h>
```

Data Fields

- void ∗ pEstTxSignal [BBLIB_MAX_TX_LAYER_NUM][BBLIB_N_SYMB_PER_SF]
- void ∗ pGain [BBLIB_MAX_TX_LAYER_NUM]

6.81.1 Detailed Description

Structure defines the mmse_irc_mimo response interface in 5GNR.

6.81.2 Field Documentation

6.81.2.1 pEstTxSignal

`void* bblib_mmse_irc_mimo_5gnr_response::pEstTxSignal[BBLIB_MAX_TX_LAYER_NUM][BBLIB_N_SYMB_PER_SF]`

Data pointer points to nTx∗nSymbol estimated TX signal

6.81.2.2 pGain

`void* bblib_mmse_irc_mimo_5gnr_response::pGain[BBLIB_MAX_TX_LAYER_NUM]`

Pointer points to nTx∗1 estimated post SINR

6.82 bblib_mmse_mimo_request Struct Reference

`#include <phy_rx_mimo_mmse.h>`

Data Fields

- int32_t nLayer
- int32_t nRxAnt
- void ∗ pChState [BBLIB_MAX_RX_ANT_NUM][BBLIB_MAX_TX_LAYER_NUM]
- void ∗ pRxSignal [BBLIB_MAX_RX_ANT_NUM][BBLIB_N_SYMB_PER_SF]
- int32_t nSigma2
- int16_t nStartSC
- int16_t nSubCarrier
- int16_t nTotalAlignedSubCarrier
- int16_t nChSymb
- int16_t nSymb
- int16_t nSymbPerDmrs [BBLIB_N_SYMB_PER_SF]
- int16_t ∗ pSymbIndex
- int16_t nLinInterpEnable
- int16_t nDmrsChSymb
- int16_t nMappingType
- int16_t nGranularity
- float fEstCfo [BBLIB_MAX_TX_LAYER_NUM]
- int16_t nFftSize
- int16_t nNumerology
- int16_t nEnableFoComp

6.82.1 Detailed Description

Request struct of MMSE MIMO.

6.82.2 Field Documentation

6.82.2.1 fEstCfo

`float bblib_mmse_mimo_request::fEstCfo[BBLIB_MAX_TX_LAYER_NUM]`

Normalized frequency offset estimates for each layer

6.82.2.2 nChSymb

`int16_t bblib_mmse_mimo_request::nChSymb`

Number of Channel Symbols - valid range [1-(N_SYMB_PER_SF-1)]

6.82.2.3 nDmrsChSymb

`int16_t bblib_mmse_mimo_request::nDmrsChSymb`

Number of Dmrs Symbols - valid range [1-4]. Defines how many CEs stored on input at this function call

6.82.2.4 nEnableFoComp

`int16_t bblib_mmse_mimo_request::nEnableFoComp`

Flag to enable frequency offset compensation

6.82.2.5 nFftSize

`int16_t bblib_mmse_mimo_request::nFftSize`

FFT Size

6.82.2.6 nGranularity

`int16_t bblib_mmse_mimo_request::nGranularity`

Defines the number of adjacent symbols sharing the same CE inside the slot

6.82.2.7 nLayer

`int32_t bblib_mmse_mimo_request::nLayer`

Number of Layers - valid values 1, 2, 4, 8, 16

6.82.2.8 nLinInterpEnable

`int16_t bblib_mmse_mimo_request::nLinInterpEnable`

0 - stored DMRS CE is used directly (nearest neighbor), 1 - time linear interpolation is used

6.82.2.9 nMappingType

`int16_t bblib_mmse_mimo_request::nMappingType`

Dmrs Type - A is 0 and B is 1

6.82.2.10 nNumerology

`int16_t bblib_mmse_mimo_request::nNumerology`

Numerology

6.82.2.11 nRxAnt

`int32_t bblib_mmse_mimo_request::nRxAnt`

Number of Rx antennas - valid values 1, 2, 4, 8, 16

6.82.2.12 nSigma2

`int32_t bblib_mmse_mimo_request::nSigma2`

Noise power

6.82.2.13 nStartSC

`int16_t bblib_mmse_mimo_request::nStartSC`

Start Subcarrier

6.82.2.14 nSubCarrier

`int16_t bblib_mmse_mimo_request::nSubCarrier`

Number of granted subcarriers - valid range [1-3276]

6.82.2.15 nSymb

`int16_t bblib_mmse_mimo_request::nSymb`

Number of granted Symbols - valid range [1-(N_SYMB_PER_SF-1)]

6.82.2.16 nSymbPerDmrs

`int16_t bblib_mmse_mimo_request::nSymbPerDmrs[BBLIB_N_SYMB_PER_SF]`

Number of granted Symbols Per Dmrs - valid range [1-(N_SYMB_PER_SF-1)]

6.82.2.17 nTotalAlignedSubCarrier

`int16_t bblib_mmse_mimo_request::nTotalAlignedSubCarrier`

Number of Aligned Total Granted Subcarriers, Decide the Buffer Offset, Need to Be Mulitple of 16.

6.82.2.18 pChState

`void* bblib_mmse_mimo_request::pChState[BBLIB_MAX_RX_ANT_NUM][BBLIB_MAX_TX_LAYER_NUM]`

Data pointer points to nRxAnt∗nTxLayer channel, format 16S13

6.82.2.19 pRxSignal

`void* bblib_mmse_mimo_request::pRxSignal[BBLIB_MAX_RX_ANT_NUM][BBLIB_N_SYMB_PER_SF]`

Data pointer points to nRxAnt∗nSymbol received data, format 16S13

6.82.2.20 pSymbIndex

`int16_t* bblib_mmse_mimo_request::pSymbIndex`

Pointer for Data Symbol index

6.83 bblib_mmse_mimo_response Struct Reference

`#include <phy_rx_mimo_mmse.h>`

Data Fields

- void ∗ pEstTxSignal [BBLIB_MAX_TX_LAYER_NUM][BBLIB_N_SYMB_PER_SF]
- void ∗ pPostSINR [BBLIB_MAX_TX_LAYER_NUM]

6.83.1 Detailed Description

Response struct of MMSE MIMO.

6.83.2 Field Documentation

6.83.2.1 pEstTxSignal

```
void* bblib_mmse_mimo_response::pEstTxSignal[BBLIB_MAX_TX_LAYER_NUM][BBLIB_N_SYMB_PER_SF]
```

Data pointer points to nTx*nSymbol estimated TX signal, format 16S13

6.83.2.2 pPostSINR

```
void* bblib_mmse_mimo_response::pPostSINR[BBLIB_MAX_TX_LAYER_NUM]
```

Pointer points to nTx*1 estimated post SINR, floating number

6.84 bblib_modulation_request Struct Reference

```
#include <phy_modulation.h>
```

Data Fields

- uint8_t * data_in
- enum bblib_modulation_order mod_order
- int32_t data_len_bytes
- uint8_t n_skip

6.84.1 Detailed Description

Request structure for the modulation mapper that contains the input buffer, modulation order, length of the data and number of bits to be skipped in the input buffer.

6.84.2 Field Documentation

6.84.2.1 data_in

`uint8_t* bblib_modulation_request::data_in`

The input bits buffer. It must be 64 bytes aligned. The following bit order is assumed in the byte: [MSB] b(i)b(i+1)...b(i+7) [LSB].

6.84.2.2 data_len_bytes

`int32_t bblib_modulation_request::data_len_bytes`

The length of the input data in bytes. Data length must be $>$ 0.

6.84.2.3 mod_order

`enum bblib_modulation_order bblib_modulation_request::mod_order`

Modulation order that is applied to the input data. Supported values are 2 (QPSK), 4 (16QAM), 6 (64QAM) and 8 (256QAM).

6.84.2.4 n_skip

`uint8_t bblib_modulation_request::n_skip`

Number of bits to be skipped at the begining of the buffer. It is only used for intergation with FlexRAN LTE refPHY and affects 64QAM. It should be set to 0 in all other cases.

6.85 bblib_modulation_response Struct Reference

`#include <phy_modulation.h>`

Data Fields

- complex_int16_t * symbols_out
- int32_t num_symbols

6.85.1 Detailed Description

Response structure for the modulation mapper that contains number of output symbols and and buffer with them. The output memory buffer must be allocated by the calling function.

6.85.2 Field Documentation

6.85.2.1 num_symbols

```
int32_t bblib_modulation_response::num_symbols
```

Number of IQ symbols in the output buffer.

6.85.2.2 symbols_out

```
complex_int16_t* bblib_modulation_response::symbols_out
```

Complex output symbols buffer. It must be 64 bytes aligned. format 16S13

6.86 bblib_mul_beta_request Struct Reference

```
#include <phy_precoding_5gnr.h>
```

Data Fields

- void * data_in
- int32_t data_start
- int32_t data_len
- float beta

6.86.1 Detailed Description

beta multiplication structure (used in request).

6.86.2 Field Documentation

6.86.2.1 beta

```
float bblib_mul_beta_request::beta
```

Represent amplitude sacling factor, positive value

6.86.2.2 data_in

```
void* bblib_mul_beta_request::data_in
```

The input data sequence with any FXP format

6.86.2.3 data_len

`int32_t bblib_mul_beta_request::data_len`

The number of data to do multiplication.

6.86.2.4 data_start

`int32_t bblib_mul_beta_request::data_start`

The index for the data to start beta multiplication.

6.87 bblib_mul_beta_response Struct Reference

`#include <phy_precoding_5gnr.h>`

Data Fields

- void * data_out

6.87.1 Detailed Description

beta multiplication structure (used in response).

6.87.2 Field Documentation

6.87.2.1 data_out

`void* bblib_mul_beta_response::data_out`

The output data sequence with same input FXP format

6.88 bblib_nr_zc_sequence_gen_request Struct Reference

`#include <phy_nr_zc_sequence_gen.h>`

Data Fields

- size_t num_re
- uint8_t u
- uint8_t v
- uint8_t cyclic_shift
- uint8_t cyclic_max
- int16_t scale
- int16_t * ptr_ZcBaseSeq36PlusTable

6.88.1 Detailed Description

Structure defines the ZC sequecen generation function request interface in 5GNR.

6.88.2 Field Documentation

6.88.2.1 cyclic_max

```
uint8_t bblib_nr_zc_sequence_gen_request::cyclic_max
```

Cyclic maxsignalled from higher layers.

6.88.2.2 cyclic_shift

```
uint8_t bblib_nr_zc_sequence_gen_request::cyclic_shift
```

Cyclic shift for cell signalled from higher layers.

6.88.2.3 num_re

```
size_t bblib_nr_zc_sequence_gen_request::num_re
```

Number of resource elements in the sequence.

6.88.2.4 ptr_ZcBaseSeq36PlusTable

```
int16_t* bblib_nr_zc_sequence_gen_request::ptr_ZcBaseSeq36PlusTable
```

The Table of Zc base sequence.

6.88.2.5 scale

```
int16_t bblib_nr_zc_sequence_gen_request::scale
```

fixed point scaling factor.

6.88.2.6 u

`uint8_t bblib_nr_zc_sequence_gen_request::u`

DMRS base sequence ID in the sequence group.

6.88.2.7 v

`uint8_t bblib_nr_zc_sequence_gen_request::v`

DMRS group sequence ID.

6.89 bblib_nr_zc_sequence_gen_response Struct Reference

```
#include <phy_nr_zc_sequence_gen.h>
```

Data Fields

- complex_int16_t * ref_dmrs

6.89.1 Detailed Description

Structure defines the ZC sequecen generation function response interface in 5GNR.

6.89.2 Field Documentation

6.89.2.1 ref_dmrs

`complex_int16_t* bblib_nr_zc_sequence_gen_response::ref_dmrs`

Pointer to the reference DMRS symbols, must be cache aligned.

6.90 bblib_pbch_remapping_request Struct Reference

```
#include <phy_remapping_ctrlch.h>
```

Data Fields

- BandwidthEnum sys_bw
- int32_t cell_id
- complex_int16_t * input
- int32_t nin_len
- int32_t n_ch_reg
- int32_t * ch_seq_idx

6.90.1 Detailed Description

Request structure for PBCH remapping.

6.90.2 Field Documentation

6.90.2.1 cell_id

```
int32_t bblib_pbch_remapping_request::cell_id
```

Cell ID.

6.90.2.2 ch_seq_idx

```
int32_t* bblib_pbch_remapping_request::ch_seq_idx
```

Offset for re mapping.

6.90.2.3 input

```
complex_int16_t* bblib_pbch_remapping_request::input
```

Input after precoding.

6.90.2.4 n_ch_reg

```
int32_t bblib_pbch_remapping_request::n_ch_reg
```

REG numbers.

6.90.2.5 nin_len

```
int32_t bblib_pbch_remapping_request::nin_len
```

Length, should be = 4*nChReg.

6.90.2.6 sys_bw

`BandwidthEnum bblib_pbch_remapping_request::sys_bw`

Bandwidth.

6.91 bblib_pbch_remapping_response Struct Reference

`#include <phy_remapping_ctrlch.h>`

Data Fields

- Matrix ∗ output

6.91.1 Detailed Description

Request structure for PBCH remapping.

6.91.2 Field Documentation

6.91.2.1 output

`Matrix* bblib_pbch_remapping_response::output`

Matrix to save output.

6.92 bblib_pdcch_remapping_5gnr_request Struct Reference

`#include <phy_remapping_pdcch_5gnr.h>`

Data Fields

- uint16_t n_start_rb
- uint16_t n_end_rb
- uint8_t nNrOfSymbols
- uint8_t nCCEStart
- uint8_t nCCEPoolLen
- uint8_t coreset_cce_reg_maptype
- uint8_t coreset_reg_bundle_size
- uint8_t coreset_interleaver_size
- uint16_t coreset_shift_index
- complex_int16_t ∗ data_input
- complex_int16_t ∗ dmrs_input [BBLIB_REMAPPING_PDCCH_5GNR_MAX_SYMBOL]

6.92.1 Detailed Description

Request structure for PDCCH remapping 5gnr.

6.92.2 Field Documentation

6.92.2.1 coreset_cce_reg_maptype

`uint8_t bblib_pdcch_remapping_5gnr_request::coreset_cce_reg_maptype`

CORSET-CCE-TO-REG-mapping-type. 0: non-interleaved CCE-to-REG mapping, 1: interleaved CCE-to-REG mapping

6.92.2.2 coreset_interleaver_size

`uint8_t bblib_pdcch_remapping_5gnr_request::coreset_interleaver_size`

CORESET-interleaver-size. Value: 2, 3, 6

6.92.2.3 coreset_reg_bundle_size

`uint8_t bblib_pdcch_remapping_5gnr_request::coreset_reg_bundle_size`

CORESET-REG-bundle-size. Value: 2, 3, 6

6.92.2.4 coreset_shift_index

`uint16_t bblib_pdcch_remapping_5gnr_request::coreset_shift_index`

If it is for a PDCCH transmitted in a CORESET configured by the PBCH or SIB1, L2 need to set it to physical cell ID. Otherwise, L2 needs to set it to CORESET-shift-index.

6.92.2.5 data_input

`complex_int16_t* bblib_pdcch_remapping_5gnr_request::data_input`

PDCCH data Input after modulation

6.92.2.6 dmrs_input

`complex_int16_t* bblib_pdcch_remapping_5gnr_request::dmrs_input[BBLIB_REMAPPING_PDCCH_5GNR_MAX_SYMBOL]`

dmrs_input is an array of pointers to dmrs input buffers, the size of the array must equal bblic_remapping_pdcch←_5gnr_max_symbol

6.92.2.7 n_end_rb

`uint16_t bblib_pdcch_remapping_5gnr_request::n_end_rb`

End RB of the CORESET.

6.92.2.8 n_start_rb

`uint16_t bblib_pdcch_remapping_5gnr_request::n_start_rb`

Start RB of the CORESET.

6.92.2.9 nCCEPoolLen

`uint8_t bblib_pdcch_remapping_5gnr_request::nCCEPoolLen`

Aggregation level used: Value: 1, 2, 4, 8, 16

6.92.2.10 nCCEStart

`uint8_t bblib_pdcch_remapping_5gnr_request::nCCEStart`

CCE start index used to send the DCI:

6.92.2.11 nNrOfSymbols

`uint8_t bblib_pdcch_remapping_5gnr_request::nNrOfSymbols`

Contiguous time duration of the CORESET in number of symbols. Value: 1, 2, 3

6.93 bblib_pdcch_remapping_5gnr_response Struct Reference

`#include <phy_remapping_pdcch_5gnr.h>`

Data Fields

- complex_int16_t ∗ output [BBLIB_REMAPPING_PDCCH_5GNR_MAX_SYMBOL]

6.93.1 Detailed Description

Response structure for PDCCH remapping 5gnr.

6.93.2 Field Documentation

6.93.2.1 output

```
complex_int16_t* bblib_pdcch_remapping_5gnr_response::output[BBLIB_REMAPPING_PDCCH_5GNR_MAX_SYMBOL]
```

PDCCH after RE Mapping 5gnr output. output is an array of pointers to output buffers the size of the array will equal to nNrOfSymbol

6.94 bblib_pdcch_remapping_request Struct Reference

```
#include <phy_remapping_ctrlch.h>
```

Data Fields

- int32_t pdcch_packet
- BandwidthEnum sys_bw
- int32_t cell_id
- complex_int16_t * input
- int32_t nin_len
- int32_t n_ch_reg
- int32_t * ch_seq_idx

6.94.1 Detailed Description

Request structure for PDSCH remapping.

6.94.2 Field Documentation

6.94.2.1 cell_id

```
int32_t bblib_pdcch_remapping_request::cell_id
```

Cell ID.

6.94.2.2 ch_seq_idx

```
int32_t* bblib_pdcch_remapping_request::ch_seq_idx
```

Offset for re mapping

6.94.2.3 input

`complex_int16_t* bblib_pdcch_remapping_request::input`

Input after precoding

6.94.2.4 n_ch_reg

`int32_t bblib_pdcch_remapping_request::n_ch_reg`

REG numbers

6.94.2.5 nin_len

`int32_t bblib_pdcch_remapping_request::nin_len`

Length, should be = 4∗nChReg

6.94.2.6 pdcch_packet

`int32_t bblib_pdcch_remapping_request::pdcch_packet`

pdcch_packet: 1 - PDCCH_PACKET defined; 0 - PDCCH_PACKET not defined.

6.94.2.7 sys_bw

`BandwidthEnum bblib_pdcch_remapping_request::sys_bw`

Bandwidth.

6.95 bblib_pdcch_remapping_response Struct Reference

```
#include <phy_remapping_ctrlch.h>
```

Data Fields

- Matrix ∗ output

6.95.1 Detailed Description

Request structure for PDSCH remapping.

6.95.2 Field Documentation

6.95.2.1 output

```
Matrix* bblib_pdcch_remapping_response::output
```

Matrix to save output.

6.96 bblib_phase_noise_compensation_5gnr_request Struct Reference

```
#include <phase_noise_5gnr.h>
```

Data Fields

- int16_t * pCompenData
- complex_int16_t phaseNoise
- int32_t nSC

6.96.1 Detailed Description

Structure for input parameters in API of phase noise compensation for 5GNR.

6.96.2 Field Documentation

6.96.2.1 nSC

```
int32_t bblib_phase_noise_compensation_5gnr_request::nSC
```

number of sub-carriers

6.96.2.2 pCompenData

```
int16_t* bblib_phase_noise_compensation_5gnr_request::pCompenData
```

Input symbol for phase noise compensation

6.96.2.3 phaseNoise

`complex_int16_t bblib_phase_noise_compensation_5gnr_request::phaseNoise`

Phase noise for compensation

6.97 bblib_phase_noise_compensation_5gnr_response Struct Reference

```
#include <phase_noise_5gnr.h>
```

Data Fields

- int16_t ∗ pCompenOut

6.97.1 Detailed Description

Structure for output parameters in API of phase noise compensation for 5GNR.

6.97.2 Field Documentation

6.97.2.1 pCompenOut

`int16_t* bblib_phase_noise_compensation_5gnr_response::pCompenOut`

Output symbol for phase noise compensation

6.98 bblib_phase_noise_estimation_5gnr_request Struct Reference

```
#include <phase_noise_5gnr.h>
```

Data Fields

- void ∗ dmrsCe [BBLIB_PHASE_NOISE_MAX_RX_ANT]
- void ∗ ptrsDataPerSymbol [BBLIB_PHASE_NOISE_MAX_RX_ANT]
- void ∗ ptrsCe [BBLIB_PHASE_NOISE_MAX_RX_ANT]
- void ∗ taCompen
- int16_t ∗ ptrsSequence
- int16_t nSubCarrier
- int32_t nRx

6.98.1 Detailed Description

Structure for input parameters in API of phase noise estimation for 5GNR.

6.98.2 Field Documentation

6.98.2.1 dmrsCe

`void* bblib_phase_noise_estimation_5gnr_request::dmrsCe[BBLIB_PHASE_NOISE_MAX_RX_ANT]`

Dmrs channel estimation results

6.98.2.2 nRx

`int32_t bblib_phase_noise_estimation_5gnr_request::nRx`

Ant number of receiver

6.98.2.3 nSubCarrier

`int16_t bblib_phase_noise_estimation_5gnr_request::nSubCarrier`

number of sub-carriers for PTRS granted for one UE

6.98.2.4 ptrsCe

`void* bblib_phase_noise_estimation_5gnr_request::ptrsCe[BBLIB_PHASE_NOISE_MAX_RX_ANT]`

PTRS channel estimation results

6.98.2.5 ptrsDataPerSymbol

`void* bblib_phase_noise_estimation_5gnr_request::ptrsDataPerSymbol[BBLIB_PHASE_NOISE_MAX_RX_ANT]`

PTRS data per symbol

6.98.2.6 ptrsSequence

`int16_t* bblib_phase_noise_estimation_5gnr_request::ptrsSequence`

PTRS PN refrence Sequence

6.98.2.7 taCompen

```
void* bblib_phase_noise_estimation_5gnr_request::taCompen
```

TA compentation

6.99 bblib_phase_noise_estimation_5gnr_response Struct Reference

```
#include <phase_noise_5gnr.h>
```

Data Fields

- complex_int16_t * phaseNoise

6.99.1 Detailed Description

Structure for output parameters in API of phase noise estimation for 5GNR.

6.99.2 Field Documentation

6.99.2.1 phaseNoise

```
complex_int16_t* bblib_phase_noise_estimation_5gnr_response::phaseNoise
```

Phase noise

6.100 bblib_phy_pucch_f1_mul_omega_request Struct Reference

```
#include <phy_pucch_5gnr.h>
```

Data Fields

- int16_t n_shift_right
- uint16_t n_sym_num
- uint16_t n_td_occ_idx
- uint16_t n_fullband_sc
- int16_t n_freq_hopping
- int16_t * p_seq [BBLIB_PUCCH_SYMB_PER_SLOT]

6.100.1 Detailed Description

Request structure for PUCCH format 1 multiplication with Omega.

6.100.2 Field Documentation

6.100.2.1 n_freq_hopping

`int16_t bblib_phy_pucch_f1_mul_omega_request::n_freq_hopping`

enable or disable intra slot freq hopping

6.100.2.2 n_fullband_sc

`uint16_t bblib_phy_pucch_f1_mul_omega_request::n_fullband_sc`

number of sub-carriers in full band

6.100.2.3 n_shift_right

`int16_t bblib_phy_pucch_f1_mul_omega_request::n_shift_right`

shift right bits after multiplication

6.100.2.4 n_sym_num

`uint16_t bblib_phy_pucch_f1_mul_omega_request::n_sym_num`

Number of symbols in PUCCH format 1

6.100.2.5 n_td_occ_idx

`uint16_t bblib_phy_pucch_f1_mul_omega_request::n_td_occ_idx`

input data aligned on 64byte

6.100.2.6 p_seq

`int16_t* bblib_phy_pucch_f1_mul_omega_request::p_seq[BBLIB_PUCCH_SYMB_PER_SLOT]`

Pointer to low PAPR sequence, Q16S14 for the I/Q data

6.101 bblib_phy_pucch_f1_mul_omega_response Struct Reference

```
#include <phy_pucch_5gnr.h>
```

Data Fields

- int16_t * p_seq [BBLIB_PUCCH_SYMB_PER_SLOT]

6.101.1 Detailed Description

Response structure for PUCCH format 1 multiplication with Omega.

6.101.2 Field Documentation

6.101.2.1 p_seq

```
int16_t* bblib_phy_pucch_f1_mul_omega_response::p_seq[BBLIB_PUCCH_SYMB_PER_SLOT]
```

Pointer to output vectors, Q16S14 for the I/Q data

6.102 bblib_phy_pucch_f1_mul_payload_request Struct Reference

```
#include <phy_pucch_5gnr.h>
```

Data Fields

- int16_t n_shift_right
- uint16_t n_sym_num
- uint16_t n_payload_len
- uint8_t * p_payload
- int16_t * p_input

6.102.1 Detailed Description

Request structure for PUCCH format 1 multiplication with payload symbol.

6.102.2 Field Documentation

6.102.2.1 n_payload_len

`uint16_t bblib_phy_pucch_f1_mul_payload_request::n_payload_len`

Payload length in bits

6.102.2.2 n_shift_right

`int16_t bblib_phy_pucch_f1_mul_payload_request::n_shift_right`

shift right bits after multiplication

6.102.2.3 n_sym_num

`uint16_t bblib_phy_pucch_f1_mul_payload_request::n_sym_num`

Number of symbols in PUCCH format 1

6.102.2.4 p_input

`int16_t* bblib_phy_pucch_f1_mul_payload_request::p_input`

Pointer to input symbols including DMRS & UCI, Q16S14 for the I/Q data

6.102.2.5 p_payload

`uint8_t* bblib_phy_pucch_f1_mul_payload_request::p_payload`

Pointer to payload 1 or 2 bits

6.103 bblib_phy_pucch_f1_mul_payload_response Struct Reference

`#include <phy_pucch_5gnr.h>`

Data Fields

- int16_t * p_output

6.103.1 Detailed Description

Response structure for PUCCH format 1 multiplication with payload symbol.

6.103.2 Field Documentation

6.103.2.1 p_output

```
int16_t* bblib_phy_pucch_f1_mul_payload_response::p_output
```

Pointer to output symbols including DMRS & UCI, Q16S14 for the I/Q data

6.104 bblib_polar_decoder_5gnr_request Struct Reference

```
#include <phy_polar_decoder_5gnr.h>
```

Data Fields

- CACHE_ALIGNED uint8_t frozen_bits [k_max_codeword_size/8]
- CACHE_ALIGNED uint8_t parity_bits [k_max_codeword_size/8]
- int8_t * llr_buffer
- int order

6.104.1 Detailed Description

Request structure for Polar Decoder 5G NR that contains input LLRs, the decoder order and positions of frozen and parity bits.

6.104.2 Field Documentation

6.104.2.1 frozen_bits

```
CACHE_ALIGNED uint8_t bblib_polar_decoder_5gnr_request::frozen_bits[k_max_codeword_size/8]
```

The incoming frozen bits. There is one bit per codeword bit, and the bits are compacted to save space. The array is internally converted to std:bitset and SimdBitset class.

6.104.2.2 llr_buffer

```
int8_t* bblib_polar_decoder_5gnr_request::llr_buffer
```

Pointer to the buffer used to store the codeword 8-bit integer LLRs.

6.104.2.3 order

`int bblib_polar_decoder_5gnr_request::order`

Order of the decoder. Must be no more than k_max_order.

6.104.2.4 parity_bits

`CACHE_ALIGNED uint8_t bblib_polar_decoder_5gnr_request::parity_bits[k_max_codeword_size/8]`

This is used in the PC-CA decoder only. It represents the position of the parity bits. The array is internally converted to std:bitset and SimdBitset class.

6.105 bblib_polar_decoder_5gnr_response Struct Reference

`#include <phy_polar_decoder_5gnr.h>`

Data Fields

- CACHE_ALIGNED uint8_t codeword_lists [k_max_codeword_size]
- CACHE_ALIGNED uint8_t message_lists [k_max_codeword_size]
- CACHE_ALIGNED uint8_t compacted_final_message [1+k_max_codeword_size/8]
- CACHE_ALIGNED float metrics [k_max_list_size]
- unsigned num_msg_bits

6.105.1 Detailed Description

Response structure for Polar Decoder 5G NR that contains the decoded codewords and messages as well as the final message in the compacted form. More over message size in bits and list metrics are returned.

6.105.2 Field Documentation

6.105.2.1 codeword_lists

`CACHE_ALIGNED uint8_t bblib_polar_decoder_5gnr_response::codeword_lists[k_max_codeword_size]`

This is the original output. It is the decoded codeword, and is a transformation of the message. This output is a list of 8 best-guess codewords for a list decoder or a codewords bits each stored in a LSB of consecutive bytes.

For list decoder each output uint8_t contains one bit of the 8 list entries. So, a 64-bit codeword will contain 64 uint8_t outputs. Response.codewordLists[0] contains bit#0 of list entry#0, bit#0 of list entry#1, bit#0 of list entry#2, etc.

6.105.2.2 compacted_final_message

```
CACHE_ALIGNED uint8_t bblib_polar_decoder_5gnr_response::compacted_final_message[1+k_max_codeword_size/8]
```

Message(s) in the list is compacted into a series of uint8_t words, extracted from messageLists. If no message passes the CRC check, then no message is emitted.

For list decoder each N-bit message is contained in 8 (list-8) separate arrays of compacted uint8_t words: these form binary messages. Only one such message is emitted, and this is the one that passes the CRC checks.

6.105.2.3 message_lists

```
CACHE_ALIGNED uint8_t bblib_polar_decoder_5gnr_response::message_lists[k_max_codeword_size]
```

The actual message (including CRC) is contained within this, with the "frozen-bits" removed. This is a list of the most 8 best-guess messages for list decoder or single bits for list 1 decoder.

For list decoder each output uint8_t contains one bit of the 8 list entries. So, a N-bit message will contain N uint8_t outputs. Response.messageLists[0] contains bit#0 of list entry#0, bit#0 of list entry#1, bit#0 of list entry#2, etc.

6.105.2.4 metrics

```
CACHE_ALIGNED float bblib_polar_decoder_5gnr_response::metrics[k_max_list_size]
```

The set of output metrics, one per list.

6.105.2.5 num_msg_bits

```
unsigned bblib_polar_decoder_5gnr_response::num_msg_bits
```

The message size in bits, minus the CRC11/CRC6 bits. It is ZERO if no message passes the CRC checks

6.106 bblib_polar_encoder_5gnr_request Struct Reference

```
#include <phy_polar_encoder_5gnr.h>
```

Data Fields

- uint8_t ∗ pIn
- uint16_t ∗ pQn
- uint16_t tempBufInd
- uint16_t E
- uint16_t K
- uint16_t nPC
- uint16_t nPCWm
- uint16_t nMax
- uint16_t lil
- uint16_t lbil
- uint16_t N
- struct bblib_polar_tables ∗ polarTable

6.106.1 Detailed Description

Request structure for polar encoder.

6.106.2 Field Documentation

6.106.2.1 E

`uint16_t bblib_polar_encoder_5gnr_request::E`

Output sequence length of rate matching

6.106.2.2 Ibil

`uint16_t bblib_polar_encoder_5gnr_request::Ibil`

Parameter for interleaving in rate matching(PBCH,PDCCH=0;PUCCH=1)

6.106.2.3 Iil

`uint16_t bblib_polar_encoder_5gnr_request::Iil`

Parameter for interleaving in channel coding(PBCH,PDCCH=1;PUCCH=0)

6.106.2.4 K

`uint16_t bblib_polar_encoder_5gnr_request::K`

Input sequence length of channel coding, it's sum of payload size and CRC parity bits size

6.106.2.5 N

`uint16_t bblib_polar_encoder_5gnr_request::N`

the length of encoder bits

6.106.2.6 nMax

`uint16_t bblib_polar_encoder_5gnr_request::nMax`

Parameter for calculate the output sequence length of channel coding

6.106.2.7 nPC

`uint16_t bblib_polar_encoder_5gnr_request::nPC`

Number of parity check bits in polar information sequence

6.106.2.8 nPCWm

`uint16_t bblib_polar_encoder_5gnr_request::nPCWm`

Number of minimum row weight parity check bits in polar information sequence

6.106.2.9 pIn

`uint8_t* bblib_polar_encoder_5gnr_request::pIn`

Original input data of polar encoder, the sequence length should be K

6.106.2.10 polarTable

`struct bblib_polar_tables* bblib_polar_encoder_5gnr_request::polarTable`

constant table for polar encoder

6.106.2.11 pQn

`uint16_t* bblib_polar_encoder_5gnr_request::pQn`

polar encoder sequence

6.106.2.12 tempBufInd

`uint16_t bblib_polar_encoder_5gnr_request::tempBufInd`

index of temp buffer of size 8N byte used in encoding

6.107 bblib_polar_encoder_5gnr_response Struct Reference

```
#include <phy_polar_encoder_5gnr.h>
```

Data Fields

- uint8_t ∗ pOut
- uint16_t E

6.107.1 Detailed Description

Reponse structure for polar encoder.

6.107.2 Field Documentation

6.107.2.1 E

`uint16_t bblib_polar_encoder_5gnr_response::E`

Output sequence length of rate matching

6.107.2.2 pOut

`uint8_t* bblib_polar_encoder_5gnr_response::pOut`

Output data of polar encoder, the sequence length is E

6.108 bblib_polar_rate_dematching_5gnr_request Struct Reference

`#include <phy_polar_rate_dematching_5gnr.h>`

Data Fields

- int8_t * p_LLR
- int8_t * p_temp_buff
- int16_t E
- int16_t K
- int16_t nPC
- int16_t nPCWm
- int16_t nMax
- int16_t Iil
- int16_t Ibil
- int16_t N

6.108.1 Detailed Description

Request structure containing pointer to data and its length.

6.108.2 Field Documentation

6.108.2.1 E

`int16_t bblib_polar_rate_dematching_5gnr_request::E`

The length after rate matching.

6.108.2.2 Ibil

`int16_t bblib_polar_rate_dematching_5gnr_request::Ibil`

The value of Ibil. Parameter for interleaving in rate matching (PBCH,PDCCH=0;PUCCH=1).

6.108.2.3 Iil

`int16_t bblib_polar_rate_dematching_5gnr_request::Iil`

The value of Iil. Parameter for interleaving in channel coding (PBCH,PDCCH=1;PUCCH=0).

6.108.2.4 K

`int16_t bblib_polar_rate_dematching_5gnr_request::K`

The bit length of input information.

6.108.2.5 N

`int16_t bblib_polar_rate_dematching_5gnr_request::N`

The length of polar encoding.

6.108.2.6 nMax

`int16_t bblib_polar_rate_dematching_5gnr_request::nMax`

The maximum value to calculate N.

6.108.2.7 nPC

`int16_t bblib_polar_rate_dematching_5gnr_request::nPC`

The bit length of parity.

6.108.2.8 nPCWm

`int16_t bblib_polar_rate_dematching_5gnr_request::nPCWm`

The bit length of special parity.

6.108.2.9 p_LLR

```
int8_t* bblib_polar_rate_dematching_5gnr_request::p_LLR
```

The pointer to polar rate_dematching input data: the log-likelihood ratios (LLRs)

6.108.2.10 p_temp_buff

```
int8_t* bblib_polar_rate_dematching_5gnr_request::p_temp_buff
```

The pointer to temp buffer of size E+5N byte, allocate it outside for considering BBU pool.

6.109 bblib_polar_rate_dematching_5gnr_response Struct Reference

```
#include <phy_polar_rate_dematching_5gnr.h>
```

Data Fields

- uint8_t ∗ frozen_bits
- uint8_t ∗ parity_bits
- int8_t ∗ llr_buffer

6.109.1 Detailed Description

reponse structure containing pointer to data.

6.109.2 Field Documentation

6.109.2.1 frozen_bits

```
uint8_t* bblib_polar_rate_dematching_5gnr_response::frozen_bits
```

Pointer to buffer used to store frozen bits, one bit per codeword bit, and the bits are compacted to save space.

6.109.2.2 llr_buffer

```
int8_t* bblib_polar_rate_dematching_5gnr_response::llr_buffer
```

Pointer to the buffer used to store the de-rate matched 8-bit integer LLRs.

6.109.2.3 parity_bits

```
uint8_t* bblib_polar_rate_dematching_5gnr_response::parity_bits
```

Pointer to buffer used to store parity bits

6.110 bblib_polar_tables Struct Reference

```
#include <phy_polar_encoder_5gnr.h>
```

Data Fields

- uint16_t * pQInfoTable
- uint16_t * pQInfoSortTable
- uint16_t * pQInterleaverTable
- uint32_t * pIndexTable

6.110.1 Detailed Description

polar encoder table ptrs

6.110.2 Field Documentation

6.110.2.1 pIndexTable

```
uint32_t* bblib_polar_tables::pIndexTable
```

table for pq index

6.110.2.2 pQInfoSortTable

```
uint16_t* bblib_polar_tables::pQInfoSortTable
```

table for pq sort info

6.110.2.3 pQInfoTable

```
uint16_t* bblib_polar_tables::pQInfoTable
```

table for pq info

6.110.2.4 pQInterleaverTable

```
uint16_t* bblib_polar_tables::pQInterleaverTable
```

table for pq interleaver

6.111 bblib_prach_5gnr_detect_request Struct Reference

```
#include <phy_prach_5gnr.h>
```

Data Fields

- uint16_t n_occ
- uint16_t n_root
- int32_t noise_threshold [PRACH_MAX_OCCASION][PRACH_PREAMBLE_SEQ_NUM]
- uint16_t ifft_size
- uint16_t lra
- uint16_t nta
- uint16_t zero_corr_zone_cfg
- uint16_t logical_idx
- bool fo_check
- int32_t ∗ combine_buf [PRACH_MAX_OCCASION][PRACH_PREAMBLE_SEQ_NUM]

6.111.1 Detailed Description

Structure defining the prach_5gnr_detect input interface.

6.111.2 Field Documentation

6.111.2.1 combine_buf

```
int32_t* bblib_prach_5gnr_detect_request::combine_buf[PRACH_MAX_OCCASION][PRACH_PREAMBLE_SEQ_↩
NUM]
```

Pointer to buffer containing combined PDP 1024 or 2048 input from pipeline

6.111.2.2 fo_check

```
bool bblib_prach_5gnr_detect_request::fo_check
```

prach fo check enable/disable switch .

6.111.2.3 ifft_size

```
uint16_t bblib_prach_5gnr_detect_request::ifft_size
```

Number of points in IFFT

6.111.2.4 logical_idx

```
uint16_t bblib_prach_5gnr_detect_request::logical_idx
```

logical idx, TS 38.211 in Tables 6.3.3.1-3/4.

6.111.2.5 lra

```
uint16_t bblib_prach_5gnr_detect_request::lra
```

Preamble length, L_RA, TS 38.211 6.3.3. Valid values 139 or 839 only

6.111.2.6 n_occ

```
uint16_t bblib_prach_5gnr_detect_request::n_occ
```

num occasions

6.111.2.7 n_root

```
uint16_t bblib_prach_5gnr_detect_request::n_root
```

num root seq

6.111.2.8 noise_threshold

```
int32_t bblib_prach_5gnr_detect_request::noise_threshold[PRACH_MAX_OCCASION][PRACH_PREAMBLE_←
SEQ_NUM]
```

threshold set

6.111.2.9 nta

```
uint16_t bblib_prach_5gnr_detect_request::nta
```

Units of the reported TA given in number of TS. Calculated using equation N_TA = TA∗16∗64/2^m where m is based on sub-carrier spacing based on TS 38.211 Table 4.2.1

6.111.2.10 zero_corr_zone_cfg

```
uint16_t bblib_prach_5gnr_detect_request::zero_corr_zone_cfg
```

zeroCorrelationZoneConfig, TS 38.211 6.3.3

6.112 bblib_prach_5gnr_detect_response Struct Reference

```
#include <phy_prach_5gnr.h>
```

Data Fields

- uint16_t nReport [PRACH_MAX_OCCASION]
- uint16_t idxPreamble [PRACH_MAX_OCCASION][PRACH_PREAMBLE_SEQ_NUM]
- uint16_t value_TA [PRACH_MAX_OCCASION][PRACH_PREAMBLE_SEQ_NUM]
- int32_t power [PRACH_MAX_OCCASION][PRACH_PREAMBLE_SEQ_NUM]

6.112.1 Detailed Description

Structure defining the prach_5gnr_detect output interface.

6.112.2 Field Documentation

6.112.2.1 idxPreamble

```
uint16_t bblib_prach_5gnr_detect_response::idxPreamble[PRACH_MAX_OCCASION][PRACH_PREAMBLE_SE←
Q_NUM]
```

Idx of detected preambles for all

6.112.2.2 nReport

```
uint16_t bblib_prach_5gnr_detect_response::nReport[PRACH_MAX_OCCASION]
```

Number of detected preambles for all

6.112.2.3 power

```
int32_t bblib_prach_5gnr_detect_response::power[PRACH_MAX_OCCASION][PRACH_PREAMBLE_SEQ_NUM]
```

power of detected preambles for all

6.112.2.4 value_TA

```
uint16_t bblib_prach_5gnr_detect_response::value_TA[PRACH_MAX_OCCASION][PRACH_PREAMBLE_SEQ_N←
UM]
```

ta of detected preambles for all

6.113 bblib_prach_5gnr_threshold_request Struct Reference

```
#include <phy_prach_5gnr.h>
```

Data Fields

- uint16_t n_ant
- uint16_t ifft_size
- uint16_t n_repeat
- uint16_t zero_corr_zone_cfg
- bblib_prach_5gnr_format format
- uint16_t lra
- int16_t ∗ td_corr_in

6.113.1 Detailed Description

Structure defining the prach_5gnr_threshold input interface.

6.113.2 Field Documentation

6.113.2.1 format

```
bblib_prach_5gnr_format bblib_prach_5gnr_threshold_request::format
```

Preamble format, TS 38.211 6.3.3

6.113.2.2 ifft_size

```
uint16_t bblib_prach_5gnr_threshold_request::ifft_size
```

Number of points in IFFT

6.113.2.3 lra

```
uint16_t bblib_prach_5gnr_threshold_request::lra
```

Preamble length, L_RA, TS 38.211 6.3.3. Valid values 139 or 839 only

6.113.2.4 n_ant

```
uint16_t bblib_prach_5gnr_threshold_request::n_ant
```

Number of rx antennas

6.113.2.5 n_repeat

```
uint16_t bblib_prach_5gnr_threshold_request::n_repeat
```

Number of repeated symbols used. For lra=139, Calculated as Nu / fft_size. Nu defined in TS 38.211, Table 6.3.3.1-2; For format0, n_repeat=1

6.113.2.6 td_corr_in

```
int16_t* bblib_prach_5gnr_threshold_request::td_corr_in
```

Pointer to buffer containing complex input data. Data is the time domain transformation of the frequency domain correlation of the received signal and a single ZC root sequence.The data is presented in I Q order and 16s10 format.

The buffer contains data for all antennas and repeated symbols in the following order:

ant(0) sym(0) ... ant(0) sym(n_repeat-1) .. .. ant_(n_ant-1) sym(0) ... ant_(n_ant-1) sym(n_repeat-1)

Total buffer size is 2∗fft_size∗n_ant∗n_repeat.

Buffer should be 64-byte aligned

6.113.2.7 zero_corr_zone_cfg

```
uint16_t bblib_prach_5gnr_threshold_request::zero_corr_zone_cfg
```

zeroCorrelationZoneConfig, TS 38.211 6.3.3

6.114 bblib_prach_5gnr_threshold_response Struct Reference

```
#include <phy_prach_5gnr.h>
```

Data Fields

- int32_t noise_threshold [MAX_NOISE_THRESHOLD_PER_ROOT]
- int32_t combine_buf [PRACH_IFFT_SIZE_2048]

6.114.1 Detailed Description

Structure defining the prach_5gnr_threshold output interface.

6.114.2 Field Documentation

6.114.2.1 combine_buf

```
int32_t bblib_prach_5gnr_threshold_response::combine_buf[PRACH_IFFT_SIZE_2048]
```

combine buffer

6.114.2.2 noise_threshold

```
int32_t bblib_prach_5gnr_threshold_response::noise_threshold[MAX_NOISE_THRESHOLD_PER_ROOT]
```

element of thrSet[occ][root]

6.115 bblib_prach_5gnr_threshold_uplift_request Struct Reference

```
#include <phy_prach_5gnr.h>
```

Data Fields

- uint16_t n_occ
- uint16_t n_root
- uint32_t min_noise_threshold
- int32_t noise_threshold [PRACH_MAX_OCCASION][PRACH_PREAMBLE_SEQ_NUM]

6.115.1 Detailed Description

Structure defining the prach_5gnr_threshold_uplift input interface.

6.115.2 Field Documentation

6.115.2.1 min_noise_threshold

```
uint32_t bblib_prach_5gnr_threshold_uplift_request::min_noise_threshold
```

If non-zero, this parameter is compared with uplifted noise_thresold and if smaller, uplifted value is set to this. It helps remove false prach detects.

6.115.2.2 n_occc

```
uint16_t bblib_prach_5gnr_threshold_uplift_request::n_occ
```

number of occasions

6.115.2.3 n_root

```
uint16_t bblib_prach_5gnr_threshold_uplift_request::n_root
```

Number of root seq

6.115.2.4 noise_threshold

```
int32_t bblib_prach_5gnr_threshold_uplift_request::noise_threshold[PRACH_MAX_OCCASION][PRACH←
_PREAMBLE_SEQ_NUM]
```

noise threshold set buffer

6.116 bblib_prach_5gnr_threshold_uplift_response Struct Reference

```
#include <phy_prach_5gnr.h>
```

Data Fields

- int32_t noise_threshold [PRACH_MAX_OCCASION][PRACH_PREAMBLE_SEQ_NUM]

6.116.1 Detailed Description

Structure defining the prach_5gnr_threshold_uplift output interface.

6.116.2 Field Documentation

6.116.2.1 noise_threshold

```
int32_t bblib_prach_5gnr_threshold_uplift_response::noise_threshold[PRACH_MAX_OCCASION][PRAC←
H_PREAMBLE_SEQ_NUM]
```

noise threshold set buffer

6.117 bblib_prach_5gnr_zc_gen_request Struct Reference

```
#include <phy_prach_5gnr.h>
```

Data Fields

- uint16_t logical_idx
- uint16_t lra

6.117.1 Detailed Description

Structure defining the prach_5gnr_zc_gen input interface.

6.117.2 Field Documentation

6.117.2.1 logical_idx

```
uint16_t bblib_prach_5gnr_zc_gen_request::logical_idx
```

Logical index i. The logical index is used to obtain sequence number according to TS 38.211 Tables 6.3.3.1-3/4.

6.117.2.2 lra

```
uint16_t bblib_prach_5gnr_zc_gen_request::lra
```

Preamble sequence length L_RA, TS 38.211 6.3.3. Valid values are 139 or 839 only

6.118 bblib_prach_5gnr_zc_gen_response Struct Reference

```
#include <phy_prach_5gnr.h>
```

Data Fields

- int16_t * fd_zc_seq

6.118.1 Detailed Description

Structure defining the prach_5gnr_zc_gen output interface.

6.118.2 Field Documentation

6.118.2.1 fd_zc_seq

`int16_t* bblib_prach_5gnr_zc_gen_response::fd_zc_seq`

Pointer to output buffer of complex zadoff-chu sequence in I Q order and 16s15 format. Buffer should be 64-byte aligned

6.119 bblib_prbs_request Struct Reference

`#include <pseudo_random_seq_gen.h>`

Data Fields

- uint32_t c_init
- uint16_t num_bits
- uint16_t gold_code_advance

6.119.1 Detailed Description

Request structure for Pseudo random sequence generation.

6.119.2 Field Documentation

6.119.2.1 c_init

`uint32_t bblib_prbs_request::c_init`

cinit value- see TS38.211 5.2 for more details

6.119.2.2 gold_code_advance

`uint16_t bblib_prbs_request::gold_code_advance`

Gold code- Fixed for 4G and NR at 1600

6.119.2.3 num_bits

```
uint16_t bblib_prbs_request::num_bits
```

Number of bits to output

6.120 bblib_prbs_response Struct Reference

```
#include <pseudo_random_seq_gen.h>
```

Data Fields

- uint16_t num_bits
- uint8_t * bits

6.120.1 Detailed Description

Response structure for Pseudo random sequence generation.

6.120.2 Field Documentation

6.120.2.1 bits

```
uint8_t* bblib_prbs_response::bits
```

Output bit sequence of size num_bits (as defined in request), should be 64 byte aligned

6.120.2.2 num_bits

```
uint16_t bblib_prbs_response::num_bits
```

Number of bits in the output sequence

6.121 bblib_precoding_5gnr_antennas Struct Reference

```
#include <phy_precoding_5gnr.h>
```

Data Fields

- uint32_t num_values
- void * values

6.121.1 Detailed Description

Antennas structure (used in response).

The output array of antenna data. The number of antennas will be equal to the number of antennas specified in the precoding parameter. The antennas are passed in as an array of pointers to antennas since each antenna may be in a different message in a different region of memory to the other antennas.

6.121.2 Field Documentation

6.121.2.1 num_values

```
uint32_t bblib_precoding_5gnr_antennas::num_values
```

The number of IQ values.

6.121.2.2 values

```
void* bblib_precoding_5gnr_antennas::values
```

The antenna data (must be 64 byte aligned). In floating point mode array of complex floating point numbers (complex_float∗). In mixed mode: array of complex 16b fixed point numbers with same format of input. In full fixed point mode: array of complex 16b fixed point numbers with same format of input

6.122 bblib_precoding_5gnr_layers Struct Reference

```
#include <phy_precoding_5gnr.h>
```

Data Fields

- uint32_t num_values
- void ∗ values

6.122.1 Detailed Description

Layers structure (used in request).

The input array of sub-carrier layers. The number of layers will be equal to the number of layers specified in the precoding parameter. Note that an array of pointers to layers is required since the layers may come from different messages which are stored in different memory regions.

6.122.2 Field Documentation

6.122.2.1 num_values

```
uint32_t bblib_precoding_5gnr_layers::num_values
```

The number of IQ values.

6.122.2.2 values

```
void* bblib_precoding_5gnr_layers::values
```

The layers data (must be 64 byte aligned). In floating point mode: array of half precision floating point numbers (complex_half∗). In mixed mode: array of any complex 16b fixed point numbers. In full fixed point: array of any complex 16b fixed point numbers

6.123 bblib_precoding_5gnr_precoding Struct Reference

```
#include <phy_precoding_5gnr.h>
```

Data Fields

- void ∗ data
- uint32_t m_num_antennas
- uint32_t m_num_layers

6.123.1 Detailed Description

Precoding matrix structure (used in request).

This defines the beam-formed precoding data to use.

6.123.2 Field Documentation

6.123.2.1 data

```
void* bblib_precoding_5gnr_precoding::data
```

The precoding data stored row-major (must be 64 byte aligned). In floating point mode: array of complex floating point numbers (complex_float∗). In mixed mode and full fixed point mode: array of complex 16b fixed point numbers Q16s15 [-1...+0.9999] (complex_int16_t ∗).

6.123.2.2 m_num_antennas

```
uint32_t bblib_precoding_5gnr_precoding::m_num_antennas
```

The number of rows in the precode matrix. Valid values of number of antennas are: 2, 4, 8, 16.

6.123.2.3 m_num_layers

```
uint32_t bblib_precoding_5gnr_precoding::m_num_layers
```

The number of columns in the precode matrix. Valid values of number of layers for given number of antennas are:

- 1, 2 (m_num_antennas = 2);
- 1, 2, 4 (m_num_antennas = 4)
- 1, 2, 4, 8 (m_num_antennas = 8)
- 1, 8 (m_num_antennas = 16).

6.124 bblib_precoding_5gnr_request Struct Reference

```
#include <phy_precoding_5gnr.h>
```

Data Fields

- struct bblib_precoding_5gnr_precoding * precoding
- struct bblib_precoding_5gnr_layers ** layers
- precoding_mode mode

6.124.1 Detailed Description

Request structure.

6.124.2 Field Documentation

6.124.2.1 layers

```
struct bblib_precoding_5gnr_layers** bblib_precoding_5gnr_request::layers
```

Layers input data structures.

6.124.2.2 mode

`precoding_mode bblib_precoding_5gnr_request::mode`

Precoding mode:

- floating point mode: half-precision layer data, all other data single precision floating point numbers;
- mixed mode: fixed point input and output data with floating point calculations;
- full fixed point mode with 16b accumulators: fixed point input and output with fixed point calculations; faster than mixed mode but less accurate due to the use of a 16 bit accumulation.

6.124.2.3 precoding

`struct bblib_precoding_5gnr_precoding* bblib_precoding_5gnr_request::precoding`

Precoding input data structure.

6.125 bblib_precoding_5gnr_response Struct Reference

`#include <phy_precoding_5gnr.h>`

Data Fields

- struct bblib_precoding_5gnr_antennas ** antennas

6.125.1 Detailed Description

Response structure.

6.125.2 Field Documentation

6.125.2.1 antennas

`struct bblib_precoding_5gnr_antennas** bblib_precoding_5gnr_response::antennas`

Antennas input data structures.

6.126 bblib_precoding_codebook_fetch_5gnr_request Struct Reference

```
#include <phy_precoding_5gnr.h>
```

Data Fields

- enum bblib_precoding_cb_type codebook_type
- enum bblib_precoding_cb_mode codebook_mode
- enum bblib_precoding_trans_scheme nTransmissionScheme
- uint16_t n1_n2
- uint8_t num_layer
- uint8_t num_ant
- uint8_t pmi

6.126.1 Detailed Description

Request structure for 5g DL precoding codebook fetch function.

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

6.126.2 Field Documentation

6.126.2.1 codebook_mode

```
enum bblib_precoding_cb_mode bblib_precoding_codebook_fetch_5gnr_request::codebook_mode
```

codebook mode according to 5.2.2.2 in 38.214

6.126.2.2 codebook_type

```
enum bblib_precoding_cb_type bblib_precoding_codebook_fetch_5gnr_request::codebook_type
```

codebook type according to 5.2.2.2 in 38.214

6.126.2.3 n1_n2

```
uint16_t bblib_precoding_codebook_fetch_5gnr_request::n1_n2
```

n1<<8+n2

6.126.2.4 nTransmissionScheme

```
enum bblib_precoding_trans_scheme bblib_precoding_codebook_fetch_5gnr_request::nTransmission↩
Scheme
```

Transmission scheme

6.126.2.5 num_ant

```
uint8_t bblib_precoding_codebook_fetch_5gnr_request::num_ant
```

The antenna port number for one UE

6.126.2.6 num_layer

```
uint8_t bblib_precoding_codebook_fetch_5gnr_request::num_layer
```

The layer number for one UE

6.126.2.7 pmi

```
uint8_t bblib_precoding_codebook_fetch_5gnr_request::pmi
```

pmi according to 5.2.2.2 in 38.214

6.127 bblib_precoding_codebook_fetch_5gnr_response Struct Reference

```
#include <phy_precoding_5gnr.h>
```

Data Fields

- struct bblib_precoding_5gnr_precoding ∗ precoding

6.127.1 Detailed Description

Response structure for 5g DL precoding codebook fetch function.

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

6.127.2 Field Documentation

6.127.2.1 precoding

```
struct bblib_precoding_5gnr_precoding* bblib_precoding_codebook_fetch_5gnr_response::precoding
```

Precoding matrix data structure.

6.128 bblib_precoding_request Struct Reference

```
#include <phy_precoding.h>
```

Data Fields

- int8_t transmode
- int8_t cddtype
- int32_t m0_symbol_no
- int32_t m1_symbol_no
- int8_t ant_no
- int8_t codeword_no
- int8_t layer_no
- int8_t index
- int16_t * data_in

6.128.1 Detailed Description

Request structure for precoding.

6.128.2 Field Documentation

6.128.2.1 ant_no

```
int8_t bblib_precoding_request::ant_no
```

The number of tx ants.

6.128.2.2 cddtype

```
int8_t bblib_precoding_request::cddtype
```

The preocoding type, value 0/1, 0:without CDD, 1:large delay CDD.

6.128.2.3 codeword_no

`int8_t bblib_precoding_request::codeword_no`

The number of code words.

6.128.2.4 data_in

`int16_t* bblib_precoding_request::data_in`

The input data.

6.128.2.5 index

`int8_t bblib_precoding_request::index`

Codebook index.

6.128.2.6 layer_no

`int8_t bblib_precoding_request::layer_no`

The number of layers.

6.128.2.7 m0_symbol_no

`int32_t bblib_precoding_request::m0_symbol_no`

Length of input symbols for code word0.

6.128.2.8 m1_symbol_no

`int32_t bblib_precoding_request::m1_symbol_no`

length of input symbols for code word1.

6.128.2.9 transmode

`int8_t bblib_precoding_request::transmode`

The preocoding type, value 0/1, 0:Transmit diversity, 1:Spatial multiplexing.

6.129 bblib_precoding_response Struct Reference

```
#include <phy_precoding.h>
```

Data Fields

- int16_t * data_out

6.129.1 Detailed Description

Response structure with output data.

6.129.2 Field Documentation

6.129.2.1 data_out

```
int16_t* bblib_precoding_response::data_out
```

The output data.

6.130 bblib_pucch_cestimate_5gnr_dmrs_request Struct Reference

```
#include <phy_pucch_cestimate_5gnr.h>
```

Data Fields

- enum cestimate_pucch_5gnr_phy_formats pucch_format
- uint16_t prb_num
- uint32_t slot_num
- uint16_t dmrs_scram_id
- uint16_t start_sym
- uint16_t sym_num

6.130.1 Detailed Description

Request structure for 5GNR PUCCH chan estimator DMRS generator.

6.130.2 Field Documentation

6.130.2.1 dmrs_scram_id

```
uint16_t bblib_pucch_cestimate_5gnr_dmrs_request::dmrs_scram_id
```

Ts211 6.4.1.3.2.1 Scrambling-ID

6.130.2.2 prb_num

```
uint16_t bblib_pucch_cestimate_5gnr_dmrs_request::prb_num
```

Number of PUCCH PRB spanned by DMRS sequence

6.130.2.3 pucch_format

```
enum cestimate_pucch_5gnr_phy_formats bblib_pucch_cestimate_5gnr_dmrs_request::pucch_format
```

PUCCH Format

6.130.2.4 slot_num

```
uint32_t bblib_pucch_cestimate_5gnr_dmrs_request::slot_num
```

Slot index within a frame

6.130.2.5 start_sym

```
uint16_t bblib_pucch_cestimate_5gnr_dmrs_request::start_sym
```

Starting symbol

6.130.2.6 sym_num

```
uint16_t bblib_pucch_cestimate_5gnr_dmrs_request::sym_num
```

Symbol number: 1∼2

6.131 bblib_pucch_cestimate_5gnr_dmrs_response Struct Reference

```
#include <phy_pucch_cestimate_5gnr.h>
```

Data Fields

- complex_int16_t ∗ dmrs_sym [BBLIB_N_SYMB_PER_SF]

6.131.1 Detailed Description

Response structure for 5GNR PUCCH chan estimator DMRS generator.

6.131.2 Field Documentation

6.131.2.1 dmrs_sym

`complex_int16_t* bblib_pucch_cestimate_5gnr_dmrs_response::dmrs_sym[BBLIB_N_SYMB_PER_SF]`

Array of pointers to DMRS buffers, in fixed point Q16S14 format, number of entries in the array must equal sym_num

6.132 bblib_pucch_cestimate_5gnr_request Struct Reference

`#include <phy_pucch_cestimate_5gnr.h>`

Data Fields

- enum cestimate_pucch_5gnr_phy_formats pucch_format
- uint8_t num_rx_ants
- complex_int16_t ∗ data [BBLIB_MAX_PUCCH_RX_AP_NUM][BBLIB_N_SYMB_PER_SF]
- uint32_t slot_num
- uint16_t dmrs_scram_id
- uint16_t(∗ start_prb_num)[BBLIB_N_SYMB_PER_SF]
- uint16_t prb_num
- uint16_t sym_num
- int16_t n_mu
- uint16_t n_fft_size
- uint16_t additional_dmrs
- int16_t n_freq_hopping
- complex_int16_t ∗ dmrs_sym [BBLIB_N_SYMB_PER_SF]

6.132.1 Detailed Description

Request structure for 5GNR PUCCH channel estimator.

6.132.2 Field Documentation

6.132.2.1 additional_dmrs

`uint16_t bblib_pucch_cestimate_5gnr_request::additional_dmrs`

additional DMRS or not. 1: additional dmrs. 0: no addtional dmrs

6.132.2.2 data

`complex_int16_t* bblib_pucch_cestimate_5gnr_request::data[BBLIB_MAX_PUCCH_RX_AP_NUM][BBLIB_N_SYMB_PER_SF]`

Rx antennas data, format is I,Q,I,Q..., in fixed point Q16S13 format, number of entries in the array is num_rx_ants by sym_num

6.132.2.3 dmrs_scram_id

`uint16_t bblib_pucch_cestimate_5gnr_request::dmrs_scram_id`

Ts211 6.4.1.3.2.1 Scrambling-ID

6.132.2.4 dmrs_sym

`complex_int16_t* bblib_pucch_cestimate_5gnr_request::dmrs_sym[BBLIB_N_SYMB_PER_SF]`

Array of pointers to DMRS buffers, in fixed point Q16S14 format, number of entries in the array equal to number of symbol that DMRS exist in

6.132.2.5 n_fft_size

`uint16_t bblib_pucch_cestimate_5gnr_request::n_fft_size`

FFT size

6.132.2.6 n_freq_hopping

`int16_t bblib_pucch_cestimate_5gnr_request::n_freq_hopping`

intra slot freq hopping or not. 1: intra slot freq hopping. 0: no intra slot freq hopping

6.132.2.7 n_mu

`int16_t bblib_pucch_cestimate_5gnr_request::n_mu`

Numerology, determine sub carrier spacing, Value: 0->4

6.132.2.8 num_rx_ants

`uint8_t bblib_pucch_cestimate_5gnr_request::num_rx_ants`

Number of Rx antennas: 2 or 4

6.132.2.9 prb_num

```
uint16_t bblib_pucch_cestimate_5gnr_request::prb_num
```

PRB number: 1∼16

6.132.2.10 pucch_format

```
enum cestimate_pucch_5gnr_phy_formats bblib_pucch_cestimate_5gnr_request::pucch_format
```

PUCCH Format

6.132.2.11 slot_num

```
uint32_t bblib_pucch_cestimate_5gnr_request::slot_num
```

Slot index within a frame

6.132.2.12 start_prb_num

```
uint16_t(* bblib_pucch_cestimate_5gnr_request::start_prb_num)[BBLIB_N_SYMB_PER_SF]
```

Starting PRB index, pointer to an array of size BBLIB_N_SYMB_PER_SF

6.132.2.13 sym_num

```
uint16_t bblib_pucch_cestimate_5gnr_request::sym_num
```

Symbol number: 1∼14

6.133 bblib_pucch_cestimate_5gnr_response Struct Reference

```
#include <phy_pucch_cestimate_5gnr.h>
```

Data Fields

- complex_int16_t * ls [BBLIB_MAX_PUCCH_RX_AP_NUM][BBLIB_N_SYMB_PER_SF]
- complex_int16_t * ce [BBLIB_MAX_PUCCH_RX_AP_NUM][BBLIB_N_SYMB_PER_SF]
- complex_int16_t * ce_ta_comp [BBLIB_N_SYMB_PER_SF]
- int16_t n_est_ta
- uint16_t ls_sym_num
- uint16_t ce_sym_num
- float est_sigma2
- float est_SNRdB

6.133.1 Detailed Description

Response structure for 5GNR PUCCH chan estimator.

6.133.2 Field Documentation

6.133.2.1 ce

`complex_int16_t* bblib_pucch_cestimate_5gnr_response::ce[BBLIB_MAX_PUCCH_RX_AP_NUM][BBLIB_N_SYMB_PER_SF]`

Data points to CE output, in fixed point Q16S13 format, number of entries in the array is num_rx_ants by ce_sym←
_num

6.133.2.2 ce_sym_num

`uint16_t bblib_pucch_cestimate_5gnr_response::ce_sym_num`

symbol number of ce output

6.133.2.3 ce_ta_comp

`complex_int16_t* bblib_pucch_cestimate_5gnr_response::ce_ta_comp[BBLIB_N_SYMB_PER_SF]`

Data points to TA compensation vector, in fixed point Q16S15 format, number of entries in the array is num_rx_ants by ce_sym_num

6.133.2.4 est_sigma2

`float bblib_pucch_cestimate_5gnr_response::est_sigma2`

Estimated noise power

6.133.2.5 est_SNRdB

`float bblib_pucch_cestimate_5gnr_response::est_SNRdB`

Estimated SNR in dB

6.133.2.6 ls

`complex_int16_t* bblib_pucch_cestimate_5gnr_response::ls[BBLIB_MAX_PUCCH_RX_AP_NUM][BBLIB_N_SYMB_PER_SF]`

Data points to CE Ls, in fixed point Q16S12 format, number of entries in the array is num_rx_ants by ls_sym_num

6.133.2.7 ls_sym_num

```
uint16_t bblib_pucch_cestimate_5gnr_response::ls_sym_num
```

symbol number of ls output

6.133.2.8 n_est_ta

```
int16_t bblib_pucch_cestimate_5gnr_response::n_est_ta
```

Estimated TA value

6.134 bblib_pucch_equ_5gnr_request Struct Reference

```
#include <phy_pucch_equ_5gnr.h>
```

Data Fields

- enum equ_pucch_5gnr_phy_formats pucch_format
- uint8_t method
- uint8_t rxAntNum
- uint16_t startPRB [BBLIB_N_SYMB_PER_SF]
- uint16_t prbNum
- uint16_t symNum
- float nSigma2
- complex_int16_t ∗ data [MAX_N_ANT][BBLIB_N_SYMB_PER_SF]
- complex_int16_t ∗ chEst [MAX_N_ANT][BBLIB_N_SYMB_PER_SF]

6.134.1 Detailed Description

Request structure for PUCCH Equalization.

6.134.2 Field Documentation

6.134.2.1 chEst

```
complex_int16_t* bblib_pucch_equ_5gnr_request::chEst[MAX_N_ANT][BBLIB_N_SYMB_PER_SF]
```

Pointer to PUCCH channel estimate results. Numbers are stored as IQ values in fixed point Q16s13 format. Buffer should be 64 byte aligned.

6.134.2.2 data

`complex_int16_t* bblib_pucch_equ_5gnr_request::data[MAX_N_ANT][BBLIB_N_SYMB_PER_SF]`

Rx Antennas value. Numbers are stored as IQ values in fixed point Q16s13 format. Buffer should be 64 byte aligned.

6.134.2.3 method

`uint8_t bblib_pucch_equ_5gnr_request::method`

0: MRC; 1: MMSE;

6.134.2.4 nSigma2

`float bblib_pucch_equ_5gnr_request::nSigma2`

Noise power, used in MMSE method

6.134.2.5 prbNum

`uint16_t bblib_pucch_equ_5gnr_request::prbNum`

PRB number - valid values [1-16].

6.134.2.6 pucch_format

`enum equ_pucch_5gnr_phy_formats bblib_pucch_equ_5gnr_request::pucch_format`

PUCCH Format

6.134.2.7 rxAntNum

`uint8_t bblib_pucch_equ_5gnr_request::rxAntNum`

Number of RX antennas - valid values [2 or 4].

6.134.2.8 startPRB

`uint16_t bblib_pucch_equ_5gnr_request::startPRB[BBLIB_N_SYMB_PER_SF]`

Starting PRB Index.

6.134.2.9 symNum

```
uint16_t bblib_pucch_equ_5gnr_request::symNum
```

Data Symbol Number - valid values [1-BBLIB_N_SYMB_PER_SF].

6.135 bblib_pucch_equ_5gnr_response Struct Reference

```
#include <phy_pucch_equ_5gnr.h>
```

Data Fields

- complex_int16_t * mimoOut [BBLIB_N_SYMB_PER_SF]
- float * pPostSINR [BBLIB_N_SYMB_PER_SF]
- uint32_t numSamples [BBLIB_N_SYMB_PER_SF]

6.135.1 Detailed Description

Response structure for PUCCH Equalization.

6.135.2 Field Documentation

6.135.2.1 mimoOut

```
complex_int16_t* bblib_pucch_equ_5gnr_response::mimoOut[BBLIB_N_SYMB_PER_SF]
```

Pointer to the output data. Numbers are stored as IQ inputs in fixed point Q16s13 format. Buffer should be 64 byte aligned and be of size prbNum*BBLIB_N_SC_PER_PRB.

6.135.2.2 numSamples

```
uint32_t bblib_pucch_equ_5gnr_response::numSamples[BBLIB_N_SYMB_PER_SF]
```

Number of IQ samples stored in mimOut.

6.135.2.3 pPostSINR

```
float* bblib_pucch_equ_5gnr_response::pPostSINR[BBLIB_N_SYMB_PER_SF]
```

Pointer points to estimated post SINR. Buffer should be of size prbNum*BBLIB_N_SC_PER_PRB.

6.136 bblib_pucch_f0_detect_request Struct Reference

```
#include <phy_pucch_5gnr.h>
```

Data Fields

- uint16_t payload_len
- int16_t * seq [BBLIB_PUCCH_SYMB_PER_SLOT]
- complex_int16_t * data [BBLIB_MAX_PUCCH_F0_RX_AP_NUM][BBLIB_PUCCH_SYMB_PER_SLOT]
- uint8_t num_rx_ant
- uint8_t m0
- bool sr_check
- uint32_t alg_sel
- uint16_t start_symbol_num
- uint16_t num_symb
- enum bblib_pucch_fullband_sc n_fullband_sc
- uint16_t * prb_idx
- uint32_t num_seq
- uint32_t num_cand
- uint8_t * seq_id
- uint32_t freq_hop
- uint32_t common_noise

6.136.1 Detailed Description

Request structure for PUCCH format 0 detection.

6.136.2 Field Documentation

6.136.2.1 alg_sel

```
uint32_t bblib_pucch_f0_detect_request::alg_sel
```

To switch between current and new algorithms

6.136.2.2 common_noise

```
uint32_t bblib_pucch_f0_detect_request::common_noise
```

For new algorithm: use common noise estimation across all antennas

6.136.2.3 data

`complex_int16_t* bblib_pucch_f0_detect_request::data[BBLIB_MAX_PUCCH_F0_RX_AP_NUM][BBLIB_PUCCH_SYMB_PER_SLOT]`

Rx Antennas value. Numbers are stored as IQ values in fixed point Q16s13 format. Buffer should be 64 byte aligned.

6.136.2.4 freq_hop

`uint32_t bblib_pucch_f0_detect_request::freq_hop`

For new algorithm: intra-slot frequency hopping

6.136.2.5 m0

`uint8_t bblib_pucch_f0_detect_request::m0`

m0 assigned to UE, can be 0∼11

6.136.2.6 n_fullband_sc

`enum bblib_pucch_fullband_sc bblib_pucch_f0_detect_request::n_fullband_sc`

Number of subcarriers in full band, equal to nRB∗12, where nRB is the total resource blocks defined in 3GPP TS 38.104 V15.2.0 Table 5.3.2-1 for FR1 and Table 5.3.2-2 for FR2. This parameter is used as offset into buffer seq

6.136.2.7 num_cand

`uint32_t bblib_pucch_f0_detect_request::num_cand`

For new algorithm: number of candidate sequences

6.136.2.8 num_rx_ant

`uint8_t bblib_pucch_f0_detect_request::num_rx_ant`

number of receiver antennas, can be 1 or 2

6.136.2.9 num_seq

`uint32_t bblib_pucch_f0_detect_request::num_seq`

For new algorithm: total number of sequences to process (cands + noise)

6.136.2.10 num_symb

`uint16_t bblib_pucch_f0_detect_request::num_symb`

number of symbols

6.136.2.11 payload_len

`uint16_t bblib_pucch_f0_detect_request::payload_len`

expected payload length 1 or 2 bits

6.136.2.12 prb_idx

`uint16_t* bblib_pucch_f0_detect_request::prb_idx`

array containing the prb index for each symbol. Array size must equal num_symb

6.136.2.13 seq

`int16_t* bblib_pucch_f0_detect_request::seq[BBLIB_PUCCH_SYMB_PER_SLOT]`

the low papr sequences bblib_pucch_low_papr_seq_gen_5gnr in fixed point Q16s13 format

6.136.2.14 seq_id

`uint8_t* bblib_pucch_f0_detect_request::seq_id`

For new algorithm: array with sequence indexes for this UE (maximum 12)

6.136.2.15 sr_check

`bool bblib_pucch_f0_detect_request::sr_check`

if set to true, the function will check for scheduling request this determines the number of cycle shifts

6.136.2.16 start_symbol_num

`uint16_t bblib_pucch_f0_detect_request::start_symbol_num`

starting symbol number, allow offset

6.137 bblib_pucch_f0_detect_response Struct Reference

```
#include <phy_pucch_5gnr.h>
```

Data Fields

- uint32_t * corr_power
- bool dtx
- bool sr_present
- uint8_t payload

6.137.1 Detailed Description

Response structure for PUCCH format 0 detection.

6.137.2 Field Documentation

6.137.2.1 corr_power

```
uint32_t* bblib_pucch_f0_detect_response::corr_power
```

array of size max cyclic shift length (8) that will contain the calculated correlation power

6.137.2.2 dtx

```
bool bblib_pucch_f0_detect_response::dtx
```

True if dtx is present

6.137.2.3 payload

```
uint8_t bblib_pucch_f0_detect_response::payload
```

decoded pay load

6.137.2.4 sr_present

```
bool bblib_pucch_f0_detect_response::sr_present
```

True if scheduling request is present

6.138 bblib_pucch_fo_compensation_5gnr_request Struct Reference

```
#include <phy_pucch_focompensation_5gnr.h>
```

Data Fields

- float f_offset_angel_est
- int16_t n_rxant
- int16_t cp_size1
- int16_t cp_size
- int16_t n_prb
- uint16_t(∗ start_prb_num)[BBLIB_N_SYMB_PER_SF]
- int16_t n_fft_size
- int16_t n_comp_symb
- int16_t ∗ p_foc_in [FOC_MAX_RX_ANT_NUM][BBLIB_N_SYMB_PER_SF]

6.138.1 Detailed Description

Structure defines the Frequency Offset Compensation request in 5GNR.

6.138.2 Field Documentation

6.138.2.1 cp_size

`int16_t bblib_pucch_fo_compensation_5gnr_request::cp_size`

Normal CP Size

6.138.2.2 cp_size1

`int16_t bblib_pucch_fo_compensation_5gnr_request::cp_size1`

CP Size of first symbol

6.138.2.3 f_offset_angel_est

`float bblib_pucch_fo_compensation_5gnr_request::f_offset_angel_est`

Estimated frequency offset angel

6.138.2.4 n_comp_symb

`int16_t bblib_pucch_fo_compensation_5gnr_request::n_comp_symb`

Number of compensation symbols

6.138.2.5 n_fft_size

`int16_t bblib_pucch_fo_compensation_5gnr_request::n_fft_size`

FFT size

6.138.2.6 n_prb

`int16_t bblib_pucch_fo_compensation_5gnr_request::n_prb`

Number of PRBs

6.138.2.7 n_rxant

`int16_t bblib_pucch_fo_compensation_5gnr_request::n_rxant`

Number of Rx antennas

6.138.2.8 p_foc_in

`int16_t* bblib_pucch_fo_compensation_5gnr_request::p_foc_in[FOC_MAX_RX_ANT_NUM][BBLIB_N_SYMB_PER_SF]`

Data pointer points to nRx∗nSymbol frequency offset compensation input buffer

6.138.2.9 start_prb_num

`uint16_t(* bblib_pucch_fo_compensation_5gnr_request::start_prb_num)[BBLIB_N_SYMB_PER_SF]`

Starting PRB index, pointer to an array of size BBLIB_N_SYMB_PER_SF

6.139 bblib_pucch_fo_compensation_5gnr_response Struct Reference

`#include <phy_pucch_focompensation_5gnr.h>`

Data Fields

- int16_t ∗ p_comp_out [FOC_MAX_RX_ANT_NUM][BBLIB_N_SYMB_PER_SF]

6.139.1 Detailed Description

Structure defines the Frequency Offset Compensation response in 5GNR.

6.139.2 Field Documentation

6.139.2.1 p_comp_out

`int16_t* bblib_pucch_fo_compensation_5gnr_response::p_comp_out[FOC_MAX_RX_ANT_NUM][BBLIB_N_SYMB_PER_SF]`

Data pointer points to nRx∗nSymbol Frequency offset compensation output buffer

6.140 bblib_pucch_low_papr_request Struct Reference

`#include <phy_pucch_5gnr.h>`

Data Fields

- uint16_t seq_length
- double fAlpha
- uint16_t u
- uint16_t v
- uint16_t scale

6.140.1 Detailed Description

Request structure for PUCCH low papr sequence generation.

6.140.2 Field Documentation

6.140.2.1 fAlpha

`double bblib_pucch_low_papr_request::fAlpha`

f Alpha as per 38.211 6.3.2.2.2

6.140.2.2 scale

`uint16_t bblib_pucch_low_papr_request::scale`

Scaling factor for fxp output, to multiply flp sin/cos function. Value 14 for Q16s14 output

6.140.2.3 seq_length

```
uint16_t bblib_pucch_low_papr_request::seq_length
```

Length of the sequence

6.140.2.4 u

```
uint16_t bblib_pucch_low_papr_request::u
```

as per 38.211 6.3.2.3.1

6.140.2.5 v

```
uint16_t bblib_pucch_low_papr_request::v
```

as per 38.211 6.3.2.3.1

6.141 bblib_pucch_low_papr_response Struct Reference

```
#include <phy_pucch_5gnr.h>
```

Data Fields

- int16_t * seq

6.141.1 Detailed Description

Response structure for PUCCH low papr sequence generation.

6.141.2 Field Documentation

6.141.2.1 seq

```
int16_t* bblib_pucch_low_papr_response::seq
```

64byte aligned output buffer that will contain the sequence generated. The size of the buffer must be >= seq_length. Each element in the sequence is a complex int16_t, It is packed in I0/Q0/I1/Q1......, need to allocate seq_length*4 bytes for *seq.

6.142 bblib_pucch_ndash_request Struct Reference

```
#include <phy_cestimate_pucch.h>
```

Data Fields

- int16_t cyclic_shift
- int16_t delta_shift
- int16_t bandwidth_rb
- int16_t * res_indices [4]
- uint32_t * pucch_index
- uint16_t num_pucch
- uint8_t num_res
- CP_MODE cyclic_prefix_mode

6.142.1 Detailed Description

Request structure for the pilot positions calcuation function.

Note

Only first num_res elements in the res_index are used and only first num_res of pilot_positions are populated.
bandwidth_rb is only used by PUCCH formats 2/2a/2b.
num_res and cyclic_prefix_mode is only used by PUCCH formats 1/1a/1b.
Formats 2/2a/2b use the first pointer in res_indices and outputs to pilot_positions_res0.

6.142.2 Field Documentation

6.142.2.1 bandwidth_rb

```
int16_t bblib_pucch_ndash_request::bandwidth_rb
```

The bandwidth in terms of resource blocks that are available for use by PUCCH 36.211, Chapter 5.4 - $N_{RB}^{(2)}$.

6.142.2.2 cyclic_prefix_mode

```
CP_MODE bblib_pucch_ndash_request::cyclic_prefix_mode
```

According to the mode c and d parameters are set as defined in 36.211, Chapter 5.4.1.

6.142.2.3 cyclic_shift

```
int16_t bblib_pucch_ndash_request::cyclic_shift
```

Cyclic shift provided by higher layers as defined in 36.211, Chapter 5.4 - $N_{cs}^{(1)}$.

6.142.2.4 delta_shift

```
int16_t bblib_pucch_ndash_request::delta_shift
```

Delta shift PUCCH provided by higher layers as defined in 36.211, Chapter 5.4 - Δ_{shift}^{PUCCH}.

6.142.2.5 num_pucch

```
uint16_t bblib_pucch_ndash_request::num_pucch
```

Number of PUCCH per resource.

6.142.2.6 num_res

```
uint8_t bblib_pucch_ndash_request::num_res
```

Number of PUCCH resources to process. The valid numbers are integers from 1 to MAX_NUM_RES.

6.142.2.7 pucch_index

```
uint32_t* bblib_pucch_ndash_request::pucch_index
```

Buffer to PUCCH indices. The buffer size has to be a multiply of 512 bits.

6.142.2.8 res_indices

```
int16_t* bblib_pucch_ndash_request::res_indices[4]
```

Resource used for transmission of PUCCH represented by the non-negative indices as defined in 36.211, Chapter 5.4 - $n_{PUCCH}^{(x,\widetilde{p})}$. Each pointer points to a indices buffer of a different PUCCH resource. Pointer 0 points to the buffer for the first resource, pointer 1 to the buffer for the second resource, etc. Only first num_res pointers are used. Ech buffer size has to be a multiply of 512 bits.

6.143 bblib_pucch_ndash_response Struct Reference

```
#include <phy_cestimate_pucch.h>
```

Data Fields

- int16_t * pilot_positions_res0
- int16_t * pilot_positions_res1
- int16_t * pilot_positions_res2
- int16_t * pilot_positions_res3

6.143.1 Detailed Description

The resource indices within the two resource blocks in the two slots of a subframe to which the PUCCH is mapped for every PUCCH in the resource.

6.143.2 Field Documentation

6.143.2.1 pilot_positions_res0

```
int16_t* bblib_pucch_ndash_response::pilot_positions_res0
```

Pointer to the buffer with pilot positions for the PUCCH resource 0 in the interleaved form: slot0, slot1, slot0, slot1, etc.

6.143.2.2 pilot_positions_res1

```
int16_t* bblib_pucch_ndash_response::pilot_positions_res1
```

Pointer to the buffer with pilot positions for the PUCCH resource 1 in the interleaved form: slot0, slot1, slot0, slot1, etc.

6.143.2.3 pilot_positions_res2

```
int16_t* bblib_pucch_ndash_response::pilot_positions_res2
```

Pointer to the buffer with pilot positions for the PUCCH resource 2 in the interleaved form: slot0, slot1, slot0, slot1, etc.

6.143.2.4 pilot_positions_res3

```
int16_t* bblib_pucch_ndash_response::pilot_positions_res3
```

Pointer to the buffer with pilot positions for the PUCCH resource 3 in the interleaved form: slot0, slot1, slot0, slot1, etc.

6.144 bblib_pucch_seq_gen_request Struct Reference

```
#include <phy_pucch_5gnr.h>
```

Data Fields

- uint16_t hopping_id
- uint16_t nsubframe_per_frame
- uint16_t nslot_per_subframe
- uint16_t nsymbol_per_slot

6.144.1 Detailed Description

Request structure for PUCCH Random sequence gen.

6.144.2 Field Documentation

6.144.2.1 hopping_id

`uint16_t bblib_pucch_seq_gen_request::hopping_id`

hopping id 3GPP 38.211 6.3.2.2

6.144.2.2 nslot_per_subframe

`uint16_t bblib_pucch_seq_gen_request::nslot_per_subframe`

number of slots per subframe

6.144.2.3 nsubframe_per_frame

`uint16_t bblib_pucch_seq_gen_request::nsubframe_per_frame`

number of subframes per frame

6.144.2.4 nsymbol_per_slot

`uint16_t bblib_pucch_seq_gen_request::nsymbol_per_slot`

number of symbols per slot

6.145 bblib_pucch_seq_gen_response Struct Reference

`#include <phy_pucch_5gnr.h>`

Data Fields

- uint8_t * cs
- uint8_t * gh
- uint8_t * v

6.145.1 Detailed Description

Response structure for PUCCH Random sequence gen. For the 3 sequence cs, gh and v, the bits is arranged from big to small bit in one byte.

6.145.2 Field Documentation

6.145.2.1 cs

`uint8_t* bblib_pucch_seq_gen_response::cs`

buffer to store the random sequence generated based on the generator seed hopping_id defined in bblib_pucch_seq_gen_request. buffer should be 64byte aligned and of max size 8960

6.145.2.2 gh

`uint8_t* bblib_pucch_seq_gen_response::gh`

buffer to store the random sequence generated based on the generator seed floor(hopping_id/30). buffer should be 64byte aligned and of max size 1280

6.145.2.3 v

`uint8_t* bblib_pucch_seq_gen_response::v`

buffer to store the random sequence generated based on the generator seed (floor(hopping_id/30)<<5) + hopping_id%30. buffer should be 64byte aligned and of max size 160

6.146 bblib_pusch_fo_compensation_5gnr_request Struct Reference

`#include <phy_pusch_focompensation_5gnr.h>`

Data Fields

- float f_offset_angel_est
- int16_t n_rxant
- int16_t cp_size1
- int16_t cp_size
- int16_t n_prb
- int16_t n_fft_size
- int16_t n_comp_start_symb
- int16_t n_comp_symb
- int16_t n_reGroup
- int16_t n_mu
- int16_t ∗ p_foc_in [FOC_MAX_RX_ANT_NUM][FOC_N_SYMB_PER_SLOT]

6.146.1 Detailed Description

Structure defines the Frequency Offset Compensation request in 5GNR.

6.146.2 Field Documentation

6.146.2.1 cp_size

`int16_t bblib_pusch_fo_compensation_5gnr_request::cp_size`

Normal CP Size

6.146.2.2 cp_size1

`int16_t bblib_pusch_fo_compensation_5gnr_request::cp_size1`

CP Size of first symbol

6.146.2.3 f_offset_angel_est

`float bblib_pusch_fo_compensation_5gnr_request::f_offset_angel_est`

Estimated frequency offset angel

6.146.2.4 n_comp_start_symb

`int16_t bblib_pusch_fo_compensation_5gnr_request::n_comp_start_symb`

first symbol index for compensation

6.146.2.5 n_comp_symb

`int16_t bblib_pusch_fo_compensation_5gnr_request::n_comp_symb`

Number of compensation symbols

6.146.2.6 n_fft_size

`int16_t bblib_pusch_fo_compensation_5gnr_request::n_fft_size`

FFT size

6.146.2.7 n_mu

```
int16_t bblib_pusch_fo_compensation_5gnr_request::n_mu
```

Numerology, determine sub carrier spacing, Value: 0->4

6.146.2.8 n_prb

```
int16_t bblib_pusch_fo_compensation_5gnr_request::n_prb
```

Number of PRBs

6.146.2.9 n_reGroup

```
int16_t bblib_pusch_fo_compensation_5gnr_request::n_reGroup
```

Number of REs with same compensation value

6.146.2.10 n_rxant

```
int16_t bblib_pusch_fo_compensation_5gnr_request::n_rxant
```

Number of Rx antennas

6.146.2.11 p_foc_in

```
int16_t* bblib_pusch_fo_compensation_5gnr_request::p_foc_in[FOC_MAX_RX_ANT_NUM][FOC_N_SYMB_PE↩
R_SLOT]
```

Data pointer points to nRx*nSymbol frequency offset compensation input buffer

6.147 bblib_pusch_fo_compensation_5gnr_response Struct Reference

```
#include <phy_pusch_focompensation_5gnr.h>
```

Data Fields

- int16_t * p_comp_out [FOC_MAX_RX_ANT_NUM][FOC_N_SYMB_PER_SLOT]

6.147.1 Detailed Description

Structure defines the Frequency Offset Compensation response in 5GNR.

6.147.2 Field Documentation

6.147.2.1 p_comp_out

```
int16_t* bblib_pusch_fo_compensation_5gnr_response::p_comp_out[FOC_MAX_RX_ANT_NUM][FOC_N_SYMB↔
_PER_SLOT]
```

Data pointer points to nRx∗nSymbol Frequency offset compensation output buffer

6.148 bblib_pusch_fo_estimation_5gnr_request Struct Reference

```
#include <phy_pusch_foestimate_5gnr.h>
```

Data Fields

- int16_t n_dmrs_config_type
- int16_t n_rxant
- int16_t n_layer
- int16_t n_start_prb
- int16_t n_prb
- int16_t n_mu
- int16_t n_fft_size
- int16_t n_fl_dmrs_symb
- int16_t n_dmrs_symb
- int16_t n_dmrs_symb_diff [CE_MAX_DMRS_SYMBOL-1]
- int16_t n_dmrs_port [CE_MAX_TX_LAYER_NUM]
- float f_boost_linear [CE_MAX_TX_LAYER_NUM]
- int16_t n_enable_ta_est
- int16_t n_enable_ta_comp
- int16_t n_interp_method
- int16_t ∗ p_dmrs_base_seq [CE_MAX_TX_LAYER_NUM][CE_MAX_DMRS_SYMBOL]
- int16_t ∗ p_foe_in [CE_MAX_RX_ANT_NUM][CE_MAX_DMRS_SYMBOL]
- int16_t ∗ p_foe_buff
- int16_t n_priori_ta

6.148.1 Detailed Description

Structure defines the Frequency Offset Estimation in 5GNR.

6.148.2 Field Documentation

6.148.2.1 f_boost_linear

```
float bblib_pusch_fo_estimation_5gnr_request::f_boost_linear[CE_MAX_TX_LAYER_NUM]
```

DMRS boosting linear value

6.148.2.2 n_dmrs_config_type

```
int16_t bblib_pusch_fo_estimation_5gnr_request::n_dmrs_config_type
```

DMRS configuration type: 1 or 2

6.148.2.3 n_dmrs_port

```
int16_t bblib_pusch_fo_estimation_5gnr_request::n_dmrs_port[CE_MAX_TX_LAYER_NUM]
```

DMRS port, type1: 0∼7; type2: 0∼11

6.148.2.4 n_dmrs_symb

```
int16_t bblib_pusch_fo_estimation_5gnr_request::n_dmrs_symb
```

Number of DMRS symbols

6.148.2.5 n_dmrs_symb_diff

```
int16_t bblib_pusch_fo_estimation_5gnr_request::n_dmrs_symb_diff[CE_MAX_DMRS_SYMBOL-1]
```

symbol number between DMRS two symbols

6.148.2.6 n_enable_ta_comp

```
int16_t bblib_pusch_fo_estimation_5gnr_request::n_enable_ta_comp
```

bitmap flag shows whether enable TA compensation bit0: TA pre-compensation; bit 1: TA post-compensation

6.148.2.7 n_enable_ta_est

```
int16_t bblib_pusch_fo_estimation_5gnr_request::n_enable_ta_est
```

flag shows whether enable TA estimation

6.148.2.8 n_fft_size

```
int16_t bblib_pusch_fo_estimation_5gnr_request::n_fft_size
```

FFT size

6.148.2.9 n_fl_dmrs_symb

`int16_t bblib_pusch_fo_estimation_5gnr_request::n_fl_dmrs_symb`

Number of front loaded DMRS symbols

6.148.2.10 n_interp_method

`int16_t bblib_pusch_fo_estimation_5gnr_request::n_interp_method`

flag shows which interpolation method used, 0 or 2

6.148.2.11 n_layer

`int16_t bblib_pusch_fo_estimation_5gnr_request::n_layer`

Number of Layers

6.148.2.12 n_mu

`int16_t bblib_pusch_fo_estimation_5gnr_request::n_mu`

Numerology, determine sub carrier spacing, Value: 0->4

6.148.2.13 n_prb

`int16_t bblib_pusch_fo_estimation_5gnr_request::n_prb`

Number of PRBs

6.148.2.14 n_priori_ta

`int16_t bblib_pusch_fo_estimation_5gnr_request::n_priori_ta`

priori information for TA pre-compensation

6.148.2.15 n_rxant

`int16_t bblib_pusch_fo_estimation_5gnr_request::n_rxant`

Number of Rx antennas

6.148.2.16 n_start_prb

`int16_t bblib_pusch_fo_estimation_5gnr_request::n_start_prb`

Starting PRB number

6.148.2.17 p_dmrs_base_seq

```
int16_t* bblib_pusch_fo_estimation_5gnr_request::p_dmrs_base_seq[CE_MAX_TX_LAYER_NUM][CE_MAX_↩
DMRS_SYMBOL]
```

Data pointer points to DMRS base sequence, format 16S14

6.148.2.18 p_foe_buff

```
int16_t* bblib_pusch_fo_estimation_5gnr_request::p_foe_buff
```

FOE internal buffer, length is n_prb∗768 byte

6.148.2.19 p_foe_in

```
int16_t* bblib_pusch_fo_estimation_5gnr_request::p_foe_in[CE_MAX_RX_ANT_NUM][CE_MAX_DMRS_SYMB↩
OL]
```

Data pointer points to nRx∗nSymbol FOE input buffer for each user, first pointer of start RB for the user

6.149 bblib_pusch_fo_estimation_5gnr_response Struct Reference

```
#include <phy_pusch_foestimate_5gnr.h>
```

Data Fields

- int16_t ∗ p_ce_ls [CE_MAX_RX_ANT_NUM][CE_MAX_DMRS_SYMBOL]
- float f_offset_angel_est

6.149.1 Detailed Description

Structure defines the Frequency Offset Estimation in 5GNR.

6.149.2 Field Documentation

6.149.2.1 f_offset_angel_est

```
float bblib_pusch_fo_estimation_5gnr_response::f_offset_angel_est
```

Estimated noise power

6.149.2.2 p_ce_ls

```
int16_t* bblib_pusch_fo_estimation_5gnr_response::p_ce_ls[CE_MAX_RX_ANT_NUM][CE_MAX_DMRS_SYMB↩
OL]
```

Data pointer points to nRx CE LS result buffer

6.150 bblib_pusch_irc_symbol_processing_request Struct Reference

```
#include <phy_pusch_irc_symbol_processing_5gnr.h>
```

Data Fields

- uint16_t nLayerInGroup
- uint16_t nLayerPerUE
- uint16_t nUeInGroup
- uint16_t nRxAnt
- uint16_t nStartSC
- uint16_t nSubCarrier
- uint16_t nTotalSubCarrier
- uint16_t nTotalAlignedSubCarrier
- uint16_t nChSymb
- uint16_t nSymb
- uint16_t nSymbPerDmrs [BBLIB_N_SYMB_PER_SF]
- uint8_t nLlrFxpPoints
- uint8_t nTpFlag
- float nSigma2
- enum bblib_modulation_order eModOrder [BBLIB_MAX_MU]
- uint16_t * pSymbIndex
- int16_t * pChState [BBLIB_MAX_RX_ANT_NUM][BBLIB_MAX_TX_LAYER_NUM]
- int16_t * pRxSignal [BBLIB_MAX_RX_ANT_NUM][BBLIB_N_SYMB_PER_SF]
- void * pRnn_Re [BBLIB_N_SYMB_PER_SF][BBLIB_MAX_RX_ANT_NUM][BBLIB_MAX_RX_ANT_NUM]
- void * pRnn_Im [BBLIB_N_SYMB_PER_SF][BBLIB_MAX_RX_ANT_NUM][BBLIB_MAX_RX_ANT_NUM]
- int16_t nLinInterpEnable
- int16_t nDmrsChSymb
- int16_t nMappingType
- int16_t nGranularity
- float fEstCfo [BBLIB_MAX_TX_LAYER_NUM]
- int16_t nFftSize
- int16_t nNumerology
- int16_t nEnableFoComp

6.150.1 Detailed Description

Request struct of PUSCH symbol processing.

6.150.2 Field Documentation

6.150.2.1 eModOrder

```
enum bblib_modulation_order bblib_pusch_irc_symbol_processing_request::eModOrder[BBLIB_MAX_MU]
```

Supported Modulation Values are: 1 (pi/2 BPSK), 2 (QPSK), 4 (16QAM), 6 (64QAM)

6.150.2.2 fEstCfo

```
float bblib_pusch_irc_symbol_processing_request::fEstCfo[BBLIB_MAX_TX_LAYER_NUM]
```

Normalized frequency offset estimates for each layer

6.150.2.3 nChSymb

```
uint16_t bblib_pusch_irc_symbol_processing_request::nChSymb
```

Number of Channel Symbols - valid range [1-(N_SYMB_PER_SF-1)]

6.150.2.4 nDmrsChSymb

```
int16_t bblib_pusch_irc_symbol_processing_request::nDmrsChSymb
```

Number of Dmrs Symbols - valid range [1-4]. Defines how many CEs stored on input at this function call

6.150.2.5 nEnableFoComp

```
int16_t bblib_pusch_irc_symbol_processing_request::nEnableFoComp
```

Flag to enable frequency offset compensation

6.150.2.6 nFftSize

```
int16_t bblib_pusch_irc_symbol_processing_request::nFftSize
```

FFT Size

6.150.2.7 nGranularity

```
int16_t bblib_pusch_irc_symbol_processing_request::nGranularity
```

Defines the number of adjacent symbols sharing the same CE inside the slot

6.150.2.8 nLayerInGroup

```
uint16_t bblib_pusch_irc_symbol_processing_request::nLayerInGroup
```

Number of Layers in One MU Group - valid values 1, 2, 4

6.150.2.9 nLayerPerUE

`uint16_t bblib_pusch_irc_symbol_processing_request::nLayerPerUE`

Number of Layers per UE, Only Support Equal Layers Number Now - valid values 1, 2

6.150.2.10 nLinInterpEnable

`int16_t bblib_pusch_irc_symbol_processing_request::nLinInterpEnable`

0 - stored DMRS CE is used directly (nearest neighbor), 1 - time linear interpolation is used

6.150.2.11 nLlrFxpPoints

`uint8_t bblib_pusch_irc_symbol_processing_request::nLlrFxpPoints`

Indicate the Decimal Digits of Llr Output Fixed Point Value. Right now need to be 0∼7

6.150.2.12 nMappingType

`int16_t bblib_pusch_irc_symbol_processing_request::nMappingType`

Dmrs Type - A is 0 and B is 1

6.150.2.13 nNumerology

`int16_t bblib_pusch_irc_symbol_processing_request::nNumerology`

Numerology

6.150.2.14 nRxAnt

`uint16_t bblib_pusch_irc_symbol_processing_request::nRxAnt`

Number of Rx Virtual Antennas - valid values 1, 2, 4

6.150.2.15 nSigma2

`float bblib_pusch_irc_symbol_processing_request::nSigma2`

Noise power

6.150.2.16 nStartSC

`uint16_t bblib_pusch_irc_symbol_processing_request::nStartSC`

Start Subcarrier Index

6.150.2.17 nSubCarrier

`uint16_t bblib_pusch_irc_symbol_processing_request::nSubCarrier`

Number of Granted Subcarriers For This Function Call - valid range [1-3276]

6.150.2.18 nSymb

`uint16_t bblib_pusch_irc_symbol_processing_request::nSymb`

Number of Total Granted Symbols - valid range [1-(N_SYMB_PER_SF-1)]. Dummy, not used right now

6.150.2.19 nSymbPerDmrs

`uint16_t bblib_pusch_irc_symbol_processing_request::nSymbPerDmrs[BBLIB_N_SYMB_PER_SF]`

Number of Granted Symbols Per Dmrs - valid range [1-(N_SYMB_PER_SF-1)]

6.150.2.20 nTotalAlignedSubCarrier

`uint16_t bblib_pusch_irc_symbol_processing_request::nTotalAlignedSubCarrier`

Number of Aligned Total Granted Subcarriers, Decide the Buffer Offset, Need to Be Mulitple of 16.

6.150.2.21 nTotalSubCarrier

`uint16_t bblib_pusch_irc_symbol_processing_request::nTotalSubCarrier`

Number of Total Granted Subcarriers, Used In RB Level Split Case, Not Less Than nSubCarrier - valid range [1-3276]

6.150.2.22 nTpFlag

`uint8_t bblib_pusch_irc_symbol_processing_request::nTpFlag`

Indicate Transform Precoding is Enabled or Not, 0 disable, other enable

6.150.2.23 nUeInGroup

`uint16_t bblib_pusch_irc_symbol_processing_request::nUeInGroup`

Number of UEs in This MU Group - valid values 1,2

6.150.2.24 pChState

`int16_t* bblib_pusch_irc_symbol_processing_request::pChState[BBLIB_MAX_RX_ANT_NUM][BBLIB_MAX_TX_LAYER_NUM]`

Data Pointer Points to nRxAnt∗nTxLayer Channel, The Layers Need to Be Mapped UE by UE to Get Correct Layer Demapping Output, format 16S13

6.150.2.25 pRnn_Im

`void* bblib_pusch_irc_symbol_processing_request::pRnn_Im[BBLIB_N_SYMB_PER_SF][BBLIB_MAX_RX_ANT_NUM][BBLIB_MAX_`

Data pointer points to nRxAnt∗nRxAnt∗nSymbol Rnn data

6.150.2.26 pRnn_Re

`void* bblib_pusch_irc_symbol_processing_request::pRnn_Re[BBLIB_N_SYMB_PER_SF][BBLIB_MAX_RX_ANT_NUM][BBLIB_MAX_`

Data pointer points to nRxAnt∗nRxAnt∗nSymbol Rnn data

6.150.2.27 pRxSignal

`int16_t* bblib_pusch_irc_symbol_processing_request::pRxSignal[BBLIB_MAX_RX_ANT_NUM][BBLIB_N_SYMB_PER_SF]`

Data Pointer Points to nRxAnt∗nSymbol Received Data, format 16S13

6.150.2.28 pSymbIndex

`uint16_t* bblib_pusch_irc_symbol_processing_request::pSymbIndex`

Pointer for Data Symbol index

6.151 bblib_pusch_irc_symbol_processing_response Struct Reference

`#include <phy_pusch_irc_symbol_processing_5gnr.h>`

Data Fields

- int8_t ∗ pLlr [BBLIB_MAX_MU]
- float ∗ pMmseGain [BBLIB_MAX_TX_LAYER_NUM]
- float ∗ pPostSINR [BBLIB_MAX_TX_LAYER_NUM]
- void ∗ pEstTxSignal [BBLIB_MAX_TX_LAYER_NUM][BBLIB_N_SYMB_PER_SF]

6.151.1 Detailed Description

Response struct of PUSCH symbol processing.

6.151.2 Field Documentation

6.151.2.1 pEstTxSignal

```
void* bblib_pusch_irc_symbol_processing_response::pEstTxSignal[BBLIB_MAX_TX_LAYER_NUM][BBLIB_N_SYMB_PER_SF]
```

Data pointer points to nTx∗nSymbol estimated TX signal

6.151.2.2 pLlr

```
int8_t* bblib_pusch_irc_symbol_processing_response::pLlr[BBLIB_MAX_MU]
```

Pointer to Output Buffer of LLRs, buffer should be 64 byte aligned, output format 8S(nLlrFxpPoints)

6.151.2.3 pMmseGain

```
float* bblib_pusch_irc_symbol_processing_response::pMmseGain[BBLIB_MAX_TX_LAYER_NUM]
```

Pointer Points to nTx∗1 Estimated MMSE Gain, floating number

6.151.2.4 pPostSINR

```
float* bblib_pusch_irc_symbol_processing_response::pPostSINR[BBLIB_MAX_TX_LAYER_NUM]
```

Pointer Points to nTx∗1 pPostSINR, floating number

6.152 bblib_pusch_symbol_processing_request Struct Reference

```
#include <phy_pusch_symbol_processing_5gnr.h>
```

Data Fields

- uint16_t nLayerInGroup
- uint16_t nLayerPerUE
- uint16_t nUeInGroup
- uint16_t nRxAnt
- uint16_t nStartSC
- uint16_t nSubCarrier
- uint16_t nTotalSubCarrier
- uint16_t nTotalAlignedSubCarrier
- uint16_t nChSymb
- uint16_t nSymb
- uint16_t nSymbPerDmrs [BBLIB_N_SYMB_PER_SF]
- uint8_t nDMRSType
- uint8_t nNrOfCDMs
- uint8_t nNrOfDMRSSymbols
- uint16_t * pDmrsSymbolIdx
- uint8_t * pDmrsPortIdx [BBLIB_MAX_MU]
- uint8_t nLlrFxpPoints
- uint8_t nTpFlag
- float nSigma2
- enum bblib_modulation_order eModOrder [BBLIB_MAX_MU]
- uint16_t * pSymbIndex
- int16_t * pChState [BBLIB_MAX_RX_ANT_NUM][BBLIB_MAX_TX_LAYER_NUM]
- int16_t * pRxSignal [BBLIB_MAX_RX_ANT_NUM][BBLIB_N_SYMB_PER_SF]
- int16_t nLinInterpEnable
- int16_t nDmrsChSymb
- int16_t nMappingType
- int16_t nGranularity
- float fEstCfo [BBLIB_MAX_TX_LAYER_NUM]
- int16_t nFftSize
- int16_t nNumerology
- int16_t nEnableFoComp

6.152.1 Detailed Description

Request struct of PUSCH symbol processing.

6.152.2 Field Documentation

6.152.2.1 eModOrder

```
enum bblib_modulation_order bblib_pusch_symbol_processing_request::eModOrder[BBLIB_MAX_MU]
```

Supported Modulation Values are: 1 (pi/2 BPSK), 2 (QPSK), 4 (16QAM), 6 (64QAM)

6.152.2.2 fEstCfo

```
float bblib_pusch_symbol_processing_request::fEstCfo[BBLIB_MAX_TX_LAYER_NUM]
```

Normalized frequency offset estimates for each layer

6.152.2.3 nChSymb

```
uint16_t bblib_pusch_symbol_processing_request::nChSymb
```

Number of Channel Symbols - valid range [1-(N_SYMB_PER_SF-1)]

6.152.2.4 nDmrsChSymb

```
int16_t bblib_pusch_symbol_processing_request::nDmrsChSymb
```

Number of Dmrs Symbols - valid range [1-4]. Defines how many CEs stored on input at this function call

6.152.2.5 nDMRSType

```
uint8_t bblib_pusch_symbol_processing_request::nDMRSType
```

DMRS Type for PUSCH

6.152.2.6 nEnableFoComp

```
int16_t bblib_pusch_symbol_processing_request::nEnableFoComp
```

Flag to enable frequency offset compensation

6.152.2.7 nFftSize

```
int16_t bblib_pusch_symbol_processing_request::nFftSize
```

FFT Size

6.152.2.8 nGranularity

```
int16_t bblib_pusch_symbol_processing_request::nGranularity
```

Defines the number of adjacent symbols sharing the same CE inside the slot

6.152.2.9 nLayerInGroup

```
uint16_t bblib_pusch_symbol_processing_request::nLayerInGroup
```

Number of Layers in One MU Group - valid values 1, 2, 4

6.152.2.10 nLayerPerUE

`uint16_t bblib_pusch_symbol_processing_request::nLayerPerUE`

Number of Layers per UE, Only Support Equal Layers Number Now - valid values 1, 2

6.152.2.11 nLinInterpEnable

`int16_t bblib_pusch_symbol_processing_request::nLinInterpEnable`

0 - stored DMRS CE is used directly (nearest neighbor), 1 - time linear interpolation is used

6.152.2.12 nLlrFxpPoints

`uint8_t bblib_pusch_symbol_processing_request::nLlrFxpPoints`

Indicate the Decimal Digits of Llr Output Fixed Point Value. Right now need to be 0∼7

6.152.2.13 nMappingType

`int16_t bblib_pusch_symbol_processing_request::nMappingType`

Dmrs Type - A is 0 and B is 1

6.152.2.14 nNrOfCDMs

`uint8_t bblib_pusch_symbol_processing_request::nNrOfCDMs`

Number of CDM DMRS Groups without data

6.152.2.15 nNrOfDMRSSymbols

`uint8_t bblib_pusch_symbol_processing_request::nNrOfDMRSSymbols`

Number of DMRS Symbols in slot

6.152.2.16 nNumerology

`int16_t bblib_pusch_symbol_processing_request::nNumerology`

Numerology

6.152.2.17 nRxAnt

`uint16_t bblib_pusch_symbol_processing_request::nRxAnt`

Number of Rx Virtual Antennas - valid values 1, 2, 4

6.152.2.18 nSigma2

```
float bblib_pusch_symbol_processing_request::nSigma2
```

Noise power

6.152.2.19 nStartSC

```
uint16_t bblib_pusch_symbol_processing_request::nStartSC
```

Start Subcarrier Index

6.152.2.20 nSubCarrier

```
uint16_t bblib_pusch_symbol_processing_request::nSubCarrier
```

Number of Granted Subcarriers For This Function Call - valid range [1-3276]

6.152.2.21 nSymb

```
uint16_t bblib_pusch_symbol_processing_request::nSymb
```

Number of Total Granted Symbols - valid range [1-(N_SYMB_PER_SF-1)]. Dummy, not used right now

6.152.2.22 nSymbPerDmrs

```
uint16_t bblib_pusch_symbol_processing_request::nSymbPerDmrs[BBLIB_N_SYMB_PER_SF]
```

Number of Granted Symbols Per Dmrs - valid range [1-(N_SYMB_PER_SF-1)]

6.152.2.23 nTotalAlignedSubCarrier

```
uint16_t bblib_pusch_symbol_processing_request::nTotalAlignedSubCarrier
```

Number of Aligned Total Granted Subcarriers, Decide the Buffer Offset, Need to Be Mulitple of 16.

6.152.2.24 nTotalSubCarrier

```
uint16_t bblib_pusch_symbol_processing_request::nTotalSubCarrier
```

Number of Total Granted Subcarriers, Used In RB Level Split Case, Not Less Than nSubCarrier - valid range [1-3276]

6.152.2.25 nTpFlag

```
uint8_t bblib_pusch_symbol_processing_request::nTpFlag
```

Indicate Transform Precoding is Enabled or Not, 0 disable, other enable

6.152.2.26 nUeInGroup

```
uint16_t bblib_pusch_symbol_processing_request::nUeInGroup
```

Number of UEs in This MU Group - valid values 1,2

6.152.2.27 pChState

```
int16_t* bblib_pusch_symbol_processing_request::pChState[BBLIB_MAX_RX_ANT_NUM][BBLIB_MAX_TX_LAYER_NUM]
```

Data Pointer Points to nRxAnt∗nTxLayer Channel, The Layers Need to Be Mapped UE by UE to Get Correct Layer Demapping Output, format 16S13

6.152.2.28 pDmrsPortIdx

```
uint8_t* bblib_pusch_symbol_processing_request::pDmrsPortIdx[BBLIB_MAX_MU]
```

Pointer for DMRS Port Indexes for each UE in this group

6.152.2.29 pDmrsSymbolIdx

```
uint16_t* bblib_pusch_symbol_processing_request::pDmrsSymbolIdx
```

Pointer for DMRS Symbol indexes

6.152.2.30 pRxSignal

```
int16_t* bblib_pusch_symbol_processing_request::pRxSignal[BBLIB_MAX_RX_ANT_NUM][BBLIB_N_SYMB_PER_SF]
```

Data Pointer Points to nRxAnt∗nSymbol Received Data, format 16S13

6.152.2.31 pSymbIndex

```
uint16_t* bblib_pusch_symbol_processing_request::pSymbIndex
```

Pointer for Data Symbol index

6.153 bblib_pusch_symbol_processing_response Struct Reference

```
#include <phy_pusch_symbol_processing_5gnr.h>
```

Data Fields

- int8_t * pLlr [BBLIB_MAX_MU]
- float * pMmseGain [BBLIB_MAX_TX_LAYER_NUM]
- float * pPostSINR [BBLIB_MAX_TX_LAYER_NUM]
- float * pMmseOutReal [BBLIB_MAX_TX_LAYER_NUM][BBLIB_N_SYMB_PER_SF]
- float * pMmseOutImag [BBLIB_MAX_TX_LAYER_NUM][BBLIB_N_SYMB_PER_SF]

6.153.1 Detailed Description

Response struct of PUSCH symbol processing.

6.153.2 Field Documentation

6.153.2.1 pLlr

`int8_t* bblib_pusch_symbol_processing_response::pLlr[BBLIB_MAX_MU]`

Pointer to Output Buffer of LLRs, buffer should be 64 byte aligned, output format 8S(nLlrFxpPoints)

6.153.2.2 pMmseGain

`float* bblib_pusch_symbol_processing_response::pMmseGain[BBLIB_MAX_TX_LAYER_NUM]`

Pointer Points to nTx*1 Estimated MMSE Gain, floating number

6.153.2.3 pMmseOutImag

`float* bblib_pusch_symbol_processing_response::pMmseOutImag[BBLIB_MAX_TX_LAYER_NUM][BBLIB_N_SYMB_PER_SF]`

Pointer Points to nTx*1 MMSE output real part, debug interface, floating number

6.153.2.4 pMmseOutReal

`float* bblib_pusch_symbol_processing_response::pMmseOutReal[BBLIB_MAX_TX_LAYER_NUM][BBLIB_N_SYMB_PER_SF]`

Pointer Points to nTx*1 MMSE output real part, debug interface, floating number

6.153.2.5 pPostSINR

`float* bblib_pusch_symbol_processing_response::pPostSINR[BBLIB_MAX_TX_LAYER_NUM]`

Pointer Points to nTx*1 pPostSINR, floating number

6.154 bblib_qr_decomp_request Struct Reference

```
#include <phy_qr_decomposition_5gnr.h>
```

Data Fields

- float * input
- int rows
- int cols

6.154.1 Detailed Description

The request structure used to setup the QR decomposition operation.

6.154.2 Field Documentation

6.154.2.1 cols

```
int bblib_qr_decomp_request::cols
```

Integer value representing the number of columns in the input matrices.

6.154.2.2 input

```
float* bblib_qr_decomp_request::input
```

Pointer to an array containing the input matrices. These matrices are complex valus which are stored as a 32 bit float for the real component and a second 32 bit float for the imaginary component. The input matrices in the array are stored in the following order, starting from the top left entry: RaRbRcRdIaIbIcIdRaRbRcRdIaIbIcId... The number of elements will vary based on the machine type. Array must be 64 byte aligned.

6.154.2.3 rows

```
int bblib_qr_decomp_request::rows
```

Integer value representing the number of rows in the input matrices.

6.155 bblib_qr_decomp_response Struct Reference

```
#include <phy_qr_decomposition_5gnr.h>
```

Data Fields

- float * q_out
- float * r_out
- int rows
- int cols

6.155.1 Detailed Description

The response structure returned to callee with the results of the QR decomposition.

6.155.2 Field Documentation

6.155.2.1 cols

```
int bblib_qr_decomp_response::cols
```

Integer value representing the number of columns in the Q decomposition result.

6.155.2.2 q_out

```
float* bblib_qr_decomp_response::q_out
```

Pointer to an array that will contain the Q decomposition result. The results are complex values which are stored as a 32 bit float for the real component and a second 32 bit float for the imaginary component. The q matrices in the array are stored in the following order, starting from the top left entry: RaRbRcRdIaIbIcIdRaRbRcRdIaIbIcId... The number of elements will vary based on the machine type. Array must be 64 byte aligned.

6.155.2.3 r_out

```
float* bblib_qr_decomp_response::r_out
```

Pointer to an array that will contain the R decomposition result. The results are complex values which are stored as a 32 bit float for the real component and a second 32 bit float for the imaginary component. The r matrices in the array are stored in the following order, starting from the top left entry: RaRbRcRdIaIbIcIdRaRbRcRdIaIbIcId... The number of elements will vary based on the machine type. Array must be 64 byte aligned.

6.155.2.4 rows

```
int bblib_qr_decomp_response::rows
```

Integer value representing the number of rows in the Q decomposition result.

6.156 bblib_rate_dematching_5gnr_request Struct Reference

```
#include <phy_rate_dematching_5gnr.h>
```

Data Fields

- int8_t * p_in
- int8_t * p_harq
- int32_t k0
- int32_t ncb
- int32_t start_null_index
- int32_t num_of_null
- int32_t e
- int32_t rvid
- int32_t zc
- enum bblib_modulation_order modulation_order
- int32_t base_graph
- int32_t isretx

6.156.1 Detailed Description

Request structure providing the inputs and configuration to the rate dematching.

6.156.2 Field Documentation

6.156.2.1 base_graph

```
int32_t bblib_rate_dematching_5gnr_request::base_graph
```

LDPC Base graph, which can be 1 or 2 as defined in TS38212-5.2.1.

6.156.2.2 e

```
int32_t bblib_rate_dematching_5gnr_request::e
```

E The number of post rate-matched output LLR values as defined in TS38212-5.4.2.1.

6.156.2.3 isretx

```
int32_t bblib_rate_dematching_5gnr_request::isretx
```

flag of retransmission, 0: no retransmission, clear HARQ buffer, 1: retransmission

6.156.2.4 k0

`int32_t bblib_rate_dematching_5gnr_request::k0`

k0 the start position in the circular buffer as defined in TS38212-5.4.2.1.

6.156.2.5 modulation_order

`enum bblib_modulation_order bblib_rate_dematching_5gnr_request::modulation_order`

modulation, the allowed values: 1, 2, 4, 6, or 8

6.156.2.6 ncb

`int32_t bblib_rate_dematching_5gnr_request::ncb`

Ncb the length of the circular buffer (including null bits) as defined in TS38212-5.4.2.1.

6.156.2.7 num_of_null

`int32_t bblib_rate_dematching_5gnr_request::num_of_null`

F The number of filler bits used in the encoding

6.156.2.8 p_harq

`int8_t* bblib_rate_dematching_5gnr_request::p_harq`

The pointer of HARQ buffer for both input/output, assumed to be 64B cache aligned. This is also the input for the decoder with Filler bits not included As a consequence there is no pointer in the response structure

6.156.2.9 p_in

`int8_t* bblib_rate_dematching_5gnr_request::p_in`

the pointer of rate dematching input, non cache alignment requirement, the input symbol size is 8bits

6.156.2.10 rvid

`int32_t bblib_rate_dematching_5gnr_request::rvid`

redundancy version id as defined in TS38212-5.4.2.1.

6.156.2.11 start_null_index

```
int32_t bblib_rate_dematching_5gnr_request::start_null_index
```

the start null bit position in Ncb

6.156.2.12 zc

```
int32_t bblib_rate_dematching_5gnr_request::zc
```

Lifting factor Zc as defined in TS38212-5.2.1.

6.157 bblib_rate_dematching_5gnr_response Struct Reference

```
#include <phy_rate_dematching_5gnr.h>
```

6.157.1 Detailed Description

Response structure which is empty - p_harq is also an output.

6.158 bblib_rate_match_dl_request Struct Reference

```
#include <phy_rate_match.h>
```

Data Fields

- int32_t r
- int32_t C
- int8_t direction
- int32_t Nsoft
- int32_t KMIMO
- int32_t MDL_HARQ
- int32_t G
- int32_t NL
- int32_t Qm
- int32_t rvidx
- int8_t bypass_rvidx
- int32_t Kidx
- int32_t nLen
- uint8_t ∗ tin0
- uint8_t ∗ tin1
- uint8_t ∗ tin2

6.158.1 Detailed Description

Structure for input parameters in API of rate matching for LTE.

Note

tin0, tin1, tin2, and output need to be aligned with 128bits.

6.158.2 Field Documentation

6.158.2.1 bypass_rvidx

`int8_t bblib_rate_match_dl_request::bypass_rvidx`

If set rvidx is ignored and k0 set to 0

6.158.2.2 C

`int32_t bblib_rate_match_dl_request::C`

Total number of code blocks.

6.158.2.3 direction

`int8_t bblib_rate_match_dl_request::direction`

flag of DL or UL, 1 for DL and 0 for UL.

6.158.2.4 G

`int32_t bblib_rate_match_dl_request::G`

length of bits before modulation for 1 UE in 1 subframe.

6.158.2.5 Kidx

`int32_t bblib_rate_match_dl_request::Kidx`

Position in turbo code internal interleave table, Kidx=i-1 in TS 136.212 table 5.1.3-3.

6.158.2.6 KMIMO

`int32_t bblib_rate_match_dl_request::KMIMO`

2, which is related to MIMO type.

6.158.2.7 MDL_HARQ

`int32_t bblib_rate_match_dl_request::MDL_HARQ`

Maximum number of DL HARQ.

6.158.2.8 NL

`int32_t bblib_rate_match_dl_request::NL`

Number of layer.

6.158.2.9 nLen

`int32_t bblib_rate_match_dl_request::nLen`

Length of input data from tin0/tin1/tin2 in bits, nLen=K(Kidx+1)+4 in TS 136.212 table 5.1.3-3.

6.158.2.10 Nsoft

`int32_t bblib_rate_match_dl_request::Nsoft`

Total number of soft bits according to UE categories.

6.158.2.11 Qm

`int32_t bblib_rate_match_dl_request::Qm`

Modulation type, which can be 2/4/6.

6.158.2.12 r

`int32_t bblib_rate_match_dl_request::r`

index of current code block in all code blocks.

6.158.2.13 rvidx

`int32_t bblib_rate_match_dl_request::rvidx`

Redundancy version, which can be 0/1/2/3.

6.158.2.14 tin0

`uint8_t* bblib_rate_match_dl_request::tin0`

pointer to input stream 0 from turbo encoder.

6.158.2.15 tin1

```
uint8_t* bblib_rate_match_dl_request::tin1
```

tin1 pointer to input stream 1 from turbo encoder.

6.158.2.16 tin2

```
uint8_t* bblib_rate_match_dl_request::tin2
```

tin2 pointer to input stream 2 from turbo encoder.

6.159 bblib_rate_match_dl_response Struct Reference

```
#include <phy_rate_match.h>
```

Data Fields

- uint8_t * output
- uint32_t OutputLen

6.159.1 Detailed Description

structure for outputs of rate matching for LTE.

6.159.2 Field Documentation

6.159.2.1 output

```
uint8_t* bblib_rate_match_dl_response::output
```

Output buffer for data stream after rate matching.

6.159.2.2 OutputLen

```
uint32_t bblib_rate_match_dl_response::OutputLen
```

outputLen Accumulated output length in bytes in bytes of rate matching before this code block.

6.160 bblib_rate_match_ul_request Struct Reference

```
#include <phy_rate_match.h>
```

Data Fields

- uint8_t * pdmout
- int32_t k0withoutnull
- int32_t ncb
- int32_t e
- int32_t isretx
- int32_t isinverted

6.160.1 Detailed Description

Structure for parameters in API of HARQ, deinterleaver (rate dematching) for LTE.

Note

pdmout, pharqbuffer, pinteleavebuffer and pharqout need to be aligned with 256 bits.

6.160.2 Field Documentation

6.160.2.1 e

```
int32_t bblib_rate_match_ul_request::e
```

HARQ combine input length in bytes.

6.160.2.2 isinverted

```
int32_t bblib_rate_match_ul_request::isinverted
```

input soft decisions are inverted - set to 1 if '1' bit is represented by a positive value

6.160.2.3 isretx

```
int32_t bblib_rate_match_ul_request::isretx
```

Dlag of retransmission, 0: no retransmission, 1: retransmission.

6.160.2.4 k0withoutnull

`int32_t bblib_rate_match_ul_request::k0withoutnull`

K0 without NULL based on RV, position of this input HARQ sequence in ring buffer.

6.160.2.5 ncb

`int32_t bblib_rate_match_ul_request::ncb`

Dyclic buffer length.

6.160.2.6 pdmout

`uint8_t* bblib_rate_match_ul_request::pdmout`

Demodulation output, and input of HARQ.

6.161 bblib_rate_match_ul_response Struct Reference

`#include <phy_rate_match.h>`

Data Fields

- uint8_t * pharqbuffer
- uint8_t * pinteleavebuffer
- uint8_t * pharqout

6.161.1 Detailed Description

Response structure for rate macthing UL.

6.161.2 Field Documentation

6.161.2.1 pharqbuffer

`uint8_t* bblib_rate_match_ul_response::pharqbuffer`

Output of HARQ combine, and input of sub-block deinterleaver.

6.161.2.2 pharqout

```
uint8_t* bblib_rate_match_ul_response::pharqout
```

Final output of this function, and input for turbo decoder.

6.161.2.3 pinteleavebuffer

```
uint8_t* bblib_rate_match_ul_response::pinteleavebuffer
```

Output of sub-block deinterleaver, and input of turbo decoder adapter.

6.162 bblib_reed_muller_conf_fxp_request Struct Reference

```
#include <phy_reed_muller.h>
```

Data Fields

- int16_t * soft_in
- uint16_t dec_in
- int16_t nb_soft
- int16_t nb_dec
- int16_t det_offset
- enum bblib_reed_muller_code_type code_type

6.162.1 Detailed Description

Request structure for RM fixed point confidence decoder.

6.162.2 Field Documentation

6.162.2.1 code_type

```
enum bblib_reed_muller_code_type bblib_reed_muller_conf_fxp_request::code_type
```

Specifies what operation will be perfomed.

6.162.2.2 dec_in

```
uint16_t bblib_reed_muller_conf_fxp_request::dec_in
```

Reed-Muller decoded output.

6.162.2.3 det_offset

`int16_t bblib_reed_muller_conf_fxp_request::det_offset`

Offset that changes detection threshold (4G PUCCH Format 2).

6.162.2.4 nb_dec

`int16_t bblib_reed_muller_conf_fxp_request::nb_dec`

Number of decoded soft decision bits.

6.162.2.5 nb_soft

`int16_t bblib_reed_muller_conf_fxp_request::nb_soft`

Number of soft decisions in sftin.

6.162.2.6 soft_in

`int16_t* bblib_reed_muller_conf_fxp_request::soft_in`

Pointer to received soft decision buffer.

6.163 bblib_reed_muller_conf_request Struct Reference

```
#include <phy_reed_muller.h>
```

Data Fields

- float * soft_in
- uint16_t dec_in
- int16_t nb_soft
- int16_t nb_dec
- int16_t det_offset
- enum bblib_reed_muller_code_type code_type

6.163.1 Detailed Description

Request structure for RM confidence function.

6.163.2 Field Documentation

6.163.2.1 code_type

`enum bblib_reed_muller_code_type bblib_reed_muller_conf_request::code_type`

Specifies what operation will be perfomed.

6.163.2.2 dec_in

`uint16_t bblib_reed_muller_conf_request::dec_in`

Reed-Muller decoded output.

6.163.2.3 det_offset

`int16_t bblib_reed_muller_conf_request::det_offset`

Offset that changes detection threshold (4G PUCCH Format 2).

6.163.2.4 nb_dec

`int16_t bblib_reed_muller_conf_request::nb_dec`

Number of decoded soft decision bits.

6.163.2.5 nb_soft

`int16_t bblib_reed_muller_conf_request::nb_soft`

Number of soft decisions in sftin.

6.163.2.6 soft_in

`float* bblib_reed_muller_conf_request::soft_in`

Pointer to received soft decision buffer.

6.164 bblib_reed_muller_dec_fxp_request Struct Reference

```
#include <phy_reed_muller.h>
```

Data Fields

- int16_t * data_in
- int16_t nb_in
- int16_t nb_out
- enum bblib_reed_muller_code_type code_type

6.164.1 Detailed Description

Request structure for RM fixed point decoder.

6.164.2 Field Documentation

6.164.2.1 code_type

`enum bblib_reed_muller_code_type bblib_reed_muller_dec_fxp_request::code_type`

Specifies what operation will be perfomed.

6.164.2.2 data_in

`int16_t* bblib_reed_muller_dec_fxp_request::data_in`

Pointer to input soft decisions.

6.164.2.3 nb_in

`int16_t bblib_reed_muller_dec_fxp_request::nb_in`

Length of input buffer.

6.164.2.4 nb_out

`int16_t bblib_reed_muller_dec_fxp_request::nb_out`

Number of output bits to be decoded.

6.165 bblib_reed_muller_dec_request Struct Reference

`#include <phy_reed_muller.h>`

Data Fields

- float * data_in
- int16_t nb_in
- int16_t nb_out
- enum bblib_reed_muller_code_type code_type

6.165.1 Detailed Description

Request structure for RM decoder.

6.165.2 Field Documentation

6.165.2.1 code_type

`enum bblib_reed_muller_code_type bblib_reed_muller_dec_request::code_type`

Specifies what operation will be perfomed.

6.165.2.2 data_in

`float* bblib_reed_muller_dec_request::data_in`

Pointer to input soft decisions.

6.165.2.3 nb_in

`int16_t bblib_reed_muller_dec_request::nb_in`

Length of input buffer.

6.165.2.4 nb_out

`int16_t bblib_reed_muller_dec_request::nb_out`

Number of output bits to be decoded.

6.166 bblib_reed_muller_dec_response Struct Reference

```
#include <phy_reed_muller.h>
```

Data Fields

- uint16_t * data_out
- int16_t nb_out

6.166.1 Detailed Description

Response structer for RM decoder.

6.166.2 Field Documentation

6.166.2.1 data_out

`uint16_t* bblib_reed_muller_dec_response::data_out`

Pointer to final output payload.

6.166.2.2 nb_out

`int16_t bblib_reed_muller_dec_response::nb_out`

Length of the output.

6.167 bblib_reed_muller_fht_fxp_request Struct Reference

`#include <phy_reed_muller.h>`

Data Fields

- int16_t * data_in
- int16_t nb_in
- int16_t nb_out

6.167.1 Detailed Description

Request structure for RM fixed point FHT.

6.167.2 Field Documentation

6.167.2.1 data_in

`int16_t* bblib_reed_muller_fht_fxp_request::data_in`

Pointer to input soft decisions.

6.167.2.2 nb_in

```
int16_t bblib_reed_muller_fht_fxp_request::nb_in
```

Length of input buffer.

6.167.2.3 nb_out

```
int16_t bblib_reed_muller_fht_fxp_request::nb_out
```

Number of output bits to be decoded.

6.168 bblib_reed_muller_fht_request Struct Reference

```
#include <phy_reed_muller.h>
```

Data Fields

- float ∗ data_in
- int16_t nb_in
- int16_t nb_out

6.168.1 Detailed Description

Request structure for RM FHT.

6.168.2 Field Documentation

6.168.2.1 data_in

```
float* bblib_reed_muller_fht_request::data_in
```

Pointer to input soft decisions.

6.168.2.2 nb_in

```
int16_t bblib_reed_muller_fht_request::nb_in
```

Length of input buffer.

6.168.2.3 nb_out

```
int16_t bblib_reed_muller_fht_request::nb_out
```

Number of output bits to be decoded.

6.169 bblib_reed_muller_fht_response Struct Reference

```
#include <phy_reed_muller.h>
```

Data Fields

- uint16_t * data_out
- int16_t nb_out

6.169.1 Detailed Description

Response structur for RM FHT.

6.169.2 Field Documentation

6.169.2.1 data_out

```
uint16_t* bblib_reed_muller_fht_response::data_out
```

Pointer to final output payload.

6.169.2.2 nb_out

```
int16_t bblib_reed_muller_fht_response::nb_out
```

Length of the output.

6.170 bblib_remapping_pdsch_request Struct Reference

```
#include <phy_remapping_pdsch.h>
```

Data Fields

- unsigned char AntPort
- enReMapType type
- complex_int16_t * preCode0
- complex_int16_t * preCode1
- complex_int16_t * preCode2
- complex_int16_t * preCode3

6.170.1 Detailed Description

Standard request structure for psdsch remapping.

6.170.2 Field Documentation

6.170.2.1 AntPort

`unsigned char bblib_remapping_pdsch_request::AntPort`

AntPort: (valid value:2, 4).

6.170.2.2 preCode0

`complex_int16_t* bblib_remapping_pdsch_request::preCode0`

TX0 input buffer.

6.170.2.3 preCode1

`complex_int16_t* bblib_remapping_pdsch_request::preCode1`

TX1 input buffer.

6.170.2.4 preCode2

`complex_int16_t* bblib_remapping_pdsch_request::preCode2`

TX2 input buffer (4tx case)/NULL (2tx case).

6.170.2.5 preCode3

`complex_int16_t* bblib_remapping_pdsch_request::preCode3`

TX3 input buffer (4tx case)/NULL (2tx case).

6.170.2.6 type

```
enReMapType bblib_remapping_pdsch_request::type
```

reMapType: Different RB RE mapping type (valid value:0-2).

6.171 bblib_remapping_pdsch_response Struct Reference

```
#include <phy_remapping_pdsch.h>
```

Data Fields

- reMappingInput input [3][14]
- complex_int16_t * symbAnn0
- complex_int16_t * symbAnn1
- complex_int16_t * symbAnn2
- complex_int16_t * symbAnn3

6.171.1 Detailed Description

Standard response structure for psdsch remapping.

6.171.2 Field Documentation

6.171.2.1 input

```
reMappingInput bblib_remapping_pdsch_response::input[3][14]
```

Record para list of Symbol type, input/output offset for different time&frequency.

6.171.2.2 symbAnn0

```
complex_int16_t* bblib_remapping_pdsch_response::symbAnn0
```

TX0 output buffer.

6.171.2.3 symbAnn1

```
complex_int16_t* bblib_remapping_pdsch_response::symbAnn1
```

TX1 output buffer.

6.171.2.4 symbAnn2

`complex_int16_t* bblib_remapping_pdsch_response::symbAnn2`

TX2 output buffer (4tx case)/NULL (2tx case).

6.171.2.5 symbAnn3

`complex_int16_t* bblib_remapping_pdsch_response::symbAnn3`

TX3 output buffer (4tx case)/NULL (2tx case).

6.172 bblib_sample_kernel_request Struct Reference

`#include <phy_sample_kernel.h>`

Data Fields

- void ∗ a
- void ∗ b
- int num_symbols

6.172.1 Detailed Description

Structure defining the sample kernel input interface. All Kernels in the SDK follow the same input and output formats containing a request and response structure. The request contains all inputs and response contains all outputs. All Structures and enums used in public header files should contain the prefix bblib.

It is important to detail the format of the input data. e.g IQ - I Q samples are interleaved in standard complex number format

For fixed point implementations it is also be necessary to specify the number format e.g 16s0 - 16bits, signed, 0 bits after the binary point. 32u8 - 32bits unsigned, 8 bits after the binary point. In addition the range of acceptable numbers should be declared where it is less than supported by the number format.

Note

Most Kernels support both a floating point (flp) version and fixed point version (fxp). Where possible the request and response structures should be common between both flp and fxp using void pointers. If this is not possible separate definitions of request and response structures shall be defined one with the prefix fxp for fixed point inputs/outputs.

6.172.2 Field Documentation

6.172.2.1 a

```
void* bblib_sample_kernel_request::a
```

Pointer to input buffer containing complex input symbols to be multiplied by complex inputs in b - buffer must be 64 byte aligned. Input format should be I Q in either floating or int16_t depending on function call used

6.172.2.2 b

```
void* bblib_sample_kernel_request::b
```

Pointer to input buffer containing complex input symbols to be multiplied by complex inputs in a - buffer must be 64 byte aligned Input format should be I Q in either floating or int16_t depending on function call used

6.172.2.3 num_symbols

```
int bblib_sample_kernel_request::num_symbols
```

Number of input symbols in a and b. Note input symbol is I + Q, so total length of the input buffers a and b shall be num_symbols*2

6.173 bblib_sample_kernel_response Struct Reference

```
#include <phy_sample_kernel.h>
```

Data Fields

- void * result

6.173.1 Detailed Description

Structure defining the sample kernel output interface. It is important to detail the format of the data in the output buffer(s).

6.173.2 Field Documentation

6.173.2.1 result

```
void* bblib_sample_kernel_response::result
```

Pointer to output buffer of complex values

- buffer should be 64 byte aligned. The output buffer will contain data in I Q format in either floating point or int16_t depending on function call used

6.174 bblib_scramble_5gnr_request Struct Reference

```
#include <phy_scramble_5gnr.h>
```

Data Fields

- uint8_t * data_in
- uint32_t c_init
- uint32_t len

6.174.1 Detailed Description

Request structure for scrambler/descramble.

6.174.2 Field Documentation

6.174.2.1 c_init

```
uint32_t bblib_scramble_5gnr_request::c_init
```

The value of init.

6.174.2.2 data_in

```
uint8_t* bblib_scramble_5gnr_request::data_in
```

The input data before scramble, must be aligned to 64bits.

6.174.2.3 len

```
uint32_t bblib_scramble_5gnr_request::len
```

The length of input data in scrambled bits, for descrambling this corresponds to one 8 bit LLR per scrambled bit

6.175 bblib_scramble_5gnr_response Struct Reference

```
#include <phy_scramble_5gnr.h>
```

Data Fields

- uint8_t * data_out
- uint32_t len

6.175.1 Detailed Description

Response structure for scrambler/descramble.

6.175.2 Field Documentation

6.175.2.1 data_out

`uint8_t* bblib_scramble_5gnr_response::data_out`

The output data after scramble, must be aligned to 64bits.

6.175.2.2 len

`uint32_t bblib_scramble_5gnr_response::len`

The length of output data in scrambled bits, for descrambling this corresponds to one 8 bit LLR per scrambled bit

6.176 bblib_scramble_request Struct Reference

`#include <phy_scramble.h>`

Data Fields

- uint8_t * data_in
- uint32_t c_init
- uint32_t len

6.176.1 Detailed Description

Request structure for scrambler.

6.176.2 Field Documentation

6.176.2.1 c_init

`uint32_t bblib_scramble_request::c_init`

The value of init.

6.176.2.2 data_in

```
uint8_t* bblib_scramble_request::data_in
```

The input data before scramble, must be aligned to 64bits.

6.176.2.3 len

```
uint32_t bblib_scramble_request::len
```

The length of input data in bytes

6.177 bblib_scramble_response Struct Reference

```
#include <phy_scramble.h>
```

Data Fields

- uint8_t * data_out
- int32_t len

6.177.1 Detailed Description

Response structure for scrambler.

6.177.2 Field Documentation

6.177.2.1 data_out

```
uint8_t* bblib_scramble_response::data_out
```

The output data after scramble, must be aligned to 64bits.

6.177.2.2 len

```
int32_t bblib_scramble_response::len
```

The length of output data in bytes

6.178 bblib_singular_value_decomp_request Struct Reference

```
#include <singular_value_decomp.h>
```

Data Fields

- float * p_data_in
- int32_t n_matrix_dim
- int32_t n_matrix_num
- int32_t max_iter
- float * min_err

6.178.1 Detailed Description

Structure for input parameters in API of singular value decomposition.

6.178.2 Field Documentation

6.178.2.1 max_iter

```
int32_t bblib_singular_value_decomp_request::max_iter
```

Maximum iteration time for each component decomposition

6.178.2.2 min_err

```
float* bblib_singular_value_decomp_request::min_err
```

Minimum error to end the iteration for different compoenent

6.178.2.3 n_matrix_dim

```
int32_t bblib_singular_value_decomp_request::n_matrix_dim
```

The dimension of the matrices, now only support square matrix

6.178.2.4 n_matrix_num

```
int32_t bblib_singular_value_decomp_request::n_matrix_num
```

Number of the matrices to be decomposed

6.178.2.5 p_data_in

```
float* bblib_singular_value_decomp_request::p_data_in
```

Input complex matrices for decomposition, in sequence I Q I Q...and row by row total length is n_matrix_dim∗n_↩matrix_dim∗n_matrix_num∗2 32bits

6.179 bblib_singular_value_decomp_response Struct Reference

```
#include <singular_value_decomp.h>
```

Data Fields

- float ∗ p_v_out
- float ∗ p_u_out
- float ∗ p_s_value

6.179.1 Detailed Description

Structure for output parameters in API of singular value decomposition.

6.179.2 Field Documentation

6.179.2.1 p_s_value

```
float* bblib_singular_value_decomp_response::p_s_value
```

Output the singular values, only non-negative real value, total length n_matrix_dim∗n_matrix_num 32bits

6.179.2.2 p_u_out

```
float* bblib_singular_value_decomp_response::p_u_out
```

Output left singular complex matrics , in sequence I Q I Q...and column by column total length is n_matrix_dim∗n↩_matrix_dim∗n_matrix_num∗2 32bits

6.179.2.3 p_v_out

```
float* bblib_singular_value_decomp_response::p_v_out
```

Output right singular complex matrics, in sequence I Q I Q...and column by column total length is n_matrix_dim∗n↩_matrix_dim∗n_matrix_num∗2 32bits

6.180 bblib_srs_cestimate_5gnr_request Struct Reference

`#include <phy_srs_cestimate_5gnr.h>`

Data Fields

- int16_t nComb
- int16_t nCombOffset
- int16_t nStartSc
- int16_t n_mu
- int16_t n_fft_size
- int16_t nPRBs
- int16_t nRxAnts
- int16_t nPorts
- int16_t nUser
- int16_t nSrsCeMethod
- int16_t nCyclic [BBLIB_SRS_MAX_UE_NUM]
- complex_int16_t * pSrsLocalSeq [BBLIB_SRS_MAX_UE_NUM][BBLIB_SRS_MAX_PORT_PER_UE]
- complex_int16_t * pCEIn [BBLIB_SRS_MAX_RX_ANT]

6.180.1 Detailed Description

Request structure providing the inputs and configuration to the SRS CE.

6.180.2 Field Documentation

6.180.2.1 n_fft_size

`int16_t bblib_srs_cestimate_5gnr_request::n_fft_size`

FFT size

6.180.2.2 n_mu

`int16_t bblib_srs_cestimate_5gnr_request::n_mu`

Numerology, determine sub carrier spacing, Value: 0->4

6.180.2.3 nComb

`int16_t bblib_srs_cestimate_5gnr_request::nComb`

subcarrier interval.

6.180.2.4 nCombOffset

`int16_t bblib_srs_cestimate_5gnr_request::nCombOffset`

subcarrier offset relative to subcarrier 0 of a RB

6.180.2.5 nCyclic

`int16_t bblib_srs_cestimate_5gnr_request::nCyclic[BBLIB_SRS_MAX_UE_NUM]`

cyclic shift..

6.180.2.6 nPorts

`int16_t bblib_srs_cestimate_5gnr_request::nPorts`

Number of Tx antennas.

6.180.2.7 nPRBs

`int16_t bblib_srs_cestimate_5gnr_request::nPRBs`

Number of PRBs.

6.180.2.8 nRxAnts

`int16_t bblib_srs_cestimate_5gnr_request::nRxAnts`

Number of Rx antennas.

6.180.2.9 nSrsCeMethod

`int16_t bblib_srs_cestimate_5gnr_request::nSrsCeMethod`

CE interp method, 0:RE interp 1:RB interp.

6.180.2.10 nStartSc

`int16_t bblib_srs_cestimate_5gnr_request::nStartSc`

Start subcarrier number.

6.180.2.11 nUser

`int16_t bblib_srs_cestimate_5gnr_request::nUser`

number of UE.

6.180.2.12 pCEIn

```
complex_int16_t* bblib_srs_cestimate_5gnr_request::pCEIn[BBLIB_SRS_MAX_RX_ANT]
```

Data pointer points to CE input buffer. Format 16S13

6.180.2.13 pSrsLocalSeq

```
complex_int16_t* bblib_srs_cestimate_5gnr_request::pSrsLocalSeq[BBLIB_SRS_MAX_UE_NUM][BBLIB_SRS_MAX_PORT_PER_U
```

base sequence for this cdm group. Format 16S14

6.181 bblib_srs_cestimate_5gnr_response Struct Reference

```
#include <phy_srs_cestimate_5gnr.h>
```

Data Fields

- complex_int16_t * pCEIsOut [BBLIB_SRS_MAX_RX_ANT][BBLIB_SRS_MAX_UE_NUM][BBLIB_SRS_MAX_PORT_PER_UE
- complex_int16_t * pCEOut [BBLIB_SRS_MAX_RX_ANT][BBLIB_SRS_MAX_UE_NUM][BBLIB_SRS_MAX_PORT_PER_UE]
- complex_int16_t * pCEOutRBAvg [BBLIB_SRS_MAX_RX_ANT][BBLIB_SRS_MAX_UE_NUM][BBLIB_SRS_MAX_PORT_PEF
- float sigma2_mean
- float sigma2 [BBLIB_SRS_MAX_UE_NUM]
- int16_t IdftScale [BBLIB_SRS_MAX_RX_ANT/PARALLEL_FACTOR][PARALLEL_FACTOR]
- int16_t IdftShift [BBLIB_SRS_MAX_RX_ANT/PARALLEL_FACTOR]
- int16_t dftScale [BBLIB_SRS_MAX_UE_NUM][BBLIB_SRS_MAX_PORT_PER_UE][BBLIB_SRS_MAX_RX_ANT/P↩ ARALLEL_FACTOR]
- int16_t dftShift [BBLIB_SRS_MAX_UE_NUM][BBLIB_SRS_MAX_PORT_PER_UE][BBLIB_SRS_MAX_RX_ANT/P↩ ARALLEL_FACTOR]

6.181.1 Detailed Description

Response structure bundling the outputs from the SRS CE.

6.181.2 Field Documentation

6.181.2.1 dftScale

```
int16_t bblib_srs_cestimate_5gnr_response::dftScale[BBLIB_SRS_MAX_UE_NUM][BBLIB_SRS_MAX_PORT_PER_UE][BBLIB_SRS
ARALLEL_FACTOR]
```

DFT Scale number, reported by DFT module.

6.181.2.2 dftShift

```
int16_t bblib_srs_cestimate_5gnr_response::dftShift[BBLIB_SRS_MAX_UE_NUM][BBLIB_SRS_MAX_PORT_PER_UE][BBLIB_SRS
ARALLEL_FACTOR]
```

DFT Shift number, Left Shift bit before send data into DFT, make sure data using 15bits.

6.181.2.3 IdftScale

```
int16_t bblib_srs_cestimate_5gnr_response::IdftScale[BBLIB_SRS_MAX_RX_ANT/PARALLEL_FACTOR][P←
ARALLEL_FACTOR]
```

IDFT Scale number, reported by IDFT module.

6.181.2.4 IdftShift

```
int16_t bblib_srs_cestimate_5gnr_response::IdftShift[BBLIB_SRS_MAX_RX_ANT/PARALLEL_FACTOR]
```

IDFT Shift number, Left Shift bit before send data into IDFT, make sure data using 15bits.

6.181.2.5 pCEIsOut

```
complex_int16_t* bblib_srs_cestimate_5gnr_response::pCElsOut[BBLIB_SRS_MAX_RX_ANT][BBLIB_SRS_MAX_UE_NUM][BBLIB
```

Data pointer points to nRx∗nTx CE output buffer. Format 16S13

6.181.2.6 pCEOut

```
complex_int16_t* bblib_srs_cestimate_5gnr_response::pCEOut[BBLIB_SRS_MAX_RX_ANT][BBLIB_SRS_MAX_UE_NUM][BBLIB_S
```

Data pointer points to nRx∗nTx CE output buffer. Format 16S13

6.181.2.7 pCEOutRBAvg

```
complex_int16_t* bblib_srs_cestimate_5gnr_response::pCEOutRBAvg[BBLIB_SRS_MAX_RX_ANT][BBLIB_SRS_MAX_UE_NUM][BB
```

Data pointer points to nRx∗nTx CE output buffer. Format 16S13

6.181.2.8 sigma2

```
float bblib_srs_cestimate_5gnr_response::sigma2[BBLIB_SRS_MAX_UE_NUM]
```

pointer to point the Estimated linear SNR buffer for each user.

6.181.2.9 sigma2_mean

```
float bblib_srs_cestimate_5gnr_response::sigma2_mean
```

Estimated noise power.

6.182 bblib_ta_compensation_request Struct Reference

```
#include <phy_ta_compensation_5gnr.h>
```

Data Fields

- int16_t * input0
- int16_t * input1
- int32_t data_len_bytes

6.182.1 Detailed Description

Request structure for ta_compensation function.

6.182.2 Field Documentation

6.182.2.1 data_len_bytes

```
int32_t bblib_ta_compensation_request::data_len_bytes
```

data Len input/output.

6.182.2.2 input0

```
int16_t* bblib_ta_compensation_request::input0
```

Input/Output symbol data align with 64 bytes.

6.182.2.3 input1

```
int16_t* bblib_ta_compensation_request::input1
```

Input symbol data align with 64 bytes.

6.183 bblib_ta_compensation_response Struct Reference

```
#include <phy_ta_compensation_5gnr.h>
```

Data Fields

- int16_t * output
- int32_t data_len_bytes

6.183.1 Detailed Description

Response structure for ta_compensation function.

6.183.2 Field Documentation

6.183.2.1 data_len_bytes

```
int32_t bblib_ta_compensation_response::data_len_bytes
```

Number of output soft-bits.

6.183.2.2 output

```
int16_t* bblib_ta_compensation_response::output
```

Pointer to the output soft-bit after ta_compensation. Buffer has to be 64 bytes aligned.

6.184 bblib_tbcc_encoder_request Struct Reference

```
#include <phy_tbcc.h>
```

Data Fields

- int32_t num_enc_bits
- int32_t num_remaining_bits
- int32_t num_interl_bits
- uint8_t * input

6.184.1 Detailed Description

Request structure for tbcc.

Note

The input buffer must be 64-byte aligned.

6.184.2 Field Documentation

6.184.2.1 input

`uint8_t* bblib_tbcc_encoder_request::input`

Input buffer.

6.184.2.2 num_enc_bits

`int32_t bblib_tbcc_encoder_request::num_enc_bits`

Number of encoded bits.

6.184.2.3 num_interl_bits

`int32_t bblib_tbcc_encoder_request::num_interl_bits`

Number of interleaver bits.

6.184.2.4 num_remaining_bits

`int32_t bblib_tbcc_encoder_request::num_remaining_bits`

Number of remainder bits.

6.185 bblib_tbcc_encoder_response Struct Reference

`#include <phy_tbcc.h>`

Data Fields

- uint32_t * output

6.185.1 Detailed Description

Response structure for tbcc.

Note

The output buffer must be 64-byte aligned.

6.185.2 Field Documentation

6.185.2.1 output

```
uint32_t* bblib_tbcc_encoder_response::output
```

Output buffer for the first parity stream.

6.186 bblib_turbo_adapter_ul_request Struct Reference

```
#include <phy_rate_match.h>
```

Data Fields

- uint8_t * pinteleavebuffer
- int32_t ncb
- int32_t isinverted

6.186.1 Detailed Description

Request structure for parameters in API of Turbo Adapter for LTE.

Note

pinteleavebuffer and pharqout need to be aligned with 256 bits

6.186.2 Field Documentation

6.186.2.1 isinverted

```
int32_t bblib_turbo_adapter_ul_request::isinverted
```

input soft decisions are inverted - set to 1 if '1' bit is represented by a positive value

6.186.2.2 ncb

`int32_t bblib_turbo_adapter_ul_request::ncb`

cyclic buffer length

6.186.2.3 pinteleavebuffer

`uint8_t* bblib_turbo_adapter_ul_request::pinteleavebuffer`

output of sub-block deinterleaver, and input of turbo decoder adapter

6.187 bblib_turbo_adapter_ul_response Struct Reference

`#include <phy_rate_match.h>`

Data Fields

- uint8_t * pharqout

6.187.1 Detailed Description

Response structure for parameters in API of Turbo Adapter for LTE.

6.187.2 Field Documentation

6.187.2.1 pharqout

`uint8_t* bblib_turbo_adapter_ul_response::pharqout`

final output of this function, and input for turbo decoder

6.188 bblib_turbo_decoder_request Struct Reference

`#include <phy_turbo.h>`

Data Fields

- int32_t c
- int32_t k
- int32_t k_idx
- int32_t max_iter_num
- int32_t early_term_disable
- int8_t * input

6.188.1 Detailed Description

Request structure for turbo decoder.

6.188.2 Field Documentation

6.188.2.1 c

`int32_t bblib_turbo_decoder_request::c`

C index value of codeblock for TB.

Its element has 1 to 1 mapping relationship with TBS_L1.

6.188.2.2 early_term_disable

`int32_t bblib_turbo_decoder_request::early_term_disable`

If set to 1, then max_iter_num is always used regardless of CRC check pass / fail. If 0, least number of iterations are used (1 <= iter <= max_iter_num) for decoding till crc check is pass

6.188.2.3 input

`int8_t* bblib_turbo_decoder_request::input`

Input buffer must be 64 bytes aligned

6.188.2.4 k

`int32_t bblib_turbo_decoder_request::k`

K index value of codeblock for each TB.

Its element has 1 to 1 mapping relationship with TBS_L1.

6.188.2.5 k_idx

```
int32_t bblib_turbo_decoder_request::k_idx
```

Size index in TS.36.212, table 5.1.3-3 of codeblock for each TB.

Its element has 1 to 1 mapping relationship with TBS_L1.

6.188.2.6 max_iter_num

```
int32_t bblib_turbo_decoder_request::max_iter_num
```

Maximum number of decoder iterations

6.189 bblib_turbo_decoder_response Struct Reference

```
#include <phy_turbo.h>
```

Data Fields

- uint8_t * output
- int8_t * ag_buf
- uint16_t * cb_buf

6.189.1 Detailed Description

Response structure for turbo decoder.

6.189.2 Field Documentation

6.189.2.1 ag_buf

```
int8_t* bblib_turbo_decoder_response::ag_buf
```

Alfa-gamma buffer to be used for internal calculations.

The expected buffer length is 6528*16 bytes.

6.189.2.2 cb_buf

```
uint16_t* bblib_turbo_decoder_response::cb_buf
```

Code block bits buffer used for internal calculations.

The expected buffer length is K/8.

6.189.2.3 output

```
uint8_t* bblib_turbo_decoder_response::output
```

Output buffer must be 64 bytes aligned.

6.190 bblib_turbo_encoder_request Struct Reference

```
#include <phy_turbo.h>
```

Data Fields

- uint32_t length
- uint8_t case_id
- uint8_t * input_win

6.190.1 Detailed Description

Request structure for turbo encoder.

6.190.2 Field Documentation

6.190.2.1 case_id

```
uint8_t bblib_turbo_encoder_request::case_id
```

Index of the internal interleaver parameters case index - TS 36.212, Table 5.1.3-3, column 'i'.

6.190.2.2 input_win

```
uint8_t* bblib_turbo_encoder_request::input_win
```

Information and CRC bits buffer.

6.190.2.3 length

```
uint32_t bblib_turbo_encoder_request::length
```

Length of the input in bytes.

6.191 bblib_turbo_encoder_response Struct Reference

```
#include <phy_turbo.h>
```

Data Fields

- uint8_t * output_win_0
- uint8_t * output_win_1
- uint8_t * output_win_2

6.191.1 Detailed Description

Response structure for turbo encoder.

6.191.2 Field Documentation

6.191.2.1 output_win_0

```
uint8_t* bblib_turbo_encoder_response::output_win_0
```

Layer 0 bits buffer.

6.191.2.2 output_win_1

```
uint8_t* bblib_turbo_encoder_response::output_win_1
```

Layer 1 bits buffer.

6.191.2.3 output_win_2

```
uint8_t* bblib_turbo_encoder_response::output_win_2
```

Layer 2 bits buffer.

6.192 bblib_zc_sequence_gen_request Struct Reference

```
#include <phy_zc_sequence_gen.h>
```

Data Fields

- size_t num_re
- size_t layer_idx
- uint8_t dmrs_base_seq_id
- uint8_t dmrs_grp_seq_id
- uint8_t cell_cyclic_shift
- uint8_t dci_cyclic_shift
- uint8_t cell_id
- uint8_t subframe
- uint8_t delta_ss
- enum sym_type cfg
- int num_dmrs

6.192.1 Detailed Description

The request structure used to populate the generator settings.

6.192.2 Field Documentation

6.192.2.1 cell_cyclic_shift

```
uint8_t bblib_zc_sequence_gen_request::cell_cyclic_shift
```

Cyclic shift for cell signalled from higher layers.

6.192.2.2 cell_id

```
uint8_t bblib_zc_sequence_gen_request::cell_id
```

Cell ID.

6.192.2.3 cfg

```
enum sym_type bblib_zc_sequence_gen_request::cfg
```

cfg, Determines whether to generate symbols based on 36.211 or 38.211.

6.192.2.4 dci_cyclic_shift

```
uint8_t bblib_zc_sequence_gen_request::dci_cyclic_shift
```

Cyclic shift signalled from DCI.

6.192.2.5 delta_ss

`uint8_t bblib_zc_sequence_gen_request::delta_ss`

delta_ss, configured by higher layers.

6.192.2.6 dmrs_base_seq_id

`uint8_t bblib_zc_sequence_gen_request::dmrs_base_seq_id`

DMRS base sequence ID in the sequence group.

6.192.2.7 dmrs_grp_seq_id

`uint8_t bblib_zc_sequence_gen_request::dmrs_grp_seq_id`

DMRS group sequence ID.

6.192.2.8 layer_idx

`size_t bblib_zc_sequence_gen_request::layer_idx`

Layer ID, should be in the range of {0,...,(kmaxNumLayers-1)}.

6.192.2.9 num_dmrs

`int bblib_zc_sequence_gen_request::num_dmrs`

num_dmrs, Number of DMRS symbols

6.192.2.10 num_re

`size_t bblib_zc_sequence_gen_request::num_re`

Number of resource elements in the sequence.

6.192.2.11 subframe

`uint8_t bblib_zc_sequence_gen_request::subframe`

Subframe ID.

6.193 bblib_zc_sequence_gen_response Struct Reference

`#include <phy_zc_sequence_gen.h>`

Data Fields

- complex_float * ref_dmrs

6.193.1 Detailed Description

The response structure returned containing the result of the sequence generation.

6.193.2 Field Documentation

6.193.2.1 ref_dmrs

```
complex_float* bblib_zc_sequence_gen_response::ref_dmrs
```

Pointer to the reference DMRS symbols, must be cache aligned.

6.194 bblib_zf_matrix_gen_request Struct Reference

```
#include <phy_zf_matrix_gen.h>
```

Data Fields

- int16_t * pChState [BBLIB_ZF_MAX_RX_ANT_NUM][BBLIB_ZF_MAX_RX_ANT_NUM]
- int16_t * pScalingfactor
- int16_t nLayer
- int16_t nRxAnt
- int16_t nStart
- int16_t nLen
- int16_t nFlagULDL
- uint8_t nInvShiftBits

6.194.1 Detailed Description

Request struct of zf weight matrix generation.

6.194.2 Field Documentation

6.194.2.1 nFlagULDL

`int16_t bblib_zf_matrix_gen_request::nFlagULDL`

Flag to indicate UL Weight or DL Weight, 0:DL, 1:UL

6.194.2.2 nInvShiftBits

`uint8_t bblib_zf_matrix_gen_request::nInvShiftBits`

Shift bits for inversion preocess

6.194.2.3 nLayer

`int16_t bblib_zf_matrix_gen_request::nLayer`

Number of Layers - valid values 1∼8 or 16

6.194.2.4 nLen

`int16_t bblib_zf_matrix_gen_request::nLen`

Number of matrices

6.194.2.5 nRxAnt

`int16_t bblib_zf_matrix_gen_request::nRxAnt`

Number of Rx antennas - valid values 32 or 64

6.194.2.6 nStart

`int16_t bblib_zf_matrix_gen_request::nStart`

Start point

6.194.2.7 pChState

`int16_t* bblib_zf_matrix_gen_request::pChState[BBLIB_ZF_MAX_RX_ANT_NUM][BBLIB_ZF_MAX_RX_ANT_NUM]`

Data pointer points to channel

6.194.2.8 pScalingfactor

`int16_t* bblib_zf_matrix_gen_request::pScalingfactor`

Scaling factor pointer

6.195 bblib_zf_matrix_gen_response Struct Reference

```
#include <phy_zf_matrix_gen.h>
```

Data Fields

- int16_t * pWeightMatrix [BBLIB_ZF_MAX_RX_ANT_NUM][BBLIB_ZF_MAX_RX_ANT_NUM]
- int16_t * pWeightOutBufs [BBLIB_ZF_MAX_RX_ANT_NUM]

6.195.1 Detailed Description

Response struct of zf weight matrix generation.

6.195.2 Field Documentation

6.195.2.1 pWeightMatrix

```
int16_t* bblib_zf_matrix_gen_response::pWeightMatrix[BBLIB_ZF_MAX_RX_ANT_NUM][BBLIB_ZF_MAX_RX_ANT_NUM]
```

Data pointer points to weight matrix

6.195.2.2 pWeightOutBufs

```
int16_t* bblib_zf_matrix_gen_response::pWeightOutBufs[BBLIB_ZF_MAX_RX_ANT_NUM]
```

Data pointer points to weight matrix

6.196 COMPLEX32 Struct Reference

```
#include <common_typedef_sdk.h>
```

Data Fields

- float re
- float im

6.196.1 Detailed Description

Defines 64-bit complex structure; both real part and image part have 32 bit width.

6.196.2 Field Documentation

6.196.2.1 im

```
float COMPLEX32::im
```

32-bit image part

6.196.2.2 re

```
float COMPLEX32::re
```

32-bit real part

6.197 complex_double Struct Reference

```
#include <common_typedef_sdk.h>
```

Data Fields

- double re
- double im

6.197.1 Detailed Description

Defines 128-bit complex structure; both real part and image part have 64 bit width.

6.197.2 Field Documentation

6.197.2.1 im

```
double complex_double::im
```

64-bit image part

6.197.2.2 re

```
double complex_double::re
```

64-bit real part

6.198 complex_float Struct Reference

```
#include <common_typedef_sdk.h>
```

Data Fields

- float re
- float im

6.198.1 Detailed Description

Defines 64-bit complex structure; both real part and image part have 32 bit width.

6.198.2 Field Documentation

6.198.2.1 im

```
float complex_float::im
```

32-bit image part

6.198.2.2 re

```
float complex_float::re
```

32-bit real part

6.199 complex_half Struct Reference

```
#include <common_typedef_sdk.h>
```

Data Fields

- half re
- half im

6.199.1 Detailed Description

Defines 32-bit complex structure; both real part and image part have 16 bit width.

6.199.2 Field Documentation

6.199.2.1 im

```
half complex_half::im
```

16-bit image part

6.199.2.2 re

```
half complex_half::re
```

16-bit real part

6.200 complex_int16_t Struct Reference

```
#include <common_typedef_sdk.h>
```

Data Fields

- int16_t re
- int16_t im

6.200.1 Detailed Description

Defines 32-bit complex structure; both real part and image part have 16 bit width.

Same defines as COMPLEX16

6.200.2 Field Documentation

6.200.2.1 im

```
int16_t complex_int16_t::im
```

16-bit image part

6.200.2.2 re

`int16_t complex_int16_t::re`

16-bit real part

6.201 complex_int32_t Struct Reference

`#include <common_typedef_sdk.h>`

Data Fields

- int32_t re
- int32_t im

6.201.1 Detailed Description

Defines 64-bit complex structure; both real part and image part have 32 bit width.

6.201.2 Field Documentation

6.201.2.1 im

`int32_t complex_int32_t::im`

32-bit image part

6.201.2.2 re

`int32_t complex_int32_t::re`

32-bit real part

6.202 Matrix Struct Reference

`#include <phy_remapping_ctrlch.h>`

Data Fields

- complex_int16_t ∗ pData
- uint32_t offset
- int16_t nRow
- int16_t nCol
- unsigned char scale

6.202.1 Detailed Description

This structure is a Matrix data structure.

6.202.2 Field Documentation

6.202.2.1 nCol

`int16_t Matrix::nCol`

number of column.

6.202.2.2 nRow

`int16_t Matrix::nRow`

number of row.

6.202.2.3 offset

`uint32_t Matrix::offset`

offset.

6.202.2.4 pData

`complex_int16_t* Matrix::pData`

data in matrix.

6.202.2.5 scale

`unsigned char Matrix::scale`

scale factor for fixed point.

6.203 reMappingInput Struct Reference

```
#include <phy_remapping_pdsch.h>
```

Data Fields

- enSymbolPatternType SymbType
- int vals [3]
- unsigned int offset [5]

6.203.1 Detailed Description

Defined input type of RE mapping.

Vals[] were klen, dPos, ReIdx for ReMappingBySymb/ReMappingBySymb2Tx offset[0] was the input offset, and offset[1-4] were the putput offset for 4 antennas.

6.203.2 Field Documentation

6.203.2.1 offset

```
unsigned int reMappingInput::offset[5]
```

SrcOff, DesOff.

6.203.2.2 SymbType

```
enSymbolPatternType reMappingInput::SymbType
```

Symbol type.

6.203.2.3 vals

```
int reMappingInput::vals[3]
```

{klen, dPos, ReIdx}.

Chapter 7

File Documentation

7.1 bblib_common_const.h File Reference

Enumerations

- enum bblib_common_const_wireless_params { BBLIB_N_SC_PER_PRB = 12, BBLIB_N_SYMB_PER_SF = 14 }
- enum mmse_mimo_constants {
  BBLIB_MAX_RX_ANT_NUM = 16, BBLIB_MAX_TX_LAYER_NUM = 16, BBLIB_RX_DATA_FIXED_POINT = 13, BBLIB_MMSE_X_LEFT_SHIFT = BBLIB_RX_DATA_FIXED_POINT,
  BBLIB_MMSE_LEMMA_SCALING = ((BBLIB_RX_DATA_FIXED_POINT)∗2), BBLIB_MAX_MU = 4 }

7.1.1 Detailed Description

This header file defines common global constants uses throughout the bblib libraries.

7.1.2 Enumeration Type Documentation

7.1.2.1 bblib_common_const_wireless_params

```
enum bblib_common_const_wireless_params
```

This enum contains the common wireless constants accross both LTE and 5G used throughout the bblib libraries.

Enumerator

BBLIB_N_SC_PER_PRB	Number of subcarriers in a Physical Resource Block
BBLIB_N_SYMB_PER_SF	Number of symbols in sub-frame

7.1.2.2 mmse_mimo_constants

```
enum mmse_mimo_constants
```

Constants used in MMSE MIMO.

Enumerator

BBLIB_MAX_RX_ANT_NUM	MAX number of Rx antennas
BBLIB_MAX_TX_LAYER_NUM	MAX number of Tx layers
BBLIB_RX_DATA_FIXED_POINT	Fixed point of Rx data
BBLIB_MMSE_X_LEFT_SHIFT	MMSE X left shift
BBLIB_MMSE_LEMMA_SCALING	MMSE Lemma scaling
BBLIB_MAX_MU	MAX number of multiple user pair

7.2 bit_reverse.h File Reference

Functions

- int16_t bblib_bit_reverse (int8_t ∗inout, int32_t num_data)
- void **bblib_bit_reverse_avx2** (int8_t ∗inout, int32_t num_data)
- void **bblib_bit_reverse_c** (int8_t ∗inout, int32_t num_data)
- void **bblib_bit_reverse_avx512** (int8_t ∗inout, int32_t num_data)

7.2.1 Detailed Description

Source code of conversion between float and int16, with agc gain.

7.2.2 Function Documentation

7.2.2.1 bblib_bit_reverse()

```
int16_t bblib_bit_reverse (
            int8_t * inout,
            int32_t num_data )
```

Bit Reversion.

Parameters

out	*inout*	Input and Output buffer
in	*num_data*	Number of data in bits for conversion.

Returns

Return 0 for success, and -1 for error.

Note

Input and output is aligned with 512 bits.

7.3 common_typedef_sdk.h File Reference

Data Structures

- struct COMPLEX32
- struct complex_int16_t
- struct complex_int32_t
- struct complex_float
- struct complex_double
- struct complex_half

Typedefs

- typedef int16_t half

- typedef struct complex_int16_t **COMPLEX16**

Enumerations

- enum instruction_cpu_support {
  CPU_GENERIC, SSE4_2, AVX, AVX2,
  AVX_512, SNC }
- enum bblib_modulation_order {
  BBLIB_HALF_PI_BPSK = 1, BBLIB_QPSK = 2, BBLIB_PAM4 = 3, BBLIB_QAM16 = 4,
  BBLIB_PAM8 = 5, BBLIB_QAM64 = 6, BBLIB_PAM16 = 7, BBLIB_QAM256 = 8 }

7.3.1 Detailed Description

This header file defines those data type both used by eNB and UE.

7.3.2 Typedef Documentation

7.3.2.1 half

half

half is a 16-bit IEEE floating-point standard number format.

Note

In future this will be known as 'short float' or '__fp16'.
Older compilers must provide proxy support for it as a plain 16-bit integer

7.3.3 Enumeration Type Documentation

7.3.3.1 bblib_modulation_order

enum bblib_modulation_order

Common enums for modulation order.

Enumerator

BBLIB_HALF_PI_BPSK	PI/2 BPSK
BBLIB_QPSK	BPSK QPSK
BBLIB_PAM4	PAM4
BBLIB_QAM16	QAM16
BBLIB_PAM8	PAM8
BBLIB_QAM64	QAM64
BBLIB_PAM16	PAM16
BBLIB_QAM256	QAM256

7.3.3.2 instruction_cpu_support

enum instruction_cpu_support

Define instruction the CPU can support.

Enumerator

CPU_GENERIC	C
SSE4_2	SSE4_2
AVX	AVX
AVX2	AVX2
AVX_512	AVX512
SNC	Sunny Cove Instructions (for ICX)

7.4 float_int16_convert_agc.h File Reference

Functions

- int16_t bblib_float_to_int16_agc (int16_t ∗output, float ∗input, int32_t num_data, float gain)
- void **bblib_float_to_int16_agc_avx2** (int16_t ∗output, float ∗input, int32_t num_data, float gain)
- void **bblib_float_to_int16_agc_c** (int16_t ∗output, float ∗input, int32_t num_data, float gain)
- void **bblib_float_to_int16_agc_avx512** (int16_t ∗output, float ∗input, int32_t num_data, float gain)

- int16_t bblib_float_to_int16_agc_threshold (int16_t ∗output, float ∗input, int32_t num_data, float gain, int16↵_t threshold)
- int16_t **bblib_float_to_int16_agc_threshold_avx2** (int16_t ∗output, float ∗input, int32_t num_data, float gain, int16_t threshold)
- int16_t **bblib_float_to_int16_agc_threshold_c** (int16_t ∗output, float ∗input, int32_t num_data, float gain, int16_t threshold)
- int16_t **bblib_float_to_int16_agc_threshold_avx512** (int16_t ∗output, float ∗input, int32_t num_data, float gain, int16_t threshold)

- int16_t bblib_int16_to_float_agc (float ∗output, int16_t ∗input, int32_t num_data, float gain)
- void **bblib_int16_to_float_agc_avx2** (float ∗output, int16_t ∗input, int32_t num_data, float gain)
- void **bblib_int16_to_float_agc_c** (float ∗output, int16_t ∗input, int32_t num_data, float gain)
- void **bblib_int16_to_float_agc_avx512** (float ∗output, int16_t ∗input, int32_t num_data, float gain)

- int16_t bblib_int16_to_int16_agc (int16_t ∗output, int16_t ∗input, int32_t num_data, float gain)
- void **bblib_int16_to_int16_agc_avx2** (int16_t ∗output, int16_t ∗input, int32_t num_data, float gain)
- void **bblib_int16_to_int16_agc_c** (int16_t ∗output, int16_t ∗input, int32_t num_data, float gain)
- void **bblib_int16_to_int16_agc_avx512** (int16_t ∗output, int16_t ∗input, int32_t num_data, float gain)

- int16_t bblib_int16_to_int16_fxp_scale (int16_t ∗scaleOut, int16_t ∗scaleIn, int32_t num_samples, int16_t scale16)
- void **bblib_int16_to_int16_fxp_scale_c** (int16_t ∗scaleOut, int16_t ∗scaleIn, int32_t num_samples, int16↵_t scale16)
- void **bblib_int16_to_int16_fxp_scale_avx512** (int16_t ∗scaleOut, int16_t ∗scaleIn, int32_t num_samples, int16_t scale16)

7.4.1 Detailed Description

Source code of conversion between float and int16, with agc gain.

7.4.2 Function Documentation

7.4.2.1 bblib_float_to_int16_agc()

```
int16_t bblib_float_to_int16_agc (
            int16_t * output,
            float * input,
            int32_t num_data,
            float gain )
```

Conversion from float to int16, with float gain.

Parameters

in	*input*	Input buffer for float.
in	*num_data*	Number of data for conversion.
in	*gain*	Gain for agc.
out	*output*	Output buffer for int16.

Returns

Return 0 for success, and -1 for error.

Note

Input and output is aligned with 512 bits.
Also (input data∗gain) should be in the rage of -32768∼32767, for the range of int16.

7.4.2.2 bblib_float_to_int16_agc_threshold()

```
int16_t bblib_float_to_int16_agc_threshold (
            int16_t * output,
            float * input,
            int32_t num_data,
            float gain,
            int16_t threshold )
```

Conversion from float to int16, with float gain and int16 threshold.

Parameters

in	*input*	Input buffer for float.
in	*num_data*	Number of data for conversion.
in	*gain*	Gain for agc.
in	*threshold*	Threshold after agc, which should be >=0.
out	*output*	Output buffer for int16.

Returns

Return 0 for success, and -1 for error.

Note

Input and output is aligned with 512 bits.

7.4.2.3 bblib_int16_to_float_agc()

```
int16_t bblib_int16_to_float_agc (
            float * output,
            int16_t * input,
            int32_t num_data,
            float gain )
```

Conversion from int16 to float, with float gain.

Parameters

in	*input*	Input buffer for int16.
in	*num_data*	Number of data for conversion.
in	*gain*	Gain for agc.
out	*output*	Output buffer for float.

Returns

Return 0 for success, and -1 for error.

Note

Input and output is aligned with 512 bits.

7.4.2.4 bblib_int16_to_int16_agc()

```
int16_t bblib_int16_to_int16_agc (
            int16_t * output,
            int16_t * input,
            int32_t num_data,
            float gain )
```

Conversion from int16 to int16, with float gain.

Parameters

in	*input*	Input buffer for int16.
in	*num_data*	Number of data for conversion.
in	*gain*	Gain for agc.
out	*output*	Output buffer for int16.

Returns

Return 0 for success, and -1 for error.

Note

Input and output is aligned with 512 bits.

7.4.2.5 bblib_int16_to_int16_fxp_scale()

```
int16_t bblib_int16_to_int16_fxp_scale (
            int16_t * scaleOut,
            int16_t * scaleIn,
            int32_t num_samples,
            int16_t scale16 )
```

add fxp scale to int16

Parameters

in	*scaleIn*	scaling input
in	*num_samples*	number of samples
in	*scale16*	scaling in int16
out	*scaleOut*	scaling output

Returns

none

7.5 phase_noise_5gnr.h File Reference

Data Structures

- struct bblib_phase_noise_compensation_5gnr_request
- struct bblib_phase_noise_compensation_5gnr_response
- struct bblib_phase_noise_estimation_5gnr_request
- struct bblib_phase_noise_estimation_5gnr_response

Enumerations

- enum bblib_phase_noise_config { BBLIB_PHASE_NOISE_MAX_RX_ANT = 8 }

Functions

- int16_t bblib_phase_noise_5gnr_version (char *version, int buffer_size)

- int32_t bblib_ptrs_phase_noise_compensation_5gnr (struct bblib_phase_noise_compensation_5gnr_request *request, struct bblib_phase_noise_compensation_5gnr_response *response)
- int32_t **bblib_ptrs_phase_noise_compensation_5gnr_avx512** (struct bblib_phase_noise_compensation_5gnr_request *request, struct bblib_phase_noise_compensation_5gnr_response *response)
- int32_t bblib_ptrs_phase_noise_estimation_5gnr (struct bblib_phase_noise_estimation_5gnr_request *request, struct bblib_phase_noise_estimation_5gnr_response *response)
- int32_t **bblib_ptrs_phase_noise_estimation_5gnr_avx512** (struct bblib_phase_noise_estimation_5gnr_request *request, struct bblib_phase_noise_estimation_5gnr_response *response)

7.5.1 Detailed Description

External API for 5GNR Phase Noise estimation and compensation.

Overview:

The lib_phase_noise_5gnr kernel is for 5G NR phase noise estimation and compensation. This is a generic kernel which can be used for either the UL (non-transform precoding mode) and DL phase noise estimation and compensation PTRS (Phase tracking reference signal) is defined in TS38.211 section 6.4.1.2 and 7.4.1.2 (v15.2.0).

Algorithm Guidance:

The phase noise compensation is Output=Input*conj(Phase_noise), where phase noise is estimated in phase noise estimation function.

7.5.2 Enumeration Type Documentation

7.5.2.1 bblib_phase_noise_config

```
enum bblib_phase_noise_config
```

max receving antenna number

Enumerator

BBLIB_PHASE_NOISE_MAX_RX_ANT	Max number of receiving antennas

7.5.3 Function Documentation

7.5.3.1 bblib_phase_noise_5gnr_version()

```
int16_t bblib_phase_noise_5gnr_version (
            char * version,
            int buffer_size )
```

Report the version number for the phase noise library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, typically no more than BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.5.3.2 bblib_ptrs_phase_noise_compensation_5gnr()

```
int32_t bblib_ptrs_phase_noise_compensation_5gnr (
            struct bblib_phase_noise_compensation_5gnr_request * request,
            struct bblib_phase_noise_compensation_5gnr_response * response )
```

This function implements phase noise compensation with AVX512 instructions.

Parameters

in	*request*	Input struct of phase noise compensation
out	*response*	Output struct of phase noise compensation

Returns

0 for success, and -1 for error

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.5.3.3 bblib_ptrs_phase_noise_estimation_5gnr()

```
int32_t bblib_ptrs_phase_noise_estimation_5gnr (
            struct bblib_phase_noise_estimation_5gnr_request * request,
            struct bblib_phase_noise_estimation_5gnr_response * response )
```

This function implements phase noise estimation with AVX512 instructions.

Parameters

in	*request*	Input struct of phase noise estimation
out	*response*	Output struct of phase noise estimation

Returns

0 for success, and -1 for error

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.6 phy_beamforming_dl_expand.h File Reference

Data Structures

- struct bblib_beamforming_dl_expand_request
- struct bblib_beamforming_dl_expand_response

Enumerations

- enum beamforming_dl_expand_constants { BBLIB_BF_MAX_RX_ANT_NUM = 64, BBLIB_BF_MAX_UL_LAYER_NUM = 8, BBLIB_BF_MAX_DL_LAYER_NUM = 16 }

Functions

- int16_t bblib_beamforming_dl_expand_version (char *version, int buffer_size)
- int32_t bblib_beamforming_dl_expand (const bblib_beamforming_dl_expand_request *request, bblib_beamforming_dl_expand *response)
- int32_t **bblib_beamforming_dl_expand_avx512** (const bblib_beamforming_dl_expand_request *request, bblib_beamforming_dl_expand_response *response)

7.6.1 Detailed Description

External API for DL expanding for beamforming in 5GNR.

Overview: This module implements DL expanding for beamforming in 5GNR. It can support below configuration:

1. expanding from nRx X nTx to (2*nTx) X nRx, nRx = 32, nTx = 4 and nRx = 64, nTx = 8/4/2/1.

Algorithm Guidance:

1. Input matrix H_ul is nRx X nTx
2. Seperate H_ul into 2 matrices H1((nRx/2) X nTx), H2((nRx/2) X nTx)
3. Calculate H_dl = [H1 0; 0 H2] * Ror where Ror is a unit orthogonal matrix [1,1; 1,-1]

7.6.2 Enumeration Type Documentation

7.6.2.1 beamforming_dl_expand_constants

```
enum beamforming_dl_expand_constants
```

Constants used in DL expanding for beamforming.

Enumerator

BBLIB_BF_MAX_RX_ANT_NUM	MAX number of Rx antennas
BBLIB_BF_MAX_UL_LAYER_NUM	MAX number of UL layers
BBLIB_BF_MAX_DL_LAYER_NUM	MAX number of DL layers

7.6.3 Function Documentation

7.6.3.1 bblib_beamforming_dl_expand()

```
int32_t bblib_beamforming_dl_expand (
            const bblib_beamforming_dl_expand_request * request,
            bblib_beamforming_dl_expand_response * response )
```

beamforming_dl_expand.

Parameters

in	*request*	Input request structure .
out	*response*	Output response structure.

Returns

0 for success, and -1 for error

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.6.3.2 bblib_beamforming_dl_expand_version()

```
int16_t bblib_beamforming_dl_expand_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_beamforming_dl_expand library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.7 phy_cestimate.h File Reference

Data Structures

- struct bblib_cestmate_request
- struct bblib_cestmate_response

Functions

- int16_t bblib_cestimate_version (char ∗version, int buffer_size)
- int32_t bblib_lte_ChannelEstimation (struct bblib_cestmate_request ∗request, struct bblib_cestmate_response ∗response)

7.7.1 Detailed Description

External API for 4G channel estimate functions.

7.7.2 Function Documentation

7.7.2.1 bblib_cestimate_version()

```
int16_t bblib_cestimate_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_cestimate_version library.

Parameters

in	*version*	pointer to a char buffer where the version string should be copied.
in	*buffer_size*	the length of the string buffer, typically no more than BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.7.2.2 bblib_lte_ChannelEstimation()

```
int32_t bblib_lte_ChannelEstimation (
            struct bblib_cestmate_request * request,
            struct bblib_cestmate_response * response )
```

Channle Estimation procedure.

Parameters

in	*request*	Structure containing the input data, selected data type,the length of input data and the value of init.\
out	*response*	Structure containing the output data and length of it.

Note

This function is only used for eNobe UL channel.

Returns

0 for success, -1 for error.

7.8 phy_cestimate_5gnr.h File Reference

Data Structures

- struct bblib_channel_estimation_5gnr_request
- struct bblib_channel_estimation_5gnr_response

Functions

- void bblib_print_cestimate_5gnr_version ()
- int16_t bblib_cestimate_5gnr_version (char ∗version, int buffer_size)

- void bblib_cestimate_5gnr (struct bblib_channel_estimation_5gnr_request ∗request, struct bblib_channel_estimation_5gnr_resp ∗response)
- void **bblib_cestimate_5gnr_avx512** (struct bblib_channel_estimation_5gnr_request ∗p_ce_req, struct bblib_channel_estimation_5gnr_response ∗p_ce_resp)
- void **bblib_channel_estimation_ls_5gnr_type1** (struct bblib_channel_estimation_5gnr_request ∗p_ce_↩ req, struct bblib_channel_estimation_5gnr_response ∗p_ce_resp)

- void **bblib_channel_estimation_ls_5gnr_type2** (struct bblib_channel_estimation_5gnr_request ∗p_ce_↩ req, struct bblib_channel_estimation_5gnr_response ∗p_ce_resp)

- void bblib_cestimate_dct_5gnr (struct bblib_channel_estimation_5gnr_request ∗request, struct bblib_channel_estimation_5gnr_ ∗response)
- void **bblib_cestimate_dct_5gnr_avx512** (struct bblib_channel_estimation_5gnr_request ∗p_ce_req, struct bblib_channel_estimation_5gnr_response ∗p_ce_resp)

7.8.1 Detailed Description

External API for channel estiation and timing advance estimation for 5GNR.

Overview:

The lib_cestimate_5gnr kernel is a 5G NR channel estimate functions with Wiener filter or DCT. This is a generic kernel which can be used to channel estimate in both UL and DL.

Algorithm Guidance:

The 5G NR channel estimator algorithm (Wiener) can be broken down into the following steps:
1.Estimate reference signal channel using Least Square algorithm (per layer per antenna).
2.Generate interpolation weight and do the interpolation in frequency domain for all subcarriers (per layer per antenna).
3.Estimate noise power (per layer per antenna).
4.Two-iteration channel estimation. Repeat step 2 and step 3 (per antenna).
The 5G NR channel estimator algorithm (Wiener) can be broken down into the following steps:
1.Estimate reference signal channel using Least Square algorithm (per layer per antenna).
2.DCT (per layer per antenna).
3.noise esimation and cancellation (per layer per antenna).
4.IDCT (per layer per antenna).
5.frequency domain interpolation.(per layer per antenna)

7.8.2 Function Documentation

7.8.2.1 bblib_cestimate_5gnr()

```
void bblib_cestimate_5gnr (
            struct bblib_channel_estimation_5gnr_request * request,
            struct bblib_channel_estimation_5gnr_response * response )
```

cestimate_5gnr procedures.

Parameters

in	*request*	Structure containing the input data which need to be 64 bytes alignment.
out	*response*	Structure containing the decoding output data which need 64 byte alignment.

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.8.2.2 bblib_cestimate_5gnr_version()

```
int16_t bblib_cestimate_5gnr_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_cestimate_5gnr library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.8.2.3 bblib_cestimate_dct_5gnr()

```
void bblib_cestimate_dct_5gnr (
            struct bblib_channel_estimation_5gnr_request * request,
            struct bblib_channel_estimation_5gnr_response * response )
```

cestimate_dct_5gnr procedures.

Parameters

in	*request*	Structure containing the input data which need to be 64 bytes alignment.
out	*response*	Structure containing the decoding output data which need 64 byte alignment.

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.8.2.4 bblib_print_cestimate_5gnr_version()

```
void bblib_print_cestimate_5gnr_version ( )
```

printf cestimate 5gnr version

Returns

null.

7.9 phy_cestimate_pucch.h File Reference

Data Structures

- struct bblib_pucch_ndash_request
- struct bblib_pucch_ndash_response
- struct bblib_cestimate_pucch_pilot_mul_request
- struct bblib_cestimate_pucch_pilot_mul_response
- struct bblib_cestimate_pucch_part1_request
- struct bblib_cestimate_pucch_part1_response

Enumerations

- enum cestimate_pucch_constants {
  NRB_SC = 12, MAX_NUM_ANT_CEST_PUCCH = 8, NUM_SLOTS_PER_SUBF_CEST_PUCCH = 2,
  MAX_NUM_PILOT_SYM_PER_SLOT = 3,
  MAX_NUM_SUBCARRIERS = 1200, MAX_NUM_PUCCH_SDK = 160, **MAX_NUM_SLOTS** = 20,
  MAX_SYM_PER_SUBFRAME_SDK = 14,
  NUM_CQI_DATA_SYM_PER_SUBFRAME = 10 }
- enum CP_MODE { NORMAL = 0, EXTENDED = 1 }
- enum cestimate_pucch_phy_formats {
  phy_format_1 = 0, **phy_format_1A** = 1, **phy_format_1B** = 2, **phy_format_2** = 3,
  phy_format_2A = 4, **phy_format_2B** = 5, **phy_spacial_bundling** = 6, **phy_format_3** = 7 }

Functions

- int16_t bblib_cestimate_pucch_version (char ∗version, int buffer_size)

- void bblib_ndash_calculation_format1 (const struct bblib_pucch_ndash_request ∗request, struct bblib_pucch_ndash_response ∗response)
- void **bblib_ndash_calculation_format1_c** (const struct bblib_pucch_ndash_request ∗request, struct bblib_pucch_ndash_response ∗response)
- void **bblib_ndash_calculation_format1_avx2** (const struct bblib_pucch_ndash_request ∗request, struct bblib_pucch_ndash_response ∗response)
- void **bblib_ndash_calculation_format1_avx512** (const struct bblib_pucch_ndash_request ∗request, struct bblib_pucch_ndash_response ∗response)

- void bblib_ndash_calculation_format2 (const struct bblib_pucch_ndash_request ∗request, struct bblib_pucch_ndash_response ∗response)
- void **bblib_ndash_calculation_format2_c** (const struct bblib_pucch_ndash_request ∗request, struct bblib_pucch_ndash_response ∗response)

- void **bblib_ndash_calculation_format2_avx2** (const struct bblib_pucch_ndash_request ∗request, struct bblib_pucch_ndash_response ∗response)
- void **bblib_ndash_calculation_format2_avx512** (const struct bblib_pucch_ndash_request ∗request, struct bblib_pucch_ndash_response ∗response)

- void bblib_cestimate_pucch_pilot_mul_fxp (const struct bblib_cestimate_pucch_pilot_mul_request ∗request, struct bblib_cestimate_pucch_pilot_mul_response ∗response)
- void **bblib_cestimate_pucch_pilot_mul_fxp_c** (const struct bblib_cestimate_pucch_pilot_mul_request ∗request, struct bblib_cestimate_pucch_pilot_mul_response ∗response)
- void **bblib_cestimate_pucch_pilot_mul_fxp_avx2** (const struct bblib_cestimate_pucch_pilot_mul_request ∗request, struct bblib_cestimate_pucch_pilot_mul_response ∗response)
- void **bblib_cestimate_pucch_pilot_mul_fxp_avx512** (const struct bblib_cestimate_pucch_pilot_mul_request ∗request, struct bblib_cestimate_pucch_pilot_mul_response ∗response)

- void bblib_cestimate_pucch_pilot_mul_flp (const struct bblib_cestimate_pucch_pilot_mul_request ∗request, struct bblib_cestimate_pucch_pilot_mul_response ∗response)
- void **bblib_cestimate_pucch_pilot_mul_flp_c** (const struct bblib_cestimate_pucch_pilot_mul_request ∗request, struct bblib_cestimate_pucch_pilot_mul_response ∗response)
- void **bblib_cestimate_pucch_pilot_mul_flp_avx2** (const struct bblib_cestimate_pucch_pilot_mul_request ∗request, struct bblib_cestimate_pucch_pilot_mul_response ∗response)

- void bblib_cestimate_pucch_part1 (const struct bblib_cestimate_pucch_part1_request ∗request, struct bblib_cestimate_pucch_part1_response ∗response)
- void **bblib_cestimate_pucch_part1_flp** (const struct bblib_cestimate_pucch_part1_request ∗request, struct bblib_cestimate_pucch_part1_response ∗response)
- void **bblib_cestimate_pucch_part1_flp_c** (const struct bblib_cestimate_pucch_part1_request ∗request, struct bblib_cestimate_pucch_part1_response ∗response)
- void **bblib_cestimate_pucch_part1_flp_avx2** (const struct bblib_cestimate_pucch_part1_request ∗request, struct bblib_cestimate_pucch_part1_response ∗response)

7.9.1 Detailed Description

External API for LTE PUCCH Channel Estimator.

7.9.2 Enumeration Type Documentation

7.9.2.1 cestimate_pucch_constants

`enum cestimate_pucch_constants`

This configuration sets global constants and macros which are of general use throughout the module.

Enumerator

NRB_SC	Number of subcarriers per resource block.
MAX_NUM_ANT_CEST_PUCCH	Maximum number of antennas supported
NUM_SLOTS_PER_SUBF_CEST_PUCCH	Number of slots per subframe
MAX_NUM_PILOT_SYM_PER_SLOT	Maximum number of pilot symbols per slot in PUCCH
MAX_NUM_SUBCARRIERS	Maximum number of subcarriers
MAX_NUM_PUCCH_SDK	Maximum number of resource indices used for each PUCCH
MAX_SYM_PER_SUBFRAME_SDK	Max number of symbols per subframe
NUM_CQI_DATA_SYM_PER_SUBFRAME	Number of CQI symbols per subframe

7.9.2.2 CP_MODE

```
enum CP_MODE
```

Definition of available cyclic prefix modes as defined in 36.211, Chapter 5.4.1.

Enumerator

NORMAL	Normal cyclic prefix.
EXTENDED	Extended cyclic prefix.

7.9.3 Function Documentation

7.9.3.1 bblib_cestimate_pucch_part1()

```
void bblib_cestimate_pucch_part1 (
            const struct bblib_cestimate_pucch_part1_request * request,
            struct bblib_cestimate_pucch_part1_response * response )
```

LTE PUCCH Channel Estimator part 1.

Parameters

in	*request*	Request structure with required input information
out	*response*	Pilot positions for requested number of resources

7.9.3.2 bblib_cestimate_pucch_pilot_mul_flp()

```
void bblib_cestimate_pucch_pilot_mul_flp (
```

```
            const struct bblib_cestimate_pucch_pilot_mul_request * request,
            struct bblib_cestimate_pucch_pilot_mul_response * response )
```

Floating point first part of PUCCH channel estimation: pilot symbols by locally generated pilots multiplication as defined in 36.211, Chapter 5.4.

Parameters

in	*request*	Request structure with required input information
out	*response*	Structure containing the output data

7.9.3.3 bblib_cestimate_pucch_pilot_mul_fxp()

```
void bblib_cestimate_pucch_pilot_mul_fxp (
            const struct bblib_cestimate_pucch_pilot_mul_request * request,
            struct bblib_cestimate_pucch_pilot_mul_response * response )
```

Fixed point first part of PUCCH channel estimation: pilot symbols by locally generated pilots multiplication as defined in 36.211, Chapter 5.4.

Parameters

in	*request*	Request structure with required input information
out	*response*	Structure containing the output data

7.9.3.4 bblib_cestimate_pucch_version()

```
int16_t bblib_cestimate_pucch_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_cestimate_pucch library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, typically no more than BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.9.3.5 bblib_ndash_calculation_format1()

```
void bblib_ndash_calculation_format1 (
            const struct bblib_pucch_ndash_request * request,
            struct bblib_pucch_ndash_response * response )
```

Calculation of pilot positions in the formats 1/1a/1b PUCCH channel as defined in 36.211, Chapter 5.4.

Parameters

in	*request*	Request structure with required input information
out	*response*	Pilot positions for requested number of resources

7.9.3.6 bblib_ndash_calculation_format2()

```
void bblib_ndash_calculation_format2 (
            const struct bblib_pucch_ndash_request * request,
            struct bblib_pucch_ndash_response * response )
```

Calculation of pilot positions in the formats 2/2a/2b PUCCH channel as defined in 36.211, Chapter 5.4.

Parameters

in	*request*	Request structure with required input information
out	*response*	Pilot positions for requested number of resources

7.10 phy_companding.h File Reference

Data Structures

- struct bblib_compress_request
- struct bblib_compress_response
- struct bblib_decompress_request
- struct bblib_decompress_response

Functions

- int16_t bblib_companding_version (char *version, int buffer_size)

- int bblib_compress (const struct bblib_compress_request *request, struct bblib_compress_response *response)

- int **bblib_compress_sse** (const struct bblib_compress_request *request, struct bblib_compress_response *response)
- int **bblib_compress_avx2** (const struct bblib_compress_request *request, struct bblib_compress_response *response)
- int **bblib_compress_avx512** (const struct bblib_compress_request *request, struct bblib_compress_response *response)

- int bblib_decompress (const struct bblib_decompress_request *request, struct bblib_decompress_response *response)
- int **bblib_decompress_sse** (const struct bblib_decompress_request *request, struct bblib_decompress_response *response)
- int **bblib_decompress_avx2** (const struct bblib_decompress_request *request, struct bblib_decompress_response *response)
- int **bblib_decompress_avx512** (const struct bblib_decompress_request *request, struct bblib_decompress_response *response)

7.10.1 Detailed Description

External API for compading with the use of A-law algorithm.

7.10.2 Function Documentation

7.10.2.1 bblib_companding_version()

```
int16_t bblib_companding_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_companding library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.10.2.2 bblib_compress()

```
int bblib_compress (
            const struct bblib_compress_request * request,
            struct bblib_compress_response * response )
```

Compress functions - it converts a 16-bit linear PCM value to 8-bt A-law.

Parameters

in	*request*	Structure containing the input data and data length.
out	*response*	Structure containing the output data and data length.

Returns

0 for success, -1 for error

7.10.2.3 bblib_decompress()

```
int bblib_decompress (
            const struct bblib_decompress_request * request,
            struct bblib_decompress_response * response )
```

Decompress function - it converts an A-law value to 16-bit linear PCM.

Parameters

in	*request*	Structure containing the input data and data length.
out	*response*	Structure containing the output data and data length.

Returns

0 for success, -1 for error.

7.11 phy_crc.h File Reference

Data Structures

- struct bblib_crc_request
- struct bblib_crc_response

Functions

- int16_t bblib_lte_crc_version (char *version, int buffer_size)

- void bblib_lte_crc24a_gen (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc24a_gen_avx512** (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc24a_gen_snc** (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc24a_gen_sse** (struct bblib_crc_request *request, struct bblib_crc_response *response)

- void bblib_lte_crc24a_check (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc24a_check_avx512** (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc24a_check_snc** (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc24a_check_sse** (struct bblib_crc_request *request, struct bblib_crc_response *response)

- void bblib_lte_crc24b_gen (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc24b_gen_avx512** (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc24b_gen_snc** (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc24b_gen_sse** (struct bblib_crc_request *request, struct bblib_crc_response *response)

- void bblib_lte_crc24b_check (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc24b_check_avx512** (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc24b_check_snc** (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc24b_check_sse** (struct bblib_crc_request *request, struct bblib_crc_response *response)

- void bblib_lte_crc24c_gen (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc24c_gen_avx512** (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc24c_gen_snc** (struct bblib_crc_request *request, struct bblib_crc_response *response)

- void bblib_lte_crc24c_check (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc24c_check_avx512** (struct bblib_crc_request *request, struct bblib_crc_response *response)

- void **bblib_lte_crc24c_check_snc** (struct bblib_crc_request *request, struct bblib_crc_response *response)

- void bblib_lte_crc24c_1_gen (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc24c_1_gen_avx512** (struct bblib_crc_request *request, struct bblib_crc_response *response)

- void bblib_lte_crc24c_1_check (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc24c_1_check_avx512** (struct bblib_crc_request *request, struct bblib_crc_response *response)

- void bblib_lte_crc16_gen (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc16_gen_avx512** (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc16_gen_snc** (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc16_gen_sse** (struct bblib_crc_request *request, struct bblib_crc_response *response)

- void bblib_lte_crc16_check (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc16_check_avx512** (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc16_check_snc** (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc16_check_sse** (struct bblib_crc_request *request, struct bblib_crc_response *response)

- void bblib_lte_crc11_gen (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc11_gen_avx512** (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc11_gen_snc** (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc11_gen_sse** (struct bblib_crc_request *request, struct bblib_crc_response *response)

- void bblib_lte_crc11_check (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc11_check_avx512** (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc11_check_snc** (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc11_check_sse** (struct bblib_crc_request *request, struct bblib_crc_response *response)

- void bblib_lte_crc6_gen (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc6_gen_avx512** (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc6_gen_snc** (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc6_gen_sse** (struct bblib_crc_request *request, struct bblib_crc_response *response)

- void bblib_lte_crc6_check (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc6_check_avx512** (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc6_check_snc** (struct bblib_crc_request *request, struct bblib_crc_response *response)
- void **bblib_lte_crc6_check_sse** (struct bblib_crc_request *request, struct bblib_crc_response *response)

7.11.1 Detailed Description

External API for lib_crc, which comprises of CRC generate and CRC validate functions for the following CRC algorithms as specified in 3GPP TS 38.212 v15.1.1: CRC24A, CRC24B, CRC24C, CRC16, CRC11 & CRC6, all initialised with zeros, and CRC24C initialised with ones as specified in 3GPP TS 38.212 section 7.3.2.

The CRC generate function (bblib_lte_<algorithm>_gen) is used to calculate the CRC value based on the sequence of input data and data length, in bits passed to the function in the request structure. The CRC value is then appended to the end of the input data sequence and available in the response structure. Due to the nature of the algorithms being byte based, maximum performance is obtained when input data length is in multiples of 8 bits (ie. bytes), since padding of the data isn't required.

The CRC validate function (bblib_lte_<algorithm>_check) is used to validate input data that already contains a CRC appended to the end. It calculates a CRC value and then compares that value to the one at the end of the data. The result (pass or fail) is indicated in the response structure.

Testing: Each CRC algorithm's generate & validate function is tested over a range of test vectors from 1 to 65536 bits, with both multiples and non-multiples of 8 bits. A series of performance tests have also been defined generally based on 2344 & 2340 bit test vectors.

7.11.2 Function Documentation

7.11.2.1 bblib_lte_crc11_check()

```
void bblib_lte_crc11_check (
            struct bblib_crc_request * request,
            struct bblib_crc_response * response )
```

Performs CRC11 validate, indicating if the input data sequence has a valid CRC value.

Parameters

in	*request*	structure containing pointer to input data and data length.

Note

Length should be for the data part only, since the CRC algorithm determines the CRC length.

Parameters

out	*response*	structure with indication if CRC validation has passed.

Returns

void.

Note

Memory for both request.data & response.data structures should be allocated. Response.data can point to request.data if sufficient space is available at end of request.data structure for the appended CRC.
CRC polynomial is "D11 + D10 + D9 + D5 + 1", refer to 3GPP TS 38.212, section 5.1.

7.11.2.2 bblib_lte_crc11_gen()

```
void bblib_lte_crc11_gen (
            struct bblib_crc_request * request,
            struct bblib_crc_response * response )
```

Performs CRC11 generate, calculating the CRC value and appending to the data.

Parameters

in	*request*	structure containing pointer to input data and data length.
out	*response*	structure containing calculated CRC value, CRC appended data and new length.

Returns

void

Note

Memory for both request.data & response.data structures should be allocated. Response.data can point to request.data if sufficient space is available at end of request.data structure for the appended CRC.
CRC polynomial is "D11 + D10 + D9 + D5 + 1", refer to 3GPP TS 38.212, section 5.1.

7.11.2.3 bblib_lte_crc16_check()

```
void bblib_lte_crc16_check (
            struct bblib_crc_request * request,
            struct bblib_crc_response * response )
```

Performs CRC16 validate, indicating if the input data sequence has a valid CRC value.

Parameters

in	*request*	structure containing pointer to input data and data length.

Note

Length should be for the data part only, since the CRC algorithm determines the CRC length.

Parameters

out	*response*	structure with indication if CRC validation has passed.

Returns

void.

Note

Memory for both request.data & response.data structures should be allocated. Response.data can point to request.data if sufficient space is available at end of request.data structure for the appended CRC.
CRC polynomial is "D16 + D12 + D5 + 1", refer to 3GPP TS 38.212, section 5.1.

7.11.2.4 bblib_lte_crc16_gen()

```
void bblib_lte_crc16_gen (
            struct bblib_crc_request * request,
            struct bblib_crc_response * response )
```

Performs CRC16 generate, calculating the CRC value and appending to the data.

Parameters

in	*request*	structure containing pointer to input data and data length.
out	*response*	structure containing calculated CRC value, CRC appended data and new length.

Returns

void.

Note

Memory for both request.data & response.data structures should be allocated. Response.data can point to request.data if sufficient space is available at end of request.data structure for the appended CRC.
CRC polynomial is "D16 + D12 + D5 + 1", refer to 3GPP TS 38.212, section 5.1.

7.11.2.5 bblib_lte_crc24a_check()

```
void bblib_lte_crc24a_check (
            struct bblib_crc_request * request,
            struct bblib_crc_response * response )
```

Performs CRC24A validate, indicating if the input data sequence has a valid CRC value.

Parameters

in	*request*	structure containing pointer to input data and data length.

Note

Length should be for the data part only, since the CRC algorithm determines the CRC length.

Parameters

out	*response*	structure with indication if CRC validation has passed.

Returns

void

Note

Memory for both request.data & response.data structures should be allocated. Response.data can point to request.data if sufficient space is available at end of request.data structure for the appended CRC.
CRC polynomial is "D24 + D23 + D18 + D17 + D14 + D11 + D10 + D7 + D6 + D5 + D4 + D3 + D + 1", refer to 3GPP TS 38.212, section 5.1.

7.11.2.6 bblib_lte_crc24a_gen()

```
void bblib_lte_crc24a_gen (
            struct bblib_crc_request * request,
            struct bblib_crc_response * response )
```

Performs CRC24A generate, calculating the CRC value and appending to the data.

Parameters

in	*request*	structure containing pointer to input data and data length.
out	*response*	structure containing calculated CRC value, CRC appended data and new length.

Returns

void

Note

Memory for both request.data & response.data structures should be allocated. Response.data can point to request.data if sufficient space is available at end of request.data structure for the appended CRC.
CRC polynomial is "D24 + D23 + D18 + D17 + D14 + D11 + D10 + D7 + D6 + D5 + D4 + D3 + D + 1", refer to 3GPP TS 38.212, section 5.1.

7.11.2.7 bblib_lte_crc24b_check()

```
void bblib_lte_crc24b_check (
            struct bblib_crc_request * request,
            struct bblib_crc_response * response )
```

Performs CRC24B validate, indicating if the input data sequence has a valid CRC value.

Parameters

in	*request*	structure containing pointer to input data and data length.

Note

Length should be for the data part only, since the CRC algorithm determines the CRC length.

Parameters

out	*response*	structure with indication if CRC validation has passed.

Returns

void

Note

Memory for both request.data & response.data structures should be allocated. Response.data can point to request.data if sufficient space is available at end of request.data structure for the appended CRC.
CRC polynomial is "D24 + D23 + D6 + D5 + D + 1", refer to 3GPP TS 38.212, section 5.1.

7.11.2.8 bblib_lte_crc24b_gen()

```
void bblib_lte_crc24b_gen (
            struct bblib_crc_request * request,
            struct bblib_crc_response * response )
```

Performs CRC24B generate, calculating the CRC value and appending to the data.

Parameters

in	*request*	structure containing pointer to input data and data length.
out	*response*	structure containing calculated CRC value, CRC appended data and new length.

Returns

void.

Note

Memory for both request.data & response.data structures should be allocated. Response.data can point to request.data if sufficient space is available at end of request.data structure for the appended CRC.
CRC polynomial is "D24 + D23 + D6 + D5 + D + 1", refer to 3GPP TS 38.212, section 5.1.

7.11.2.9 bblib_lte_crc24c_1_check()

```
void bblib_lte_crc24c_1_check (
            struct bblib_crc_request * request,
            struct bblib_crc_response * response )
```

Performs CRC24C initialised with 1s, validate, indicating if the input data sequence has a valid CRC value.

Parameters

in	*request*	structure containing pointer to input data and data length.

Note

Length should be for the data part only, since the CRC algorithm determines the CRC length.

Parameters

out	*response*	structure with indication if CRC validation has passed.

Returns

void

Note

Memory for both request.data & response.data structures should be allocated. Response.data can point to request.data if sufficient space is available at end of request.data structure for the appended CRC.
CRC polynomial is "D24 + D23 + D21 + D20 + D17 + D15 + D13 + D12 + D8 + D4 + D2 + D + 1", refer to 3GPP TS 38.212, section 5.1.

7.11.2.10 bblib_lte_crc24c_1_gen()

```
void bblib_lte_crc24c_1_gen (
            struct bblib_crc_request * request,
            struct bblib_crc_response * response )
```

Performs CRC24C initialised with 1s, generate, calculating the CRC value and appending to the data.

Parameters

in	*request*	structure containing pointer to input data and data length.
out	*response*	structure containing calculated CRC value, CRC appended data and new length.

Returns

void.

Note

Memory for both request.data & response.data structures should be allocated. Response.data can point to request.data if sufficient space is available at end of request.data structure for the appended CRC.
CRC polynomial is "D24 + D23 + D21 + D20 + D17 + D15 + D13 + D12 + D8 + D4 + D2 + D + 1", refer to 3GPP TS 38.212, section 5.1.

7.11.2.11 bblib_lte_crc24c_check()

```
void bblib_lte_crc24c_check (
            struct bblib_crc_request * request,
            struct bblib_crc_response * response )
```

Performs CRC24C validate, indicating if the input data sequence has a valid CRC value.

Parameters

in	*request*	structure containing pointer to input data and data length.

Note

Length should be for the data part only, since the CRC algorithm determines the CRC length.

Parameters

out	*response*	structure with indication if CRC validation has passed.

Returns

void

Note

Memory for both request.data & response.data structures should be allocated. Response.data can point to request.data if sufficient space is available at end of request.data structure for the appended CRC.
CRC polynomial is "D24 + D23 + D21 + D20 + D17 + D15 + D13 + D12 + D8 + D4 + D2 + D + 1", refer to 3GPP TS 38.212, section 5.1.

7.11.2.12 bblib_lte_crc24c_gen()

```
void bblib_lte_crc24c_gen (
            struct bblib_crc_request * request,
            struct bblib_crc_response * response )
```

Performs CRC24C generate, calculating the CRC value and appending to the data.

Parameters

in	*request*	structure containing pointer to input data and data length.
out	*response*	structure containing calculated CRC value, CRC appended data and new length.

Returns

void.

Note

Memory for both request.data & response.data structures should be allocated. Response.data can point to request.data if sufficient space is available at end of request.data structure for the appended CRC.
CRC polynomial is "D24 + D23 + D21 + D20 + D17 + D15 + D13 + D12 + D8 + D4 + D2 + D + 1", refer to 3GPP TS 38.212, section 5.1.

7.11.2.13 bblib_lte_crc6_check()

```
void bblib_lte_crc6_check (
            struct bblib_crc_request * request,
            struct bblib_crc_response * response )
```

Performs CRC6 validate, indicating if the input data sequence has a valid CRC value.

Parameters

in	*request*	structure containing pointer to input data and data length.

Note

Length should be for the data part only, since the CRC algorithm determines the CRC length.

Parameters

out	*response*	structure with indication if CRC validation has passed.

Returns

void.

Note

Memory for both request.data & response.data structures should be allocated. Response.data can point to request.data if sufficient space is available at end of request.data structure for the appended CRC.
CRC polynomial is "D6 + D5 + 1", refer to 3GPP TS 38.212.

7.11.2.14 bblib_lte_crc6_gen()

```
void bblib_lte_crc6_gen (
            struct bblib_crc_request * request,
            struct bblib_crc_response * response )
```

Performs CRC6 generate, calculating the CRC value and appending to the data.

Parameters

in	*request*	structure containing pointer to input data and data length.
out	*response*	structure containing calculated CRC value, CRC appended data and new length.

Returns

void

Note

Memory for both request.data & response.data structures should be allocated. Response.data can point to request.data if sufficient space is available at end of request.data structure for the appended CRC.
CRC polynomial is "D6 + D5 + 1", refer to 3GPP TS 38.212.

7.11.2.15 bblib_lte_crc_version()

```
int16_t bblib_lte_crc_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_lte_crc library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.12 phy_dct_idct.h File Reference

Functions

- void dct_avx512 (int16_t ∗pDataIn, int16_t ∗pDataOut, const int16_t nSize, uint8_t ∗pScale)
- void idct_avx512 (int16_t ∗pDataIn, int16_t ∗pDataOut, const int16_t nSize, uint8_t ∗pScale)

7.12.1 Detailed Description

DCT and IDCT.

7.12.2 Function Documentation

7.12.2.1 dct_avx512()

```
void dct_avx512 (
            int16_t * pDataIn,
            int16_t * pDataOut,
            const int16_t nSize,
            uint8_t * pScale )
```

Calculate DCT transformation.

Parameters

in	*pDataIn*	Input buffer for dft
in	*pDataOut*	Output buffer for dft
in	*nSize*	dft size
out	*pScale*	Output scaling of dft computing

7.12.2.2 idct_avx512()

```
void idct_avx512 (
            int16_t * pDataIn,
            int16_t * pDataOut,
            const int16_t nSize,
            uint8_t * pScale )
```

Calculate IDCT transformation.

Parameters

in	*pDataIn*	Input buffer for idft
in	*pDataOut*	Output buffer for idft
in	*nSize*	idft size
out	*pScale*	Output scaling of idft computing

7.13 phy_deinterleave.h File Reference

Data Structures

- struct bblib_deinterleave_request
- struct bblib_deinterleave_response

Enumerations

- enum modulation_type { MOD_QPSK = 2, MOD_16QAM = 4, MOD_64QAM = 6 }
- enum cp_type { CP_Normal = 0, CP_Extend = 1 }
- enum ack_type { Multplexing = 0, Bundling = 1 }

Functions

- int16_t bblib_deinterleave_version (char ∗version, int buffer_size)

- int16_t bblib_deinterleave (const struct bblib_deinterleave_request ∗request, struct bblib_deinterleave_response ∗response)
- int16_t **bblib_deinterleave_c** (const struct bblib_deinterleave_request ∗request, struct bblib_deinterleave_response ∗response)
- int16_t **bblib_deinterleave_avx2** (const struct bblib_deinterleave_request ∗request, struct bblib_deinterleave_response ∗response)
- int16_t **bblib_deinterleave_avx512** (const struct bblib_deinterleave_request ∗request, struct bblib_deinterleave_response ∗response)

- int16_t bblib_deinterleave_data_only (const struct bblib_deinterleave_request ∗request, struct bblib_deinterleave_response ∗response)
- int16_t **bblib_deinterleave_data_only_c** (const struct bblib_deinterleave_request ∗request, struct bblib_deinterleave_response ∗response)
- int16_t **bblib_deinterleave_data_only_avx2** (const struct bblib_deinterleave_request ∗request, struct bblib_deinterleave_response ∗response)
- int16_t **bblib_deinterleave_data_only_avx512** (const struct bblib_deinterleave_request ∗request, struct bblib_deinterleave_response ∗response)

7.13.1 Detailed Description

External API for LTE deinterleave for the QPSK/16QAM/64QAM.

Overview:

The lib_deinterleave kernel is a 4G de-interleave for QPSK/16QAM/64QAM. And it pick up the data of UCI in the input data. The algorithm is implemented as defined in TS36.212 section 5.2.2.6 and 5.2.2.8 (v12.0.0).
Requirements and Test Coverage:

The format of input data as follow:
1>Each symbol is organized as two 16-bit fixed-pointer integers, which represent the real part and the imaginary part respectively.
2>The symbols of different subcarriers are interleaved with four symbols a group.
Functional tests output symbols have been verified bit exact against a Matlab reference model.

7.13.2 Enumeration Type Documentation

7.13.2.1 ack_type

```
enum ack_type
```

ACK type for TDLTE.

Enumerator

Multplexing	Multplexing type
Bundling	Bundling type

7.13.2.2 cp_type

```
enum cp_type
```

Cyclic prefix type for LTE.

Enumerator

CP_Normal	Normal CP
CP_Extend	Extend CP

7.13.2.3 modulation_type

```
enum modulation_type
```

modulation type for LTE.

Enumerator

MOD_QPSK	QPSK modulatino type
MOD_16QAM	16QAM modulatino type
MOD_64QAM	64QAM modulatino type

7.13.3 Function Documentation

7.13.3.1 bblib_deinterleave()

```
int16_t bblib_deinterleave (
            const struct bblib_deinterleave_request * request,
            struct bblib_deinterleave_response * response )
```

Implements deinterleave with QPSK/16QAM/64QAM including data, HARQ-ACK and Rank Information.

Parameters

in	*request*	Structure containing the configuration, input data, lengths for different data types.
out	*response*	Structure containing the output data.

Note

Refers to 3GPP TS 36.212 section 5.2.2.8.

Returns

0 on success, -1 otherwise.

7.13.3.2 bblib_deinterleave_data_only()

```
int16_t bblib_deinterleave_data_only (
            const struct bblib_deinterleave_request * request,
            struct bblib_deinterleave_response * response )
```

Implements deinterleave with QPSK/16QAM/64QAM for data channel only. No HARQ-ACK or Rank Information processed in this API allowing for better performance than bblib_deinterleave.

Parameters

in	*request*	Structure containing the configuration, input data, lengths for different data types.
out	*response*	Structure containing the output data.

Note

Refers to 3GPP TS 36.212 section 5.2.2.8.

Returns

0 on success, -1 otherwise.

7.13.3.3 bblib_deinterleave_version()

```
int16_t bblib_deinterleave_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_deinterleave library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.14 phy_demodulation.h File Reference

Data Structures

- struct bblib_demodulation_request
- struct bblib_demodulation_response

Enumerations

- enum bblib_demod_llr_polarity { **BBLIB_LLR_POLARITY_POSITIVE** = 0, **BBLIB_LLR_POLARITY_NE↩ GATIVE** = 1 }

Functions

- int16_t bblib_lte_demodulation_version (char *version, int buffer_size)
- int32_t bblib_lte_demodulation (const struct bblib_demodulation_request *request, struct bblib_demodulation_response *response)
- int32_t **bblib_lte_demodulation_avx2** (const struct bblib_demodulation_request *request, struct bblib_demodulation_response *response)
- int32_t **bblib_lte_demodulation_avx512** (const struct bblib_demodulation_request *request, struct bblib_demodulation_response *response)
- int32_t bblib_lte_demodulation_polarity (const struct bblib_demodulation_request *request, struct bblib_demodulation_response *response)
- int32_t **bblib_lte_demodulation_polarity_avx2** (const struct bblib_demodulation_request *request, struct bblib_demodulation_response *response)
- int32_t **bblib_lte_demodulation_polarity_avx512** (const struct bblib_demodulation_request *request, struct bblib_demodulation_response *response)

7.14.1 Detailed Description

LTE demodulation for the QPSK/16QAM/64QAM/256QAM.

The demodulation kernel is a LTE demodulator implemented using The Nearest Neighbor algorithm. This method calculates distances between input sample and given constelation points. The demodulation function takes 3 64-bytes aligned buffers and stores the output to the single 64-bytes aligned buffer. The result is saturated in range [-threshold, threshold - 1]. The input is assumed to be a fixed point interleaved IQ samples in the following order:

Symbol0Subcarrier0, Symbol1Subcarrier0, Symbol2Subcarrier0, Symbol3Subcarrier0, Symbol0Subcarrier1, Symbol1Subcarrier1, ...

It can be viewd as the input is "sorted" by a subcarrier. Each input buffer contains 4 symbols and a given number of subcarriers in each symbol. Each SymbolXSubcarrierY element consists of 2 16-bits samples where the first one is a I sample and the second one is a Q sample. Hence the total length of the input buffer is: 4 [symbols] * number_of_carriers * 2 [samples per complex number] = 8 * number_of_carriers.

The output data has a following order:

Symbol0Subcarrier0, Symbol0Subcarrier1, Symbol0Subcarrier2, Symbol0Subcarrier3, ..., Symbol1Subcarrier0, Symbol1Subcarrier1, Symbol1Subcarrier2, Symbol1Subcarrier3, ...

I this case the output is "sorted" by a symbol. Each SymbolXSubcarrierY consists of a number of LLRs; that number depends on the modulation order. Each LLR consist of two 8 bit numbers (stored in the one 16-bits integer) representing the result of demodulating the I and the Q sample. Each 8-bits value is within the range limited by the threshold. Inputs are processed sequentially and the output is built by concatenating the result of processing each input: output = demodulate(input1) CONCAT demodulate(input2) CONCAT demodulate(input3). The length of the output buffer is 3 [inputs] * (mod_order / 2) * 4 [symbols] * number_of_carriers = 12 * (mod_order / 2) * number_of_carriers. The output buffer is an array of 16-bits number where the upper 8 bits represent the result of demodulating the I sample and lower 8 bits of demodulation the Q sample.

7.14.2 Function Documentation

7.14.2.1 bblib_lte_demodulation()

```
int32_t bblib_lte_demodulation (
            const struct bblib_demodulation_request * request,
            struct bblib_demodulation_response * response )
```

Demodulation procedure.

Parameters

in	*request*	Structure containing the input symbols, modulation order, shift and threshold to compensate for the IDFT, and constants to scale based on noise and channel power.
out	*response*	Structure containing the soft-bits and the number of outputs.

Returns

Demodulation result, return 0 is success,return -1 is fail.

Note

Input and output buffers have to be 64 bytes aligned.
Only subset of the request parameters is used by each order of demodulation.
bblib_demod_llr_polarity is ignored in this API implementation

7.14.2.2 bblib_lte_demodulation_polarity()

```
int32_t bblib_lte_demodulation_polarity (
            const struct bblib_demodulation_request * request,
            struct bblib_demodulation_response * response )
```

Demodulation procedure.

Parameters

in	*request*	Structure containing the input symbols, modulation order, shift and threshold to compensate for the IDFT, and constants to scale based on noise and channel power.
out	*response*	Structure containing the soft-bits and the number of outputs.

Returns

Demodulation result, return 0 is success,return -1 is fail.

Note

Input and output buffers have to be 64 bytes aligned.
Only subset of the request parameters is used by each order of demodulation.
this API enables the bblib_demod_llr_polarity in the bblib_demodulation_request allowing to change the sign of the LLRs

7.14.2.3 bblib_lte_demodulation_version()

```
int16_t bblib_lte_demodulation_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_lte_demodulation library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, has to be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.15 phy_demodulation_pucch.h File Reference

Data Structures

- struct bblib_demod_pucch_request
- struct bblib_demod_pucch_response

Enumerations

- enum { BBLIB_NUM_CQI_SYMB_PER_SLOT = 5, BBLIB_NUM_CQI_SYMB_PER_SUBF = 10 }
- enum bblib_demod_pucch_format {
  BBLIB_PUCCH_FORMAT1 = 0, BBLIB_PUCCH_FORMAT1A = 1, BBLIB_PUCCH_FORMAT1B = 2, BBLIB_PUCCH_FORMAT2 = 3,
  BBLIB_PUCCH_FORMAT2A = 4, BBLIB_PUCCH_FORMAT2B = 5 }

Functions

- int16_t bblib_demod_pucch_version (char ∗version, int buffer_size)

- int bblib_demod_pucch (const struct bblib_demod_pucch_request *request, struct bblib_demod_pucch_response *response)
- int **bblib_demod_pucch_c** (const struct bblib_demod_pucch_request *request, struct bblib_demod_pucch_response *response)
- int **bblib_demod_pucch_fxp** (const struct bblib_demod_pucch_request *request, struct bblib_demod_pucch_response *response)
- int **bblib_demod_pucch_fxp_c** (const struct bblib_demod_pucch_request *request, struct bblib_demod_pucch_response *response)
- int **bblib_demod_pucch_fxp_avx2** (const struct bblib_demod_pucch_request *request, struct bblib_demod_pucch_response *response)
- int **bblib_demod_pucch_fxp_avx512** (const struct bblib_demod_pucch_request *request, struct bblib_demod_pucch_response *response)

7.15.1 Detailed Description

API for PUCCH Demodulation Library.

Overview:

The purpose of this kernel is to implement Demodulation for PUCCH formats 1, 1a, 1b, 2, 2a and 2b in fixed point. Based on the 36.211 version 14 3GPP specification, chapter 5.4.1.

Requirements and Test Coverage:

Functional Test Cases:

- 32 bit floating point implementation of demodulation for PUCCH formats 1, 1a, 1b, 2, 2a and 2b.
- 16 bit fixed point implementation of demodulation for PUCCH formats 1, 1a, 1b, 2, 2a and 2b.

Performance Tests:

- Cycle count measurements for PUCCH format 1, 1a, 1b, 2, 2a and 2b use cases.

Algorithm Guidance:

The kernel implements demodulation for the specified PUCCH format requested and stores the result to the output buffer.

For PUCCH format 1s, the kernel implements Ack/Nack demodulation. This has two parts.

1. Perform maximum ratio combining and channel compensation using the input ack/nack correlation for each slot and Rx antenna and input averaged channel estimation.
2. Detection algorithm to determine if format 1s were detected or if a deadzone was detected. This is determined by redoing the channel compensation and comparing the result with the previous to determine if they match or are within an ideal range.

For PUCCH format 2s, the kernel implements CQI demodulation. This has three parts.

1. For format 2a and 2b, use the decoded ack/nack bits to determine the channel estimate. The decoded ack/nack bits are determined using the same demodulation method used for format 1s.
2. Perform maximum ratio combining and channel compenstation using only the CQI bits.
3. Apply Rx Demapper for the CQI bits, with the soft decisions generated being output for reed muller decoding.

7.15.2 Enumeration Type Documentation

7.15.2.1 anonymous enum

```
anonymous enum
```

Some standard static values for PUCCH demodulation.

Enumerator

BBLIB_NUM_CQI_SYMB_PER_SLOT	Number of CQI data symbols per slot for PUCCH formats 2, 2a, 2b.
BBLIB_NUM_CQI_SYMB_PER_SUBF	Number of CQI data symbols per subframe for PUCCH formats 2, 2a, 2b.

7.15.2.2 bblib_demod_pucch_format

```
enum bblib_demod_pucch_format
```

Enum describing the PUCCH format used for demodulation.

Enumerator

BBLIB_PUCCH_FORMAT1	Perform demodulation for PUCCH format 1.
BBLIB_PUCCH_FORMAT1A	Perform demodulation for PUCCH format 1a.
BBLIB_PUCCH_FORMAT1B	Perform demodulation for PUCCH format 1b.
BBLIB_PUCCH_FORMAT2	Perform demodulation for PUCCH format 2.
BBLIB_PUCCH_FORMAT2A	Perform demodulation for PUCCH format 2a.
BBLIB_PUCCH_FORMAT2B	Perform demodulation for PUCCH format 2b.

7.15.3 Function Documentation

7.15.3.1 bblib_demod_pucch()

```
int bblib_demod_pucch (
            const struct bblib_demod_pucch_request * request,
            struct bblib_demod_pucch_response * response )
```

PUCCH Demodulation calculation for format 1 and 2 in floating point and and fixed point Q16s11 format. Function supports floating point in C and fixed point in C, AVX2 and AVX512.

Parameters

in	*request*	Structure containing the input buffers and settings.
out	*response*	Structure containing the output buffer.

Returns

0 if successful, negative on error.

7.15.3.2 bblib_demod_pucch_version()

```
int16_t bblib_demod_pucch_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_demodulation_pucch library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, length BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.16 phy_dft_idft.h File Reference

Data Structures

- struct bblib_dft_request
- struct bblib_dft_response
- struct bblib_dft_burst_request
- struct bblib_dft_burst_response
- struct bblib_idft_burst_request
- struct bblib_idft_burst_response

Enumerations

- enum dft_idft_flag_type { **BBLIB_DFT_TYPE**, **BBLIB_IDFT_TYPE** }

Functions

- int16_t bblib_lte_dft_idft_version (char *version, int buffer_size)

- void bblib_dft_idft_fxp (const struct bblib_dft_request *request, struct bblib_dft_response *response)
- void **bblib_dft_idft_fxp_avx2** (const struct bblib_dft_request *request, struct bblib_dft_response *response)
- void **bblib_dft_idft_fxp_avx512** (const struct bblib_dft_request *request, struct bblib_dft_response *response)

- void bblib_dft_idft_fxp_scale_avx512 (const struct bblib_dft_request *request, struct bblib_dft_response *response)
- void **bblib_dft_idft_fxp_scale_srs_avx512** (const struct bblib_dft_request *request, struct bblib_dft_response *response)
- void **bblib_dft_fxp_scale_avx512** (const struct bblib_dft_request *request, struct bblib_dft_response *response)
- void **bblib_idft_fxp_scale_avx512** (const struct bblib_dft_request *request, struct bblib_dft_response *response)
- void **bblib_idft_fxp_scale_prach_avx512** (const struct bblib_dft_request *request, struct bblib_dft_response *response)
- void **bblib_idft_fxp_scale_srs_avx512** (const struct bblib_dft_request *request, struct bblib_dft_response *response)
- void **bblib_dft_fxp_scale_srs_avx512** (const struct bblib_dft_request *request, struct bblib_dft_response *response)
- int16_t bblib_dft_idft_flp (const struct bblib_dft_request *request, struct bblib_dft_response *response)
- void bblib_dft_burst_fxp (const struct bblib_dft_request *request, struct bblib_dft_response *response)
- void **bblib_dft_burst_fxp_avx2** (const struct bblib_dft_burst_request *request, struct bblib_dft_burst_response *response)
- void **bblib_dft_burst_fxp_avx512** (const struct bblib_dft_request *request, struct bblib_dft_response *response)

- void bblib_idft_burst_fxp (const struct bblib_dft_request *request, struct bblib_dft_response *response)
- void **bblib_idft_burst_fxp_avx2** (const struct bblib_idft_burst_request *request, struct bblib_idft_burst_response *response)
- void **bblib_idft_burst_fxp_avx512** (const struct bblib_dft_request *request, struct bblib_dft_response *response)

7.16.1 Detailed Description

Header file for module with implementation of DFT/IDFT.

Overview: This module performs DFT/IDFT using mixed radix.

Algorithm Guidance: Basically, a one domain sequence N is transformed into an other domain sequence N. Considering N is composite number in LTE system, so mixed radix DFT/IDFT is introduced to reduce computing efforts. Taking 24 point DFT as an example, 6 multiply 4 equals to 24. So, processing 24-point-DFT need process radix6-DFT 4 times at first, and then process radix4-DFT 6 times.

To prevent overflow from occuring and get better performance scaling of DFT/IDF is done according to nFactor (result is multiplied by 1/nFactor). Different DFT/IDFT sizes have different value of the nFactor: 1) nFactor = 2 dft/idft size <= 48 or dft/idft size = 72 2) nFactor = 4 dft/idft size >= 60 and dft/idft size <= 120 3) nFactor = 8 other dft/idft sizes Furthermore IDFT result is scaled by dft/idft size (multiplied by dft/idft size).

7.16.2 Function Documentation

7.16.2.1 bblib_dft_burst_fxp()

```
void bblib_dft_burst_fxp (
            const struct bblib_dft_request * request,
            struct bblib_dft_response * response )
```

This function implements FXP dft for burst inputs. Single symbol FXP DFT is obtained by using the function with single input buffer (num_input_buffers=1).

Parameters

in	*request*	Structure containing the input data, dft point, and direction/type of the dft.
out	*response*	Structure containing the output data.

Note

data_in and data_out need to be aligned with 128 bits.

7.16.2.2 bblib_dft_idft_flp()

```
int16_t bblib_dft_idft_flp (
            const struct bblib_dft_request * request,
            struct bblib_dft_response * response )
```

This function implements FLP dft & idft. This function calls MKL's DftiComputeForward or DftiComputeBackward.

Parameters

in	*request*	Structure containing the input data, dft point and direction/type of the dft.
out	*response*	Structure containing the output data.

Note

data_in and data_out need to be aligned with 128 bits.

Returns

Return 0 for success, and -1 for error.

7.16.2.3 bblib_dft_idft_fxp()

```
void bblib_dft_idft_fxp (
            const struct bblib_dft_request * request,
            struct bblib_dft_response * response )
```

This function implements FXP dft & idft.

Parameters

in	*request*	Structure containing the input data, dft point and direction/type of the dft.
out	*response*	Structure containing the output data.

Note

data_in and data_out need to be aligned with 128 bits.

7.16.2.4 bblib_dft_idft_fxp_scale_avx512()

```
void bblib_dft_idft_fxp_scale_avx512 (
            const struct bblib_dft_request * request,
            struct bblib_dft_response * response )
```

This function implements AVX512 FXP dft & idft with scaling out factor.

Parameters

in	*request*	Structure containing the input data, dft point and direction/type of the dft.
out	*response*	Structure containing the output data.

Note

data_in and data_out need to be aligned with 512 bits.

7.16.2.5 bblib_idft_burst_fxp()

```
void bblib_idft_burst_fxp (
            const struct bblib_dft_request * request,
            struct bblib_dft_response * response )
```

This function implements FXP idft for burst inputs. Single symbol FXP IDFT is obtained by using the function with single input buffer (num_input_buffers=1).

Parameters

in	*request*	Structure containing the input data, idft point, and direction/type of the idft.
out	*response*	Structure containing the output data.

Note

data_in and data_out need to be aligned with 128 bits.

7.16.2.6 bblib_lte_dft_idft_version()

```
int16_t bblib_lte_dft_idft_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_lte_dft_idft library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.17 phy_dftcodebook_weightgen.h File Reference

Data Structures

- struct bblib_dftcodebook_weightgen_request
- struct bblib_dftcodebook_weightgen_response

Enumerations

- enum dftcodebook_weightgen_constants { BBLIB_DFTCODEBOOK_MAX_RX_ANT_NUM = 64, BBLIB_DFTCODEBOOK_MA = 16 }

Functions

- int16_t bblib_dftcodebook_weightgen_version (char *version, int buffer_size)
- int32_t bblib_dftcodebook_weightgen (const bblib_dftcodebook_weightgen_request *request, bblib_dftcodebook_weightgen_re *response)
- int32_t **bblib_dftcodebook_weightgen_avx512** (const bblib_dftcodebook_weightgen_request *request, bblib_dftcodebook_weightgen_response *response)

7.17.1 Detailed Description

External API for DFT-based beamforming codebook matrix generation in 5GNR.

Overview: This module implements DFT-based beamforming codebook matrix generation in 5GNR. It can support l∗32 and l∗64 configuration, l>=0.

Algorithm Guidance: DFT-based Codebook UL beamforming Matrix Gen Step 1. Calculate DFT-based codebook matrix; Step 2. Calculate Matrix Multiply of Gain = Codebook ∗ SRSCEout H; Step 3. Sort the Max nStream of the Gain, and get the corresponding nStream Indexes; Step 4. Select the corresponding nStream rows of the DFT-based codebook matrix.

7.17.2 Enumeration Type Documentation

7.17.2.1 dftcodebook_weightgen_constants

```
enum dftcodebook_weightgen_constants
```

Constants used in dft codebook based weight matrix generation.

Enumerator

BBLIB_DFTCODEBOOK_MAX_RX_ANT_NUM	MAX number of Rx antennas
BBLIB_DFTCODEBOOK_MAX_STREAM_NUM	MAX number of Output Streams

7.17.3 Function Documentation

7.17.3.1 bblib_dftcodebook_weightgen()

```
int32_t bblib_dftcodebook_weightgen (
            const bblib_dftcodebook_weightgen_request * request,
            bblib_dftcodebook_weightgen_response * response )
```

dftcodebook matrix generation.

Parameters

in	*request*	Input request structure .
out	*response*	Output response structure.

Returns

0 for success, and -1 for error

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.17.3.2 bblib_dftcodebook_weightgen_version()

```
int16_t bblib_dftcodebook_weightgen_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_dftcodebook_matrix_gen library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.18 phy_eigen_beamforming.h File Reference

Data Structures

- struct bblib_eigen_beamforming_request
- struct bblib_eigen_beamforming_response

Enumerations

- enum bblib_eigen_beamforming_max_dimension {
  k_maxNumAntennas = 32, k_maxNumUsers = 16, k_maxLayers = 8, k_maxChannelEstimates = k_max←
  NumAntennas ∗ k_maxNumUsers,
  k_maxMatrices = 16 }

Functions

- int16_t bblib_eigen_beamforming_version (char ∗version, int buffer_size)
- void bblib_eigen_beamforming (const struct bblib_eigen_beamforming_request ∗request, struct bblib_eigen_beamforming_resp ∗response)
- void bblib_eigen_beamforming_avx2 (const struct bblib_eigen_beamforming_request ∗request, struct bblib_eigen_beamforming_response ∗response)
- void **bblib_eigen_beamforming_avx512** (const struct bblib_eigen_beamforming_request ∗request, struct bblib_eigen_beamforming_response ∗response)
- void **bblib_eigen_beamforming_c** (const struct bblib_eigen_beamforming_request ∗request, struct bblib_eigen_beamforming_response ∗response)

7.18.1 Detailed Description

External API for 5G NR eigen beamforming.

Overview: This module performs Beamforming using eigen decomposition.

Algorithm Guidance:

1. The outer product of the input channel estimates is computed.
2. Eigen decomposition generates a set of eigen vectors and their corresponding eigen values.
3. The eigen values are sorted in place. An additional list giving their original indexes is generated to allow the eigen vectors to be reordered later.
4. Waterfilling is used to decide what power to allocate to each eigen vector.
5. The precoding matrix is generated by moving the eigen vectors into the order given by the sort indexes, and then scaling the entire vector by the power allocation.

7.18.2 Enumeration Type Documentation

7.18.2.1 bblib_eigen_beamforming_max_dimension

```
enum bblib_eigen_beamforming_max_dimension
```

Enum describing the maximum dimensions which can be used in eigen beamforming.

Enumerator

k_maxNumAntennas	maximum number of antennas
k_maxNumUsers	maximum number of users
k_maxLayers	maximum number of layers
k_maxChannelEstimates	maximum number of input channel estimastes
k_maxMatrices	maximum number of matrices to work on

7.18.3 Function Documentation

7.18.3.1 bblib_eigen_beamforming()

```
void bblib_eigen_beamforming (
            const struct bblib_eigen_beamforming_request * request,
            struct bblib_eigen_beamforming_response * response )
```

This is the default DL Eigen Beamforming function. This will select which beamforming function to run based on the ISA selected at compile time, the highest available ISA is used.

Parameters

in	*request*	structure
out	*response*	structure

7.18.3.2 bblib_eigen_beamforming_avx2()

```
void bblib_eigen_beamforming_avx2 (
            const struct bblib_eigen_beamforming_request * request,
            struct bblib_eigen_beamforming_response * response )
```

DL Eigen Beamforming procedures in AVX512, AVX2 and C++.

Parameters

in	*request*	structure
out	*response*	structure

7.18.3.3 bblib_eigen_beamforming_version()

```
int16_t bblib_eigen_beamforming_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_eigen_beamforming library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, typically no more than BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.19 phy_fd_correlation.h File Reference

Data Structures

- struct bblib_fd_correlation_request
- struct bblib_fd_correlation_response

Functions

- int16_t bblib_fd_correlation_version (char ∗version, int buffer_size)
- int32_t bblib_fd_correlation (const struct bblib_fd_correlation_request ∗request, struct bblib_fd_correlation_response ∗response)
- void **bblib_fd_correlation_c** (const struct bblib_fd_correlation_request ∗request, struct bblib_fd_correlation_response ∗response)
- void **bblib_fd_correlation_avx2** (const struct bblib_fd_correlation_request ∗request, struct bblib_fd_correlation_response ∗response)
- void **bblib_fd_correlation_avx512** (const struct bblib_fd_correlation_request ∗request, struct bblib_fd_correlation_response ∗response)

7.19.1 Detailed Description

frequency domain correlation.

The kernel implements frequency domain correlation for 2 sequences. The algorithm description is out = conjugated(in0) ∗ in1. The module is used for fixed point calculation. The fixed point scaled of output is: temp[31:0] = ((conjugated(in0[15:0]) ∗ in1[15:0]) >> 14) + 1 out[15:0] = temp[16:1]

7.19.2 Function Documentation

7.19.2.1 bblib_fd_correlation()

```
int32_t bblib_fd_correlation (
            const struct bblib_fd_correlation_request * request,
            struct bblib_fd_correlation_response * response )
```

fd_correlation procedure.

Parameters

in	*request*	Structure containing the input data, and the length of the sequences.
out	*response*	Structure containing the output data.

Returns

fd correlation result, return 0 is success,return -1 is fail.

Note

Input and output buffers have to be 64 bytes aligned.

7.19.2.2 bblib_fd_correlation_version()

```
int16_t bblib_fd_correlation_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_fd_correlation library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, has to be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.20 phy_fec_enc_byte_concat_soft.h File Reference

Data Structures

- struct bblib_fec_enc_byte_concat_soft_request
- struct bblib_fec_enc_byte_concat_soft_response

Functions

- int16_t bblib_fec_enc_byte_concat_soft_version (char *version, int buffer_size)

- void bblib_fec_enc_byte_concat_soft (const struct bblib_fec_enc_byte_concat_soft_request *request, struct bblib_fec_enc_byte_concat_soft_response *response)
- void **bblib_fec_enc_byte_concat_soft_sse** (const struct bblib_fec_enc_byte_concat_soft_request *request, struct bblib_fec_enc_byte_concat_soft_response *response)

7.20.1 Detailed Description

Concadenates input blocks of encoded bytes to a single output block.

Overview:

Concadenates input blocks of encoded bytes to a single output block. hLen is the numbers of input blocks. E is the pointer to the length of each code block [bits]. tBitTotal is the total length of output block [bits].

7.20.2 Function Documentation

7.20.2.1 bblib_fec_enc_byte_concat_soft()

```
void bblib_fec_enc_byte_concat_soft (
            const struct bblib_fec_enc_byte_concat_soft_request * request,
            struct bblib_fec_enc_byte_concat_soft_response * response )
```

fec_enc_byte_concat_soft

Parameters

in	*request*	Structure containing configuration information and input data.
out	*response*	Structure containing kernel outputs.

Note

bblib_fec_enc_byte_concat_soft provides the most appropriate version for the available ISA, the _avx512 etc. version allow direct access to specific ISA implementations.

7.20.2.2 bblib_fec_enc_byte_concat_soft_version()

```
int16_t bblib_fec_enc_byte_concat_soft_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_fec_enc_byte_concat_soft library.

Parameters

in	*version*	Pointer to the char buffer where the version string is be copied.
in	*buffer_size*	The length of the string buffer. Must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

Returns 0 if the version string was populated, -1 otherwise.

7.21 phy_fft_ifft.h File Reference

Data Structures

- struct bblib_fft_request
- struct bblib_fft_response

Functions

- int16_t bblib_fft_ifft_version (char *version, int buffer_size)

- void bblib_ifft (const struct bblib_fft_request *request, struct bblib_fft_response *response)
- void **bblib_ifft_avx2** (const struct bblib_fft_request *request, struct bblib_fft_response *response)
- void **bblib_ifft_avx512** (const struct bblib_fft_request *request, struct bblib_fft_response *response)
- void **bblib_ifft_c** (const struct bblib_fft_request *request, struct bblib_fft_response *response)

- void bblib_fft (const struct bblib_fft_request *request, struct bblib_fft_response *response)
- void **bblib_fft_avx2** (const struct bblib_fft_request *request, struct bblib_fft_response *response)
- void **bblib_fft_avx512** (const struct bblib_fft_request *request, struct bblib_fft_response *response)
- void **bblib_fft_c** (const struct bblib_fft_request *request, struct bblib_fft_response *response)

7.21.1 Detailed Description

Calculate 1024 points FFT and IFFT of the given input using the transpose algorithm.

Overview

IDFT defined as $x_n = \frac{1}{\sqrt{N}} * \sum_{k=0}^{N-1} X_n e^{\frac{2\pi j n}{N} k}$ where $N = 1024$ is the size of the IDFT and X_n and x_n are DFT input and output as defined below. As it can be seen IDFT is scaled to preserve the input power. The reordering of the data is done inside IFFT.

Data format

The input are interleaved IQ samples in the Q16s15 format in the natural order. The output are interleaved IQ samples in the Q16s15 format.

Input Signal Amplitude and Fixed Point Design

- FFT with 512/2048 points

The recommended average amplitude of the input signal is 2^9.

There is a 3-bit right shift embedded. For 512 points FFT, do the 1-bit shift for twiddle-factor-512 multiplex, twiddle-factor-16 muliplication, twiddle-factor-32 multiplex, respectively. For 2048 points FFT, do the 1-bit shift for twiddle-factor-2048 multiplex, twiddle-factor-64 multiplex, twiddle-factor-32 multiplex, respectively.

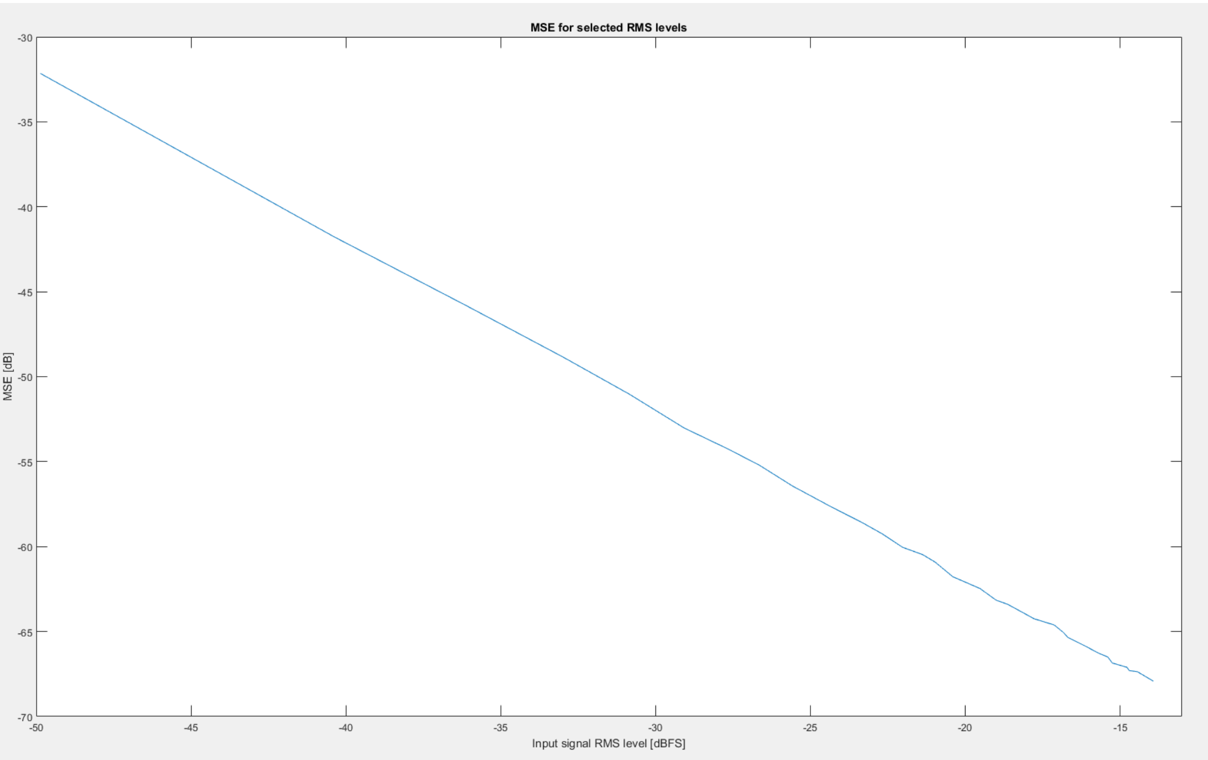
Therefore, the factor of output/input is sqrt(N)/2^3.

Under the recommended signal amplitude input, the output signal average amplitude will be sqrt(N)∗2^6.

- FFT with 1024 points

Input signal RMS magnitude has to be below -14dbFS for Gaussian distributed signals. 0 dBFS is defined as the maximum signed number that can be represented on 16 bits - that is 32767.

The MSE is as on the figure below:



Algorithm guidance

- Cooley-Turkey Alogorithm for FFT with 512/2048 points.

Cooley-Turkey FFT algorithm is a divide and conquer algorithm that recursively breaks down a DFT of any composite size N = N1∗N2 into many smaller DFT of sizes N1 and N2.

The 3 steps below are as follows:

1. Perform inner small factor N1 length DFT.

2. Perform twiddle factor multiplex.
3. Perform outer small factor N2 length DFT.

Divide 512 points FFT/IFFT calculation as (4∗4) ∗ (4∗8); Twiddle factors of 16 points, 32 points and 512 points are needed.

Divide 2048 points FFT/IFFT calculation as (4∗8) ∗ (8∗8); Twiddle factors of 32 points, 64 points and 512 points are needed.

- "Four Step" Alogorithm for FFT with 1024 points

The algorithm is based on the "Four Step" FFT algorithm described by Cooley and Agarwal, and Swarztrauber, and Gentleman and Sande. The description of it can be found in:

D. H. Bailey, "FFTs in external or hierarchical memory," Proceedings of the 1989 ACM/IEEE Conference on Supercomputing (Supercomputing '89), Reno, NV, USA, 1989, pp. 234-242.

The implementation below merges step 2 and 3 into one yielding 3 steps. It assumes the size of the input to have a natural square root and is a power of 2 .

The 3 steps below are as follows:

1. Perform sqrt(N)-points IFFT on every column of the matrix. This step can be vectorized to perform all IFFTs simultaneously.
2. Transpose the result and multiply it by roots of unity matrix.
3. The same as step 1, but on the transposed matrix.

Another useful resource for this algorithm is `http://parallelcomp.uw.hu/ch13lev1sec3.html` as it illustrates how the algorithm works on pictures (however it lacks the details needed for the implementation).

7.21.2 Function Documentation

7.21.2.1 bblib_fft()

```
void bblib_fft (
            const struct bblib_fft_request * request,
            struct bblib_fft_response * response )
```

Function performs FFT of the given size as described in the module description.

Parameters

in	*request*	Request structure with input buffer and size of the FFT.
out	*response*	Response structure with output buffer and size of the FFT.

7.21.2.2 bblib_fft_ifft_version()

```
int16_t bblib_fft_ifft_version (
            char * version,
            int buffer_size )
```

Report the version number for the fft/ifft library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.21.2.3 bblib_ifft()

```
void bblib_ifft (
            const struct bblib_fft_request * request,
            struct bblib_fft_response * response )
```

Function performs IFFT of the given size as described in the module description.

Parameters

in	*request*	Request structure with input buffer and size of the IFFT.
out	*response*	Response structure with output buffer and size of the IFFT.

7.22 phy_irc_rnn_calculation_5gnr.h File Reference

Data Structures

- struct bblib_irc_rnn_calculation_5gnr_request
- struct bblib_irc_rnn_calculation_5gnr_response

Functions

- void bblib_print_irc_rnn_calculation_5gnr_version ()
- int16_t bblib_irc_rnn_calculation_5gnr_version (char *version, int buffer_size)

- void bblib_irc_rnn_calculation_5gnr (struct bblib_irc_rnn_calculation_5gnr_request ∗request, struct bblib_irc_rnn_calculation_5gnr_response ∗response)
- void **bblib_irc_rnn_calculation_5gnr_avx512** (struct bblib_irc_rnn_calculation_5gnr_request ∗request, struct bblib_irc_rnn_calculation_5gnr_response ∗response)

7.22.1 Detailed Description

External API for Rnn calculation for IRC for 5GNR.

Overview:

The lib_irc_rnn_calculation_5gnr kernel is a 5G NR Rnn calculation for IRC module. This is a generic kernel which can be used in channel estimate in both UL and DL.

Algorithm Guidance:

The 5G NR IRC Rnn Calculation is in this step:

1. IpN = Y - HX, which is performed in CE module.
2. Rnn_dmrs = IpN ∗ IpN'
3. Rnn = mean of 4 PRBs' Rnn_dmrs, and map to all of the subcarriers.

7.22.2 Function Documentation

7.22.2.1 bblib_irc_rnn_calculation_5gnr()

```
void bblib_irc_rnn_calculation_5gnr (
            struct bblib_irc_rnn_calculation_5gnr_request * request,
            struct bblib_irc_rnn_calculation_5gnr_response * response )
```

irc_rnn_calculation_5gnr procedures.

Parameters

in	*request*	Structure containing the input data which need to be 64 bytes alignment.
out	*response*	Structure containing the decoding output data which need 64 byte alignment.

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.22.2.2 bblib_irc_rnn_calculation_5gnr_version()

```
int16_t bblib_irc_rnn_calculation_5gnr_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_irc_rnn_calculation_5gnr library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.22.2.3 bblib_print_irc_rnn_calculation_5gnr_version()

```
void bblib_print_irc_rnn_calculation_5gnr_version ( )
```

printf irc_rnn_calculation 5gnr version

Returns

null.

7.23 phy_layerdemapping_5gnr.h File Reference

Data Structures

- struct bblib_layerdemapping_5gnr_request
- struct bblib_layerdemapping_5gnr_response

Enumerations

- enum bblic_layerdemapper_config { bblib_layerdemapper_max_layers = 4 }

Functions

- int16_t bblib_layerdemapping_5gnr_version (char ∗version, int buffer_size)
- int32_t bblib_layerdemapping_5gnr (const struct bblib_layerdemapping_5gnr_request ∗request, struct bblib_layerdemapping_5gnr_response ∗response)
- int32_t **bblib_layerdemapping_5gnr_avx512** (const struct bblib_layerdemapping_5gnr_request ∗request, struct bblib_layerdemapping_5gnr_response ∗response)

7.23.1 Detailed Description

External API for 5gnr layer demapping.

Overview:

This Kernel implements Layer DeMapping for 5GNR providing support for a single code word with 2, 4 layers. The algorithm is implemented as defined in TS38.211 section 7.3.1.3 (v15.3.0). The input layer and output code word are both fixed point.

Requirements and Test Coverage:

2,4 layers for 1 code word is tested. Input data size is not required to be multiple of AVX512 register width. For best performance input and output buffers should be aligned on a 64byte boundary.

Functional tests perform verification against a reference model.

Algorithm Guidance:

The algorithm is implemented as defined in TS38.211 section 7.3.1.3 (v15.3.0). The module only supports a 2/4 layers from one code word. Layer mapping of more than 1 code word is possible by executing the function once per code word, but this use case is not tested as part of test coverage.

7.23.2 Enumeration Type Documentation

7.23.2.1 bblic_layerdemapper_config

```
enum bblic_layerdemapper_config
```

Configuration supported by the layer demapping APIs including max number of layer.

Enumerator

bblib_layerdemapper_max_layers	Maximum number of layers supported

7.23.3 Function Documentation

7.23.3.1 bblib_layerdemapping_5gnr()

```
int32_t bblib_layerdemapping_5gnr (
            const struct bblib_layerdemapping_5gnr_request * request,
            struct bblib_layerdemapping_5gnr_response * response )
```

layerdemapping_5gnr procedures.

Parameters

in	*request*	Structure containing the data start, data length and input layer data.need 64 byte alignment for pIn.
out	*response*	Structure containing the output code word data. need 64 byte alignment for pOut.

Returns

0 if success, otherwise -1.

7.23.3.2 bblib_layerdemapping_5gnr_version()

```
int16_t bblib_layerdemapping_5gnr_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_layerdemapping_5gnr library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.24 phy_layermapping_5gnr.h File Reference

Data Structures

- struct bblib_layermapping_5gnr_request
- struct bblib_layermapping_5gnr_layers
- struct bblib_layermapping_5gnr_response

Functions

- int16_t bblib_layermapping_5gnr_version (char *version, int buffer_size)

- void bblib_layermapping_5gnr_fxp (const struct bblib_layermapping_5gnr_request *request, struct bblib_layermapping_5gnr_response *response)

- void **bblib_layermapping_5gnr_flp** (const struct bblib_layermapping_5gnr_request *request, struct bblib_layermapping_5gnr_response *response)
- void **bblib_layermapping_5gnr_avx512_fxp** (const struct bblib_layermapping_5gnr_request *request, struct bblib_layermapping_5gnr_response *response)
- void **bblib_layermapping_5gnr_avx512_flp** (const struct bblib_layermapping_5gnr_request *request, struct bblib_layermapping_5gnr_response *response)
- void **bblib_layermapping_5gnr_avx2_fxp** (const struct bblib_layermapping_5gnr_request *request, struct bblib_layermapping_5gnr_response *response)
- void **bblib_layermapping_5gnr_c_fxp** (const struct bblib_layermapping_5gnr_request *request, struct bblib_layermapping_5gnr_response *response)
- void **bblib_layermapping_5gnr_c_flp** (const struct bblib_layermapping_5gnr_request *request, struct bblib_layermapping_5gnr_response *response)

7.24.1 Detailed Description

External API for 5gnr layer mapping.

Overview:

This Kernel implements Layer Mapping for 5GNR providing support for a single code word with 2,3 and 4 layers. The algorithm is implemented as defined in TS38.211 section 7.3.1.3 (v15.0.0). The input code word and output layers are both floating point.

Requirements and Test Coverage:

2,3,4 layers for 1 code word is tested. Input data size is not required to be multiple of AVX512 register width. For best performance input buffers should be aligned on a 64byte boundary.

Functional tests perform verification against a reference model.

Algorithm Guidance:

The algorithm is implemented as defined in TS38.211 section 7.3.1.3 (v15.0.0). The module only supports a 2/3/4 layers from one code word. Layer mapping of more than 1 code word is possible by executing the function once per code word, but this use case is not tested as part of test coverage.

7.24.2 Function Documentation

7.24.2.1 bblib_layermapping_5gnr_fxp()

```
void bblib_layermapping_5gnr_fxp (
            const struct bblib_layermapping_5gnr_request * request,
            struct bblib_layermapping_5gnr_response * response )
```

layermapping_5gnr procedures.

Parameters

in	*request*	Structure containing the input bits, data length and modulation order QPSK/16QAM/64QAM/256QAM.need 64 byte alignment for data_in.
out	*response*	Structure containing the complex output symbols and the number of output symbols.need 64 byte alignment for data_output.

7.24.2.2 bblib_layermapping_5gnr_version()

```
int16_t bblib_layermapping_5gnr_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_layermapping_5gnr library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.25 phy_ldpc_decoder_5gnr.h File Reference

Data Structures

- struct bblib_ldpc_decoder_5gnr_request
- struct bblib_ldpc_decoder_5gnr_response

Functions

- void bblib_print_ldpc_decoder_5gnr_version (void)

- int32_t bblib_ldpc_decoder_5gnr (struct bblib_ldpc_decoder_5gnr_request ∗request, struct bblib_ldpc_decoder_5gnr_response ∗response)
- int32_t **bblib_ldpc_decoder_5gnr_avx2** (struct bblib_ldpc_decoder_5gnr_request ∗request, struct bblib_ldpc_decoder_5gnr_response ∗response)
- int32_t **bblib_ldpc_decoder_5gnr_avx512** (struct bblib_ldpc_decoder_5gnr_request ∗request, struct bblib_ldpc_decoder_5gnr_response ∗response)

7.25.1 Detailed Description

Source code of External API for 5GNR LDPC Encoder functions.

Overview:

The bblib_ldpc_decoder_5gnr kernel is a 5G NR LDPC Encoder function. It is implemented as defined in TS38212 5.3.2

Requirements and Test Coverage:

BaseGraph 1(.2). Lifting Factor $>$176.

Algorithm Guidance:

The algorithm is implemented as defined in TS38212 5.3.2.

7.25.2 Function Documentation

7.25.2.1 bblib_ldpc_decoder_5gnr()

```
int32_t bblib_ldpc_decoder_5gnr (
            struct bblib_ldpc_decoder_5gnr_request * request,
            struct bblib_ldpc_decoder_5gnr_response * response )
```

Encoder for LDPC in 5GNR.

Parameters

in	*request*	Structure containing configuration information and input data.
out	*response*	Structure containing kernel outputs.

Note

bblib_ldpc_decoder_5gnr provides the most appropriate version for the available ISA, the _avx512 etc. version allow direct access to specific ISA implementations.

Returns

Success: return 0, else: return -1.

7.26 phy_ldpc_encoder_5gnr.h File Reference

Data Structures

- struct bblib_ldpc_encoder_5gnr_request
- struct bblib_ldpc_encoder_5gnr_response

Functions

- void bblib_print_ldpc_encoder_5gnr_version (void)

- int32_t bblib_ldpc_encoder_5gnr (struct bblib_ldpc_encoder_5gnr_request *request, struct bblib_ldpc_encoder_5gnr_response *response)
- int32_t **bblib_ldpc_encoder_5gnr_avx512** (struct bblib_ldpc_encoder_5gnr_request *request, struct bblib_ldpc_encoder_5gnr_response *response)

7.26.1 Detailed Description

Source code of External API for 5GNR LDPC Encoder functions.

Overview:

The bblib_ldpc_encoder_5gnr kernel is a 5G NR LDPC Encoder function. It is implemented as defined in TS38212 5.3.2

Requirements and Test Coverage:

BaseGraph 1(.2). Lifting Factor >176.

Algorithm Guidance:

The algorithm is implemented as defined in TS38212 5.3.2.

7.26.2 Function Documentation

7.26.2.1 bblib_ldpc_encoder_5gnr()

```
int32_t bblib_ldpc_encoder_5gnr (
            struct bblib_ldpc_encoder_5gnr_request * request,
            struct bblib_ldpc_encoder_5gnr_response * response )
```

Encoder for LDPC in 5GNR.

Parameters

in	*request*	Structure containing configuration information and input data.
out	*response*	Structure containing kernel outputs.

Note

bblib_ldpc_encoder_5gnr provides the most appropriate version for the available ISA, the _avx512 etc. version allow direct access to specific ISA implementations.

Returns

Success: return 0, else: return -1.

7.27 phy_LDPC_ratematch_5gnr.h File Reference

Data Structures

- struct bblib_LDPC_ratematch_5gnr_request
- struct bblib_LDPC_ratematch_5gnr_response

Functions

- void bblib_print_LDPC_ratematch_5gnr_version (void)

- int32_t bblib_LDPC_ratematch_5gnr (const struct bblib_LDPC_ratematch_5gnr_request *request, struct bblib_LDPC_ratematch_5gnr_response *response)
- int32_t **bblib_LDPC_ratematch_5gnr_avx512** (const struct bblib_LDPC_ratematch_5gnr_request *request, struct bblib_LDPC_ratematch_5gnr_response *response)

7.27.1 Detailed Description

Source code of External API for 5GNR LDPC ratematch functions.

Overview:

The bblib_LDPC_ratematch_5gnr kernel is a 5G NR LDPC ratematch function. It is implemeneted as defined in TS38212 5.4.2

Requirements and Test Coverage:

BaseGraph 1/2; Rvidx 0-3; Modulation Type BPSK/QPSK/16QAM/64QAM/256QAM

Algorithm Guidance:

The algorithm is implemented as defined in TS38212 5.4.2.

7.27.2 Function Documentation

7.27.2.1 bblib_LDPC_ratematch_5gnr()

```
int32_t bblib_LDPC_ratematch_5gnr (
            const struct bblib_LDPC_ratematch_5gnr_request * request,
            struct bblib_LDPC_ratematch_5gnr_response * response )
```

rate matching for LDPC in 5GNR.

Parameters

in	*request*	Structure containing configuration information and input data.
out	*response*	Structure containing kernel outputs.

Note

bblib_LDPC_ratematch_5gnr provides the most appropriate version for the available ISA, the _avx512 etc. version allow direct access to specific ISA implementations.

Returns

Success: return 0, else: return -1.

7.28 phy_llr_demapping.h File Reference

Data Structures

- struct bblib_llr_demapping_request
- struct bblib_llr_demapping_response
- struct bblib_llr_demapping_5gnr_request
- struct bblib_llr_demapping_5gnr_response
- struct bblib_llr_demapping_nn_5gnr_request
- struct bblib_llr_demapping_nn_5gnr_response

Enumerations

- enum bblib_llr_demapping_5gnr_io_format { BBLIB_LLR_DEMAPPING_F32_IN_F32_OUT, BBLIB_LLR_DEMAPPING_F32_I
BBLIB_LLR_DEMAPPING_I16_IN_I8_OUT }

Functions

- int16_t bblib_llr_demapping_version (char ∗version, int buffer_size)

- void bblib_llr_demapping (const struct bblib_llr_demapping_request ∗request, struct bblib_llr_demapping_response ∗response)
- void **bblib_llr_demapping_c** (const struct bblib_llr_demapping_request ∗request, struct bblib_llr_demapping_response ∗response)
- void **bblib_llr_demapping_avx2** (const struct bblib_llr_demapping_request ∗request, struct bblib_llr_demapping_response ∗response)
- void **bblib_llr_demapping_avx512** (const struct bblib_llr_demapping_request ∗request, struct bblib_llr_demapping_response ∗response)

- void bblib_llr_demapping_5gnr (const struct bblib_llr_demapping_5gnr_request ∗request, struct bblib_llr_demapping_5gnr_resp
∗response)
- void **bblib_llr_demapping_5gnr_c** (const struct bblib_llr_demapping_5gnr_request ∗request, struct bblib_llr_demapping_5gnr_response ∗response)
- void **bblib_llr_demapping_5gnr_avx2** (const struct bblib_llr_demapping_5gnr_request ∗request, struct bblib_llr_demapping_5gnr_response ∗response)
- void **bblib_llr_demapping_5gnr_avx512** (const struct bblib_llr_demapping_5gnr_request ∗request, struct bblib_llr_demapping_5gnr_response ∗response)

- int bblib_llr_demapping_nn_5gnr (const struct bblib_llr_demapping_nn_5gnr_request ∗request, struct bblib_llr_demapping_nn_5gnr_response ∗response)
- int **bblib_llr_demapping_nn_5gnr_avx2** (const struct bblib_llr_demapping_nn_5gnr_request ∗request, struct bblib_llr_demapping_nn_5gnr_response ∗response)
- int **bblib_llr_demapping_nn_5gnr_avx512** (const struct bblib_llr_demapping_nn_5gnr_request ∗request, struct bblib_llr_demapping_nn_5gnr_response ∗response)

7.28.1 Detailed Description

External API for LLR demapping for the QPSK/16QAM/64QAM/256QAM.

7.28.2 Enumeration Type Documentation

7.28.2.1 bblib_llr_demapping_5gnr_io_format

`enum bblib_llr_demapping_5gnr_io_format`

Enum for input and output data format.

Overview: The LLR demapper performs symbol to LLR demapping and supports QPSK, 16QAM, 64QAM and 256QAM modulation order. For 16QAM, 64QAM and 256QAM the LLRs are calculated using a piece-wise linear approximation of the log-likelihood ratio function.

Note: The 4G SDK implementation has a single noise and gain input value, 5G has a noise and gain value for each input symbol.

The LLR demapper supports floating point input and float or 8-bit fixed output with 3-bit quantisation. The input and output buffer allocation should be rounded up to a multiple of the register width.

Requirements and Test Coverage:

Functional Test Cases:

- QPSK with uniformly distributed amplitudes, SINR 4dB, 3600 input symbols
- 16-QAM with uniformly distributed amplitudes, low SINR (mean 7dB), 2400 input symbols
- 16-QAM with uniformly distributed amplitudes, high SINR (mean 10dB), 2400 input symbols
- 64-QAM with uniformly distributed amplitudes, low SINR (mean 13dB), 1200 input symbols
- 64-QAM with uniformly distributed amplitudes, high SINR (mean 20dB), 1200 input symbols
- 256-QAM with uniformly distributed amplitudes, high SINR (mean 25dB), 1200 input symbols
- 256-QAM with uniformly distributed amplitudes, low SINR (mean 19dB), 1200 input symbols

Functional tests are compared with a Matlab model.

Performance Test: cycle count measurements for QPSK, 16QAM, 64QAM and 256QAM with 48, 792, 1200 and 3300 input symbols

Algorithm Guidance: For a received symbol r, noise variance σ^2 and equaliser gain c, the following simple linear functions are used to approximate the LLRs:

QPSK Demapping

$LLR = \frac{-2cr}{\sigma^2}$

16QAM Piecewise Linear Demapping

LLR for bit 0

$$
\begin{aligned}
r_x \leq -2c, LLR &= \frac{-8c(r+c)}{2\sigma^2} \\
-2c < r_x \leq 2c, LLR &= \frac{-4cr}{2\sigma^2} \\
r_x > 2c, LLR &= \frac{-8c(r-c)}{2\sigma^2}
\end{aligned}
$$

LLR for bit 1

$$
\begin{aligned}
r_x > 0, LLR &= \frac{-4c(-r+2c)}{2\sigma^2} \\
r_x \leq 0, LLR &= \frac{-4c(r+2c)}{2\sigma^2}
\end{aligned}
$$

64QAM Piecewise Linear Demapping

LLR for bit 0

$$
\begin{aligned}
r_x < -6, LLR &= \frac{-16c(r+3c)}{2\sigma^2} \\
-6c < r_x \leq -4c, LLR &= \frac{-12c(r+2c)}{2\sigma^2} \\
-4c < r_x \leq -2c, LLR &= \frac{-8c(r+c)}{2\sigma^2} \\
-2c < r_x \leq 2c, LLR &= \frac{-4cr}{2\sigma^2} \\
2c < r_x \leq 4c, LLR &= \frac{-8c(r-c)}{2\sigma^2} \\
4c < r_x \leq 6c, LLR &= \frac{-12c(r-2c)}{2\sigma^2} \\
6c < r_x, LLR &= \frac{-16c(r-3c)}{2\sigma^2}
\end{aligned}
$$

LLR for bit 1

$$
\begin{aligned}
|r_x| > 6c, LLR &= \frac{-8c(-|r|+5c)}{2\sigma^2} \\
2c < |r_x| \leq 6c, LLR &= \frac{-4c(-|r|+4c)}{2\sigma^2} \\
|r_x| \leq 2c, LLR &= \frac{-8c(-|r|+3c)}{2\sigma^2}
\end{aligned}
$$

LLR for bit 2

$$\begin{aligned}|r_x| > 4c, LLR &= \frac{-4c(-|r_x| + 6c)}{2\sigma^2}\\ |r_x| \leq 4c, LLR &= \frac{-4c(|r| - 2c)}{2\sigma^2}\end{aligned}$$

256QAM Piecewise Linear Demapping

LLR for bit 0

$$\begin{aligned}r_x \leq -14c, LLR &= \frac{(-32c(r + 7c))}{(2\sigma^2}\\ -14c < r_x \leq -12c, LLR &= \frac{(-28c(r + 6c))}{(2\sigma^2}\\ -12c < r_x \leq -10c, LLR &= \frac{(-24c(r + 5c))}{(2\sigma^2}\\ -10c < r_x \leq -8c, LLR &= \frac{(-20c(r + 4c))}{(2\sigma^2}\\ -8c < r_x \leq -6c, LLR &= \frac{(-16c(r + 3c))}{(2\sigma^2}\\ -6c < r_x \leq -4c, LLR &= \frac{(-12c(r + 2c))}{(2\sigma^2}\\ -4c < r_x \leq -2c, LLR &= \frac{(-8c(r + 1c))}{(2\sigma^2}\\ -2c < r_x \leq 2c, LLR &= \frac{(-4cr)}{(2\sigma^2}\\ 2c < r_x \leq 4c, LLR &= \frac{(-8c(r - 1c))}{(2\sigma^2}\\ 4c < r_x \leq 6c, LLR &= \frac{(-12c(r - 2c))}{(2\sigma^2}\\ 6c < r_x \leq 8c, LLR &= \frac{(-16c(r - 3c))}{(2\sigma^2}\\ 8c < r_x \leq 10c, LLR &= \frac{(-20c(r - 4c))}{(2\sigma^2}\\ 10c < r_x \leq 12c, LLR &= \frac{(-24c(r - 5c))}{(2\sigma^2}\\ 12c < r_x \leq 14c, LLR &= \frac{(-28c(r - 6c))}{(2\sigma^2}\\ r_x > 14c, LLR &= \frac{(-32c(r - 7c))}{(2\sigma^2}\end{aligned}$$

LLR for bit 1

$$\begin{aligned}|r_x| \geq 14c, LLR &= \frac{4c(4|r| - 44c)}{2\sigma^2}\\ 14c > |r_x| \geq 12c, LLR &= \frac{3c(4|r| - 40c)}{2\sigma^2}\end{aligned}$$

$$\begin{aligned}
12c > |r_x| \geq 10c, LLR &= \frac{2c(4|r| - 36c)}{2\sigma^2} \\
10c > |r_x| \geq 6c, LLR &= \frac{1c(4|r| - 32c)}{2\sigma^2} \\
6c > |r_x| \geq 4c, LLR &= \frac{(2c(4|r| - 28c))}{(2\sigma^2} \\
4c > |r_x| \geq 2c, LLR &= \frac{(3c(4|r| - 24c))}{(2\sigma^2} \\
2c > |r_x| \geq 0, LLR &= \frac{(4c(4|r| - 20c))}{(2\sigma^2}
\end{aligned}$$

LLR for bit 2

$$\begin{aligned}
|r_x| \geq 14c, LLR &= \frac{4c(2|r| - 26c)}{2\sigma^2} \\
14c > |r_x| \geq 10c, LLR &= \frac{4c(|r| - 12c)}{2\sigma^2} \\
10c > |r_x| \geq 8c, LLR &= \frac{8c(|r| - 11c)}{2\sigma^2} \\
8c > |r_x| \geq 6c, LLR &= \frac{-8c(|r| - 5c)}{2\sigma^2} \\
6c > |r_x| \geq 2c, LLR &= \frac{-4c(|r| - 4c)}{2\sigma^2} \\
2c > |r_x| \geq 0, LLR &= \frac{-8c(|r| - 3c)}{2\sigma^2}
\end{aligned}$$

LLR for bit 3

$$\begin{aligned}
|r_x| \geq 12c, LLR &= \frac{4c(|r| - 14c)}{2\sigma^2} \\
12c > |r_x| \geq 8c, LLR &= \frac{-4c(|r| - 10c)}{2\sigma^2} \\
8c > |r_x| \geq 4c, LLR &= \frac{4c(|r| - 6c)}{2\sigma^2} \\
4c > |r_x| \geq 0c, LLR &= \frac{-4c(|r| - 2c)}{2\sigma^2}
\end{aligned}$$

Enumerator

BBLIB_LLR_DEMAPPING_F32_IN_F32_OUT	32 bit floating point input and output
BBLIB_LLR_DEMAPPING_F32_IN_I8_OUT	32 bit float input 8 bit fixed point output
BBLIB_LLR_DEMAPPING_I16_IN_I8_OUT	16 bit fixed input 8 bit fixed point output

7.28.3 Function Documentation

7.28.3.1 bblib_llr_demapping()

```
void bblib_llr_demapping (
            const struct bblib_llr_demapping_request * request,
            struct bblib_llr_demapping_response * response )
```

LTE LLR Demapping procedures using a piece-wise linear approximation. LTE supports floating point input and fixed or floating point output. See bblib_llr_demapping_response documentation for fixed point format and range. Data format is configured via an enum in the request structure.

Parameters

in	*request*	Structure containing the input bits, noise variance, magnitude data length and modulation order QPSK/16QAM/64QAM/256QAM.
out	*response*	Structure containing the output LLRs and the number of output LLRs.

7.28.3.2 bblib_llr_demapping_5gnr()

```
void bblib_llr_demapping_5gnr (
            const struct bblib_llr_demapping_5gnr_request * request,
            struct bblib_llr_demapping_5gnr_response * response )
```

5G NR LLR Demapping procedures using a piece-wise linear approximation. 5G NR supports fixed point input and output, floating point input and output or floating point input to fixed point output. See bblib_llr_demapping_5gnr_request and bblib_llr_demapping_5gnr_response structure documentation for fixed point formats and range. Data format is configured via an enum in the request structure.

Parameters

in	*request*	Structure containing the input bits, noise variance, magnitude data length and modulation order QPSK/16QAM/64QAM/256QAM.
out	*response*	Structure containing the output LLRs and the number of output LLRs.

7.28.3.3 bblib_llr_demapping_nn_5gnr()

```
int bblib_llr_demapping_nn_5gnr (
            const struct bblib_llr_demapping_nn_5gnr_request * request,
            struct bblib_llr_demapping_nn_5gnr_response * response )
```

Nearest Neighbour 5GNR Demapping procedures.

Parameters

in	*request*	Structure containing the input bits, noise variance, magnitude data length and modulation order QPSK/16QAM/64QAM/256QAM.
out	*response*	Structure containing the output LLRs and the number of output LLRs.

Returns

0 on success, negative on error

Warning

EXPERIMENTAL: Further optimization is possible, API may change without prior notice.

7.28.3.4 bblib_llr_demapping_version()

```
int16_t bblib_llr_demapping_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_llr_demapping library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, length BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.29 phy_lte_mu_mimo_equalize.h File Reference

Data Structures

- struct bblib_lte_mu_mimo_equalize_request
- struct bblib_lte_mu_mimo_equalize_response

Enumerations

- enum mu_mimo_puschshorten_flag { Shorten_Disable = 0, Shorten_Enable = 1 }
- enum lte_mu_mimo_equalize_config { NUM_SLOTS_PER_SUBF = 2, MAX_NUM_ANT = 8, MAX_PUSCH_DATASYMB_PER_
= 12, MAX_SYM_PER_SUBFRAME = 14 }

Functions

- int16_t bblib_lte_mu_mimo_equalize_version (char ∗version, int buffer_size)
- void bblib_lte_mu_mimo_equalize_print_version ()

- int32_t bblib_lte_mu_mimo_equalize_fxp (const struct bblib_lte_mu_mimo_equalize_request ∗request, struct bblib_lte_mu_mimo_equalize_response ∗response)
- int32_t **bblib_lte_mu_mimo_equalize_flp_c** (const struct bblib_lte_mu_mimo_equalize_request ∗request, struct bblib_lte_mu_mimo_equalize_response ∗response)
- int32_t **bblib_lte_mu_mimo_equalize_fxp_avx2** (const struct bblib_lte_mu_mimo_equalize_request ∗request, struct bblib_lte_mu_mimo_equalize_response ∗response)
- int32_t **bblib_lte_mu_mimo_equalize_fxp_avx512** (const struct bblib_lte_mu_mimo_equalize_request ∗request, struct bblib_lte_mu_mimo_equalize_response ∗response)

7.29.1 Detailed Description

External API for LTE mu_mimo channel equalization.

Overview:

The lib_mu_mimo_equalization_lte kernel is a LTE PUSCH equalizer for mu_mimo case.

Requirements and Test Coverage:

Supported 2 users mu_mimo with 2, 4 or 8 receiving antennas.

Algorithm Guidance:

The algorithm is perform MMSE filtering to obtain the symbol after equalization.

7.29.2 Enumeration Type Documentation

7.29.2.1 lte_mu_mimo_equalize_config

```
enum lte_mu_mimo_equalize_config
```

number of slot per subframe, max antenna number, max pusch data symbol per subframe and max number of symbols per subframe.

Enumerator

NUM_SLOTS_PER_SUBF	Number of slot per subframe
MAX_NUM_ANT	Max number of receiving antennas
MAX_PUSCH_DATASYMB_PER_SUBF	Max number of PUSCH data symbols per subframe
MAX_SYM_PER_SUBFRAME	Max number of symbols per subframe

7.29.2.2 mu_mimo_puschshorten_flag

```
enum mu_mimo_puschshorten_flag
```

pusch shorten flag for LTE mu_mimo.

Enumerator

Shorten_Disable	PUSCH shorten disabled
Shorten_Enable	PUSCH shorten enabled

7.29.3 Function Documentation

7.29.3.1 bblib_lte_mu_mimo_equalize_fxp()

```
int32_t bblib_lte_mu_mimo_equalize_fxp (
            const struct bblib_lte_mu_mimo_equalize_request * request,
            struct bblib_lte_mu_mimo_equalize_response * response )
```

lte mu_mimo equalize procedures.

Parameters

in	*request*	structure containing the input data and parameteer.
out	*response*	structure containing the output data.

Returns

int: status. 0 on success, non 0 on error

7.29.3.2 bblib_lte_mu_mimo_equalize_version()

```
int16_t bblib_lte_mu_mimo_equalize_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_lte_mu_mimo_equalize library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.30 phy_lte_su_mimo_equalize.h File Reference

Data Structures

- struct bblib_lte_su_mimo_equalize_request
- struct bblib_lte_su_mimo_equalize_response

Enumerations

- enum puschshorten_flag { Shorten_Disable = 0, Shorten_Enable = 1 }
- enum lte_su_mimo_equalize_config { NUM_SLOTS_PER_SUBF = 2, MAX_NUM_ANT = 8, MAX_PUSCH_DATASYMB_PER_ = 12 }

Functions

- int16_t bblib_lte_su_mimo_equalize_version (char ∗version, int buffer_size)
- void bblib_lte_su_mimo_equalize_print_version ()

- int32_t bblib_lte_su_mimo_equalize_fxp (const struct bblib_lte_su_mimo_equalize_request ∗request, struct bblib_lte_su_mimo_equalize_response ∗response)
- int32_t **bblib_lte_su_mimo_equalize_flp_c** (const struct bblib_lte_su_mimo_equalize_request ∗request, struct bblib_lte_su_mimo_equalize_response ∗response)
- int32_t **bblib_lte_su_mimo_equalize_fxp_c** (const struct bblib_lte_su_mimo_equalize_request ∗request, struct bblib_lte_su_mimo_equalize_response ∗response)
- int32_t **bblib_lte_su_mimo_equalize_fxp_avx2** (const struct bblib_lte_su_mimo_equalize_request ∗request, struct bblib_lte_su_mimo_equalize_response ∗response)
- int32_t **bblib_lte_su_mimo_equalize_fxp_avx512** (const struct bblib_lte_su_mimo_equalize_request ∗request, struct bblib_lte_su_mimo_equalize_response ∗response)

7.30.1 Detailed Description

External API for su_mimo LTE channel equalization.

Overview:

The lib_su_mimo_equalization_lte kernel is a LTE PUSCH equalizer for su_mimo case.

Requirements and Test Coverage:

Supported single user with 1, 2, 4 or 8 receiving antennas.

Algorithm Guidance:

The algorithm is perform MRC and channel estimate compensation for single user.

7.30.2 Enumeration Type Documentation

7.30.2.1 lte_su_mimo_equalize_config

```
enum lte_su_mimo_equalize_config
```

number of slot per subframe, max antenna number and max pusch data symbol per subframe.

Enumerator

NUM_SLOTS_PER_SUBF	Number of slot per subframe
MAX_NUM_ANT	Max number of receiving antennas
MAX_PUSCH_DATASYMB_PER_SUBF	Max number of PUSCH data symbols per subframe

7.30.2.2 puschshorten_flag

```
enum puschshorten_flag
```

pusch shorten flag for LTE.

Enumerator

Shorten_Disable	PUSCH shorten disabled
Shorten_Enable	PUSCH shorten enabled

7.30.3 Function Documentation

7.30.3.1 bblib_lte_su_mimo_equalize_fxp()

```
int32_t bblib_lte_su_mimo_equalize_fxp (
            const struct bblib_lte_su_mimo_equalize_request * request,
            struct bblib_lte_su_mimo_equalize_response * response )
```

LTE su_mimo equalize procedures.

Parameters

in	*request*	structure containing the input data and parameteer.
out	*response*	structure containing the output data.

Returns

int: status. 0 on success, non 0 on error

7.30.3.2 bblib_lte_su_mimo_equalize_version()

```
int16_t bblib_lte_su_mimo_equalize_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_lte_su_mimo_equalize library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.31 phy_matrix_inversion.h File Reference

Data Structures

- struct bblib_matrix_inv_request
- struct bblib_matrix_inv_response
- struct bblib_matrix_inv_interleave_request
- struct bblib_matrix_inv_interleave_response

Enumerations

- enum bblib_matrix_inv_type {
  BBLIB_MATRIX_INV_2x2, BBLIB_MATRIX_INV_2x2_HERMITIAN, BBLIB_MATRIX_INV_4x4, BBLIB_MATRIX_INV_4x4_HE
  BBLIB_MATRIX_INV_6x6, BBLIB_MATRIX_INV_6x6_HERMITIAN, BBLIB_MATRIX_INV_8x8, BBLIB_MATRIX_INV_8x8_HE
  BBLIB_MATRIX_INV_16x16, BBLIB_MATRIX_INV_16x16_HERMITIAN }

Functions

- int16_t bblib_matrix_inv_version (char ∗version, int buffer_size)

- int bblib_matrix_inverse (const struct bblib_matrix_inv_request ∗request, struct bblib_matrix_inv_response ∗response)
- int **bblib_matrix_inverse_c** (const struct bblib_matrix_inv_request ∗request, struct bblib_matrix_inv_response ∗response)
- int **bblib_matrix_inverse_sse** (const struct bblib_matrix_inv_request ∗request, struct bblib_matrix_inv_response ∗response)
- int **bblib_matrix_inverse_avx2** (const struct bblib_matrix_inv_request ∗request, struct bblib_matrix_inv_response ∗response)
- int **bblib_matrix_inverse_avx512** (const struct bblib_matrix_inv_request ∗request, struct bblib_matrix_inv_response ∗response)

- int bblib_matrix_inverse_interleave (const struct bblib_matrix_inv_interleave_request ∗request, struct bblib_matrix_inv_interleave_response ∗response)
- int **bblib_matrix_inverse_interleave_c** (const struct bblib_matrix_inv_interleave_request ∗request, struct bblib_matrix_inv_interleave_response ∗response)
- int **bblib_matrix_inverse_interleave_sse** (const struct bblib_matrix_inv_interleave_request ∗request, struct bblib_matrix_inv_interleave_response ∗response)
- int **bblib_matrix_inverse_interleave_avx2** (const struct bblib_matrix_inv_interleave_request ∗request, struct bblib_matrix_inv_interleave_response ∗response)
- int **bblib_matrix_inverse_interleave_avx512** (const struct bblib_matrix_inv_interleave_request ∗request, struct bblib_matrix_inv_interleave_response ∗response)

7.31.1 Detailed Description

External API for Matrix Inversion.

Overview:

The lib_matrix_inversion kernel implements Matrix Inversion for two types of floating point complex number matrices:

- Standard square matrices.
- Hermitian Positive Definite square matrices.

There are two separate implementations supported by the kernel based on the order of the input matrix data and expected order of the output matrix data. One implementation supports matrix inversion for non-interleaved input data, while the second implementation supports matrix inversion for interleaved input data.

The kernel supports matrix inversion of 2x2, 4x4, 6x6, 8x8 and 16x16 matrices.

Requirements and Test Coverage:

Functional Test Cases:

- Non-interleaved matrix inputs for square matrices of size 2x2, 4x4, 6x6, 8x8 and 16x16.
- Non-interleaved matrix inputs for Hermitian Positive Definite sqaure matrices of size 2x2, 4x4, 6x6, 8x8 and 16x16.
- Interleaved matrix inputs for square matrices of size 2x2, 4x4, 6x6, 8x8 and 16x16.
- Interleaved matrix inputs for Hermitian Positive Definite square matrices of size 2x2, 4x4, 6x6, 8x8 and 16x16.

Functional test outputs are checked by multiplying the kernel output by the test input data. The result of the multiplication should match the Identity matrix of that size, to within a difference of 0.005 between each position's value in the result matrix and equivalent value in the Identity matrix.

Performance Test Cases:

- Cycle count measurements for non-interleaved matrix inputs of square matrices of size 2x2, 4x4, 6x6, 8x8 and 16x16.
- Cycle count measurements for non-interleaved matrix inputs of Hermitian Positive Definite square matrices of size 2x2, 4x4, 6x6, 8x8 and 16x16.
- Cycle count measurements for interleaved matrix inputs of square matrices of size 2x2, 4x4, 6x6, 8x8 and 16x16.
- Cycle count measurements for interleaved matrix inputs of Hermitian Positive Definite square matrices of size 2x2, 4x4, 6x6, 8x8 and 16x16.

7.31.2 Enumeration Type Documentation

7.31.2.1 bblib_matrix_inv_type

```
enum bblib_matrix_inv_type
```

Enum describing the type of operation to perform.

Enumerator

BBLIB_MATRIX_INV_2x2	a square matrix of size 2x2.
BBLIB_MATRIX_INV_2x2_HERMITIAN	a Hermitian Positive Definite square matrix of size 2x2.
BBLIB_MATRIX_INV_4x4	a square matrix of size 4x4.
BBLIB_MATRIX_INV_4x4_HERMITIAN	a Hermitian Positive Definite square matrix of size 4x4.
BBLIB_MATRIX_INV_6x6	a square matrix of size 6x6.
BBLIB_MATRIX_INV_6x6_HERMITIAN	a Hermitian Positive Definite square matrix of size 6x6.
BBLIB_MATRIX_INV_8x8	a square matrix of size 8x8.
BBLIB_MATRIX_INV_8x8_HERMITIAN	a Hermitian Positive Definite square matrix of size 8x8.
BBLIB_MATRIX_INV_16x16	a square matrix of size 16x16.
BBLIB_MATRIX_INV_16x16_HERMITIAN	a Hermitian Positive Definite square matrix of size 16x16.

7.31.3 Function Documentation

7.31.3.1 bblib_matrix_inv_version()

```
int16_t bblib_matrix_inv_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_matrix_operation library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, typically no more than BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.31.3.2 bblib_matrix_inverse()

```
int bblib_matrix_inverse (
            const struct bblib_matrix_inv_request * request,
            struct bblib_matrix_inv_response * response )
```

Computes the inverse of a Complex square matrix.

Parameters

in	*request*	Pointer to request structure containing the information required to perform the matrix inversion.
in,out	*response*	Pointer to a response structure that will contain the output of the operation.

Returns

0 if the output array in the response structure was populated, otherwise -1.

7.31.3.3 bblib_matrix_inverse_interleave()

```
int bblib_matrix_inverse_interleave (
            const struct bblib_matrix_inv_interleave_request * request,
            struct bblib_matrix_inv_interleave_response * response )
```

Computes the inverse of a Complex square matrix from interleaved inputs.

Parameters

in	*request*	Pointer to request structure containing the information required to perform the matrix inversion.
in,out	*response*	Pointer to a response structure that will contain the output of the operation.

Returns

0 if the output array in the response structure was populated, otherwise -1.

7.32 phy_mmse_irc_mimo_5gnr.h File Reference

Data Structures

- struct bblib_mmse_irc_mimo_5gnr_request
- struct bblib_mmse_irc_mimo_5gnr_response

Functions

- int16_t bblib_mmse_irc_mimo_5gnr_version (char *version, int buffer_size)

- int32_t bblib_mmse_irc_mimo_5gnr (bblib_mmse_irc_mimo_5gnr_request *request, bblib_mmse_irc_mimo_5gnr_response *response)
- int32_t **bblib_mmse_irc_mimo_5gnr_avx512** (bblib_mmse_irc_mimo_5gnr_request *request, bblib_mmse_irc_mimo_5gnr_r *response)

7.32.1 Detailed Description

External API for MMSE-IRC MIMO detection for 5GNR.

Overview:

The lib_mmse_irc_mimo_5gnr kernel is a 5G MMSE-IRC MIMO detection. This is a generic kernel which can be used in equalization in both UL and DL.

Algorithm Guidance:

The 5G NR mmse-irc is using the outer-product of mmse-irc for the 1x2/2x2, as: estX = H' ∗ inv(H ∗ H' + Rnn) ∗ Y And the mmse gain also as an output, as: gain = H' ∗ inv(H ∗ H' + Rnn) ∗ H

7.32.2 Function Documentation

7.32.2.1 bblib_mmse_irc_mimo_5gnr()

```
int32_t bblib_mmse_irc_mimo_5gnr (
            bblib_mmse_irc_mimo_5gnr_request * request,
            bblib_mmse_irc_mimo_5gnr_response * response )
```

MMSE-IRC MIMO detection, with MMSE gain calculation.

Parameters

in	*request*	Input request structure for MMSE-IRC.
out	*response*	Output response structure for MMSE-IRC..

Returns

0 for success, and -1 for error

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.32.2.2 bblib_mmse_irc_mimo_5gnr_version()

```
int16_t bblib_mmse_irc_mimo_5gnr_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_mmse_irc_mimo_5gnr library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.33 phy_mmse_mimo_qrd_5gtf.h File Reference

Data Structures

- struct bblib_mimo_mmse_5gqrd_request
- struct bblib_mimo_mmse_5gqrd_response

Functions

- int16_t bblib_mmse_mimo_qrd_5g_version (char ∗version, int buffer_size)

- int32_t bblib_mimo_mmse_5gqrd_4x4_weight (const struct bblib_mimo_mmse_5gqrd_request ∗request, struct bblib_mimo_mmse_5gqrd_response ∗response)
- int32_t **bblib_mimo_mmse_5gqrd_4x4_weight_avx2** (const struct bblib_mimo_mmse_5gqrd_request ∗request, struct bblib_mimo_mmse_5gqrd_response ∗response)

- int32_t bblib_mimo_mmse_5gqrd_8x8_weight (const struct bblib_mimo_mmse_5gqrd_request ∗request, struct bblib_mimo_mmse_5gqrd_response ∗response)
- int32_t **bblib_mimo_mmse_5gqrd_8x8_weight_avx2** (const struct bblib_mimo_mmse_5gqrd_request ∗request, struct bblib_mimo_mmse_5gqrd_response ∗response)

7.33.1 Detailed Description

External API for 5G mimo mmse qrd functions.

7.33.2 Function Documentation

7.33.2.1 bblib_mimo_mmse_5gqrd_4x4_weight()

```
int32_t bblib_mimo_mmse_5gqrd_4x4_weight (
            const struct bblib_mimo_mmse_5gqrd_request * request,
            struct bblib_mimo_mmse_5gqrd_response * response )
```

MMSE based Channel Estimation for MIMO 4x4.

Parameters

in	*request*	Request structure with input parameters.
out	*response*	Responsestructure with the output parameter.

Returns

0 is successful, otherwise -1.

7.33.2.2 bblib_mimo_mmse_5gqrd_8x8_weight()

```
int32_t bblib_mimo_mmse_5gqrd_8x8_weight (
            const struct bblib_mimo_mmse_5gqrd_request * request,
            struct bblib_mimo_mmse_5gqrd_response * response )
```

MMSE based Channel Estimation for MIMO 8x8.

Parameters

in	*request*	Request structure with input parameters.
out	*response*	Responsestructure with the output parameter.

Returns

0 is successful, otherwise -1.

7.33.2.3 bblib_mmse_mimo_qrd_5g_version()

```
int16_t bblib_mmse_mimo_qrd_5g_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_mmse_mimo_qrd_5g_version library.

Parameters

in	*version*	pointer to a char buffer where the version string should be copied.
in	*buffer_size*	the length of the string buffer, typically no more than BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.34 phy_modulation.h File Reference

Data Structures

- struct bblib_modulation_request
- struct bblib_modulation_response

Functions

- int16_t bblib_modulation_version (char *version, int buffer_size)

- void bblib_modulation (const struct bblib_modulation_request *request, struct bblib_modulation_response *response)
- void **bblib_modulation_snc** (const struct bblib_modulation_request *request, struct bblib_modulation_response *response)
- void **bblib_modulation_avx512** (const struct bblib_modulation_request *request, struct bblib_modulation_response *response)
- void **bblib_modulation_sse** (const struct bblib_modulation_request *request, struct bblib_modulation_response *response)
- void **bblib_modulation_c** (const struct bblib_modulation_request *request, struct bblib_modulation_response *response)

7.34.1 Detailed Description

Modulation mapper for LTE and 5GNR that conforms to 3GPP TS 36.211 V12.4.0 section 7.1 (LTE) and 3GPP TS 38.211 V15.0.0 section 5.1 (5GNR).

Overview:

The lib_modulation takes input bits and maps them to the IQ samples as defined in the 3GPP specification.

The following bit order is assumed in the byte: [MSB] b(i)b(i+1)...b(i+7) [LSB].

Requirements and Test Coverage:

Supported modulation orders are QPSK, 16QAM, 64QAM and 256QAM.

Test coverage for all modulation orders: Functional check; Cycle count measurment.

The lib_modulation implementation output must conform to 3GPP modulation specification. This kernel has been proven to work in the FlexRAN LTE refPHY system.

Algorithm Guidance:

Information below only applies to the AVX512 implemenetation. SSE and C implementations use a simple lookup.

All orders of mapping implementations exploit two essential features:

- Constellations have symmetry. The uper right quadrant is reflected across the axes. This means that in most cases the code can generate one quadrant and then flip sign bits to generate the other quadrants. Flipping sign bits is very fast.
- AVX512 provides a number of fast register-based lookups which makes it possible to do most lookups in one or two instructions.

Typically these features result in mapping functions working by generating one quadrant using a table lookup from register, followed by sign-bit flipping to put the data into the right quadrant. The exception is QPSK which can do a direct lookup using mask bits.

7.34.2 Function Documentation

7.34.2.1 bblib_modulation()

```
void bblib_modulation (
            const struct bblib_modulation_request * request,
            struct bblib_modulation_response * response )
```

Functions that performs fixed point modulation mapping for input paremetrs given in the request structure and returns result in the response structure.

The first variant of the function, bblib_modulation, auto-selects isa on the compile time based on the CPU architecture. It defaults to the highest possible ISA. Others variants of this function provide direct access to the ISA specific implementation. The ISA independent function should be used as it will prevent errors when building on the architecture that does not support a selected ISA. The ISA specific functions are exposed to enable better test coverage and allow fixing the implementation in some special cases.

Parameters

in	*request*	Structure containing buffer with input bits, length of the input data in bytes and the modulation order.
out	*response*	Structure containing the complex output symbols buffer and the number of output symbols in the buffer.

Note

bblib_modulation_sse does not support 256QAM modulation.

7.34.2.2 bblib_modulation_version()

```
int16_t bblib_modulation_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_lte_modulation library.

Parameters

in	*version*	Pointer to the char buffer where the version string is be copied.
in	*buffer_size*	The length of the string buffer. Must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

Returns 0 if the version string was populated, -1 otherwise.

7.35 phy_nr_zc_sequence_gen.h File Reference

Data Structures

- struct bblib_nr_zc_sequence_gen_response
- struct bblib_nr_zc_sequence_gen_request

Functions

- int32_t bblib_nr_zc_sequence_gen (const struct bblib_nr_zc_sequence_gen_request *request, struct bblib_nr_zc_sequence_gen_response *response)

7.35.1 Detailed Description

External API for Zadoff-Chu sequence generator.

Overview:

The lib_zc_sequence_gen kernel is a Zadoff-Chu sequence generator which supports both LTE, based on 3GPP TS 36.211 Release 14.2.0, sections 5.5.1 and 5.5.2, and 5G NR, based on 3GPP TS 38.211, sections 5.5.1 and 6.4.1, implementations.

Algorithm Guidance:

7.35.2 Function Documentation

7.35.2.1 bblib_nr_zc_sequence_gen()

```
int32_t bblib_nr_zc_sequence_gen (
            const struct bblib_nr_zc_sequence_gen_request * request,
            struct bblib_nr_zc_sequence_gen_response * response )
```

zc sequence generation procedures.

Parameters

in	*request*	Input request structure for ZC sequence generation.
out	*response*	response Output response structure for ZC sequence generation.

Returns

0 for success, and -1 for error

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.36 phy_polar_decoder_5gnr.h File Reference

Data Structures

- struct bblib_polar_decoder_5gnr_request
- struct bblib_polar_decoder_5gnr_response

Enumerations

- enum { k_max_order = 10, k_max_list_size = 8, k_max_codeword_size = 1 << k_max_order, k_cache_line_size = 64 }

Functions

- int16_t bblib_polar_decoder_5gnr_version (char *version, int buffer_size)

- void bblib_polar_decoder_5gnr (const struct bblib_polar_decoder_5gnr_request *request, struct bblib_polar_decoder_5gnr_res *response)
- void **bblib_polar_decoder_5gnr_avx2** (const struct bblib_polar_decoder_5gnr_request *request, struct bblib_polar_decoder_5gnr_response *response)
- void **bblib_polar_decoder_5gnr_avx512** (const struct bblib_polar_decoder_5gnr_request *request, struct bblib_polar_decoder_5gnr_response *response)

- void bblib_polar_list_decoder_5gnr (const struct bblib_polar_decoder_5gnr_request *request, struct bblib_polar_decoder_5gnr_response *response)
- void **bblib_polar_list_decoder_5gnr_avx2** (const struct bblib_polar_decoder_5gnr_request *request, struct bblib_polar_decoder_5gnr_response *response)
- void **bblib_polar_list_decoder_5gnr_avx512** (const struct bblib_polar_decoder_5gnr_request *request, struct bblib_polar_decoder_5gnr_response *response)

- void bblib_polar_parity_list_decoder_5gnr (const struct bblib_polar_decoder_5gnr_request *request, struct bblib_polar_decoder_5gnr_response *response)
- void **bblib_polar_parity_list_decoder_5gnr_avx2** (const struct bblib_polar_decoder_5gnr_request *request, struct bblib_polar_decoder_5gnr_response *response)
- void **bblib_polar_parity_list_decoder_5gnr_avx512** (const struct bblib_polar_decoder_5gnr_request *request, struct bblib_polar_decoder_5gnr_response *response)

7.36.1 Detailed Description

External API for Polar Decoder 5G NR functions.

The SSC List-1 Decoders are Simplified Successive Cancellation Decoders. They also perform CRC11 checking of the decoded message (CA-Polar). Both decoders operate by identifying: Rate-0 nodes, Rate-1 nodes, SPC-nodes and Repetition-nodes. The emitted message is the decoded message that passes the CRC11 check, if message doesn't pass the check then no message is emitted.

The SSC List-8 Decoders are Simplified Successive Cancellation List Decoders, with the list-depth equal to 8. They also perform CRC11 checking of the surviving 8 list members (CA-Polar). Both decoders operate by identifying: Rate-0 nodes, Rate-1 nodes, SPC-nodes and Repetition-nodes. The emitted message is the first list member that passes the CRC11 check, if no member passes the check then no message is emitted.

The SSC List-8 Parity Decoders are Simplified Successive Cancellation List Decoders, with list-depth equal to 8. They also perform CRC6 checking of the surviving 8 list members (PC-CA-Polar). Both decoders operate by identifying: Rate-0 nodes, and Repetition-nodes. Parity-bit list-pruning is employed when a parity-bit is encountered. The emitted message is the first list member that passes the CRC6 check, if no member passes the check then no message is emitted.

In all of the decoders, the input LLRs are assumed to be 8-bit signed integers. Internally, 16-bit signed integer arithmetic is used. In List-8 decoders Metric-growth is normalised so that the lowest list metric is always zero. Shortening is handled by passing in the highest representable positive input LLR.

7.36.2 Enumeration Type Documentation

7.36.2.1 anonymous enum

`anonymous enum`

This configuration sets global constants and macros which are of general use throughout the module.

All current IA processors of interest align their cache lines on this boundary. If the cache alignment for future processors changes then the most restrictive alignment should be set.

Enumerator

k_max_order	Maximum order of the decoder - maximum codeword size depends on this value.
k_max_list_size	Maximum size of list-depth.
k_max_codeword_size	2^k_maxOrder.
k_cache_line_size	Cache line size, for alignment.

7.36.3 Function Documentation

7.36.3.1 bblib_polar_decoder_5gnr()

```
void bblib_polar_decoder_5gnr (
            const struct bblib_polar_decoder_5gnr_request * request,
            struct bblib_polar_decoder_5gnr_response * response )
```

Polar Decoder 5G NR: ISA variant recursive integer polar decoder.

Parameters

in	*request*	Structure containing the input frozen and LLRs.
out	*response*	Structure containing the output codewords and messages, and the length of the message in bits.

Note

the parity_bits field in the request structure is not used by this decoder.
the metrics field in the response is not used.

7.36.3.2 bblib_polar_decoder_5gnr_version()

```
int16_t bblib_polar_decoder_5gnr_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_polar_decoder_5gnr library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, typically no more than BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.36.3.3 bblib_polar_list_decoder_5gnr()

```
void bblib_polar_list_decoder_5gnr (
            const struct bblib_polar_decoder_5gnr_request * request,
            struct bblib_polar_decoder_5gnr_response * response )
```

Polar Decoder 5G NR: ISA variant recursive integer polar list 8 decoder.

Parameters

in	*request*	Structure containing the input frozen bits and LLRs.
out	*response*	Structure containing the output codewords and messages, metrics and the length of the message in bits.

Note

the parity_bits field in the request structure is not used by this decoder.

7.36.3.4 bblib_polar_parity_list_decoder_5gnr()

```
void bblib_polar_parity_list_decoder_5gnr (
            const struct bblib_polar_decoder_5gnr_request * request,
            struct bblib_polar_decoder_5gnr_response * response )
```

Polar Decoder 5G NR: ISA variant recursive integer polar parity list 8 decoder.

Parameters

in	*request*	Structure containing the input frozen and parity bits, and LLRs.
out	*response*	Structure containing the output codewords and messages, metrics and the length of the message in bits.

7.37 phy_polar_encoder_5gnr.h File Reference

Data Structures

- struct bblib_polar_tables
- struct bblib_polar_encoder_5gnr_request
- struct bblib_polar_encoder_5gnr_response

Functions

- int16_t bblib_polar_encoder_5gnr_version (char *version, int buffer_size)
- void bblib_print_polar_encoder_5gnr_version ()

- void bblib_polar_encoder_5gnr (const struct bblib_polar_encoder_5gnr_request *request, struct bblib_polar_encoder_5gnr_res *response)
- void **bblib_polar_encoder_5gnr_avx512** (const struct bblib_polar_encoder_5gnr_request *request, struct bblib_polar_encoder_5gnr_response *response)
- void **bblib_polar_encoder_5gnr_avx2** (const struct bblib_polar_encoder_5gnr_request *request, struct bblib_polar_encoder_5gnr_response *response)
- void **bblib_polar_encoder_5gnr_c** (const struct bblib_polar_encoder_5gnr_request *request, struct bblib_polar_encoder_5gnr_response *response)

7.37.1 Detailed Description

External API for 5gnr polar encoder.

Overview:

The lib_polar_encoder_5gnr kernel is a 5G NR polar encoder. This is a generic kernel which can be used to encode the polar code for PBCH, PUCCH and PDCCH from chennel coding to rate matching The algorithm is implemented as defined in TS38.212 section 5.3.1, 5.4.1, 6.3.1, 7.1 and 7.3(v15.0.0).

Requirements and Test Coverage:

K is input Input sequence length of channel coding E is Output sequence length of rate matching

for PBCH, E is 56, K is 864. for PUCCH, E need larger than 17, K need no larger than 8192. for PDCCH, E need larger than 35, K need no larger than 8192.

Algorithm Guidance:

The algorithm is implemented as defined in TS38.212 section 5.3.1, 5.4.1, 6.3.1, 7.1 and 7.3(v15.0.0).The CRC attachment has not been implemented.

7.37.2 Function Documentation

7.37.2.1 bblib_polar_encoder_5gnr()

```
void bblib_polar_encoder_5gnr (
            const struct bblib_polar_encoder_5gnr_request * request,
            struct bblib_polar_encoder_5gnr_response * response )
```

polar_encoder_5gnr procedures.

Parameters

in	*request*	structure containing the input data and parameteer, need 64 byte alignment for dataIn.
out	*response*	structure containing the output data, need 64 byte alignment for dataOut.

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.37.2.2 bblib_polar_encoder_5gnr_version()

```
int16_t bblib_polar_encoder_5gnr_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_polar_encoder_5gnr library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.38 phy_polar_encoder_internal_5gnr.h File Reference

7.38.1 Detailed Description

External API for 5gnr polar encoder.

Overview:

define some constant and structure for polar encoder.

7.39 phy_polar_rate_dematching_5gnr.h File Reference

Data Structures

- struct bblib_polar_rate_dematching_5gnr_request
- struct bblib_polar_rate_dematching_5gnr_response

Functions

- void bblib_print_polar_rate_dematching_5gnr_version ()
- int16_t bblib_polar_rate_dematching_5gnr_version (char ∗version, int buffer_size)

- int bblib_polar_rate_dematching_5gnr (const struct bblib_polar_rate_dematching_5gnr_request ∗request, struct bblib_polar_rate_dematching_5gnr_response ∗response)
- int **bblib_polar_rate_dematching_5gnr_avx2** (const struct bblib_polar_rate_dematching_5gnr_request ∗request, struct bblib_polar_rate_dematching_5gnr_response ∗response)
- int **bblib_polar_rate_dematching_5gnr_avx512** (const struct bblib_polar_rate_dematching_5gnr_request ∗request, struct bblib_polar_rate_dematching_5gnr_response ∗response)

7.39.1 Detailed Description

External API for 5gnr polar rate_dematching and rate dematching.

Overview: This module performs rate dematching for 5GNR polar decoding

7.39.2 Function Documentation

7.39.2.1 bblib_polar_rate_dematching_5gnr()

```
int bblib_polar_rate_dematching_5gnr (
            const struct bblib_polar_rate_dematching_5gnr_request * request,
            struct bblib_polar_rate_dematching_5gnr_response * response )
```

bblib_polar_rate_dematching_5gnr procedures

Parameters

in	*request*	Structure containing the input data which need to be 64 bytes alignment.
out	*response*	Structure containing the decoding output data which need 64 byte alignment.

Returns

0 if successful, negative on error

7.39.2.2 bblib_polar_rate_dematching_5gnr_version()

```
int16_t bblib_polar_rate_dematching_5gnr_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_polar_rate_dematching_5gnr library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.40 phy_prach_5gnr.h File Reference

Data Structures

- struct bblib_prach_5gnr_zc_gen_request
- struct bblib_prach_5gnr_zc_gen_response

- struct bblib_prach_5gnr_detect_request
- struct bblib_prach_5gnr_detect_response
- struct bblib_prach_5gnr_threshold_request
- struct bblib_prach_5gnr_threshold_response
- struct bblib_prach_5gnr_threshold_uplift_request
- struct bblib_prach_5gnr_threshold_uplift_response

Enumerations

- enum bblib_prach_5gnr_format { **PRACH_FORMAT_0** = 0, **PRACH_FORMAT_1**, **PRACH_FORMAT_2**, **PRACH_FORMAT_3**, **PRACH_FORMAT_A1**, **PRACH_FORMAT_A2**, **PRACH_FORMAT_A3**, **PRACH_FORMAT_B1**, **PRACH_FORMAT_B2**, **PRACH_FORMAT_B3**, **PRACH_FORMAT_B4**, **PRACH_FORMAT_C0**, **PRACH_FORMAT_C2**, **PRACH_FORMAT_LAST** }

Functions

- int16_t bblib_prach_5gnr_version (char ∗version, int buffer_size)

- int32_t bblib_prach_5gnr_zc_gen (const struct bblib_prach_5gnr_zc_gen_request ∗request, struct bblib_prach_5gnr_zc_gen_response ∗response)
- int32_t **bblib_prach_5gnr_zc_gen_c** (const struct bblib_prach_5gnr_zc_gen_request ∗request, struct bblib_prach_5gnr_zc_gen_response ∗response)
- int32_t bblib_prach_5gnr_detect (const struct bblib_prach_5gnr_detect_request ∗request, struct bblib_prach_5gnr_detect_resp ∗response)
- int32_t **bblib_prach_5gnr_detect_avx512** (const struct bblib_prach_5gnr_detect_request ∗request, struct bblib_prach_5gnr_detect_response ∗response)
- int32_t bblib_prach_5gnr_threshold (const struct bblib_prach_5gnr_threshold_request ∗request, struct bblib_prach_5gnr_threshold_response ∗response)
- int32_t **bblib_prach_5gnr_threshold_avx2** (const struct bblib_prach_5gnr_threshold_request ∗request, struct bblib_prach_5gnr_threshold_response ∗response)
- int32_t **bblib_prach_5gnr_threshold_avx512** (const struct bblib_prach_5gnr_threshold_request ∗request, struct bblib_prach_5gnr_threshold_response ∗response)
- int32_t bblib_prach_5gnr_threshold_uplift (const struct bblib_prach_5gnr_threshold_uplift_request ∗request, struct bblib_prach_5gnr_threshold_uplift_response ∗response)
- int32_t **bblib_prach_5gnr_threshold_uplift_c** (const struct bblib_prach_5gnr_threshold_uplift_request ∗request, struct bblib_prach_5gnr_threshold_uplift_response ∗response)

7.40.1 Detailed Description

External APIs for 5G PRACH kernel.

Overview:

The prach_5ngr module is intended for use within a 5G NR PRACH pipeline.

The module contains two separate sub-kernels each with their own API.

prach_5gnr_zc_gen

The prach_5gnr_zc_gen sub_kernel is a fixed point 5G NR PRACH Zadoff-chu sequence generator conforming to 3GPP TS 38.211 section 6.3.3.

prach_5gnr_detect

The prach_5gnr_detect sub kernel is a fixed point 5G NR PRACH preamble detector. The input to this function is the IFFT of the correlation of the received signal and a Zadoff-chu sequence.

Algorithm Guidance:

The prach_5gnr_zc_gen function in this module performs frequency domain Zadoff-chu sequence generation based on logical sequence index.

The frequency domain Zadoff-chu sequence must then be correlated with the frequency domain received signal, and the result transformed to the time domain using an IFFT.

NB - The frequency domain correlation and IFFT functionality are not in the scope of this kernel. This functionality is covered by other modules in the FlexRAN SDK.

The prach_5gnr_detect function in this module inputs the time domain signal and performs non-coherent combining over all antennas and repeated symbols. The power delay profile (PDP) is obtained to conduct the noise estimation, peak-search function over cyclic shift specific windows and threshold comparison. The timing advanced (TA) value are estimated for each preamble.

7.40.2 Function Documentation

7.40.2.1 bblib_prach_5gnr_detect()

```
int32_t bblib_prach_5gnr_detect (
            const struct bblib_prach_5gnr_detect_request * request,
            struct bblib_prach_5gnr_detect_response * response )
```

bblib_prach_5gnr_detect

Parameters

in	*request*	Structure containing the input buffers and lengths.
out	*response*	Structure containing the output buffer.

Returns

0 if successful, negative on error

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.40.2.2 bblib_prach_5gnr_threshold()

```
int32_t bblib_prach_5gnr_threshold (
            const struct bblib_prach_5gnr_threshold_request * request,
            struct bblib_prach_5gnr_threshold_response * response )
```

bblib_prach_5gnr_threshold

Parameters

in	*request*	Structure containing the input buffers and lengths.
out	*response*	Structure containing the output buffer.

Returns

0 if successful, negative on error

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.40.2.3 bblib_prach_5gnr_threshold_uplift()

```
int32_t bblib_prach_5gnr_threshold_uplift (
            const struct bblib_prach_5gnr_threshold_uplift_request * request,
            struct bblib_prach_5gnr_threshold_uplift_response * response )
```

bblib_prach_5gnr_threshold_uplift

Parameters

in	*request*	Structure containing the input buffers and lengths.
out	*response*	Structure containing the output buffer.

Returns

0 if successful, negative on error

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.40.2.4 bblib_prach_5gnr_version()

```
int16_t bblib_prach_5gnr_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_prach_5gnr library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, length BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.40.2.5 bblib_prach_5gnr_zc_gen()

```
int32_t bblib_prach_5gnr_zc_gen (
            const struct bblib_prach_5gnr_zc_gen_request * request,
            struct bblib_prach_5gnr_zc_gen_response * response )
```

bblib_prach_5gnr_zc_gen

Parameters

in	*request*	Structure containing the input buffers and lengths.
out	*response*	Structure containing the output buffer.

Returns

0 if successful, negative on error

Note

The 5GNR PRACH Zadoff-chu generation is performed once on cell setup therefore the cycle count of this function is not important. For this reason this is written in un-optimized c-code.

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.41 phy_precoding.h File Reference

Data Structures

- struct bblib_precoding_request
- struct bblib_precoding_response

Functions

- int16_t bblib_lte_precoding_version (char *version, int buffer_size)

- int32_t bblib_precoding (const struct bblib_precoding_request *request, struct bblib_precoding_response *response)
- int32_t **bblib_precoding_avx512** (const struct bblib_precoding_request *request, struct bblib_precoding_response *response)
- int32_t **bblib_precoding_avx2** (const struct bblib_precoding_request *request, struct bblib_precoding_response *response)
- int32_t **bblib_precoding_c** (const struct bblib_precoding_request *request, struct bblib_precoding_response *response)

7.41.1 Detailed Description

Source code of External API for precoding functions.

7.41.2 Function Documentation

7.41.2.1 bblib_lte_precoding_version()

```
int16_t bblib_lte_precoding_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_lte_precoding library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.41.2.2 bblib_precoding()

```
int32_t bblib_precoding (
            const struct bblib_precoding_request * request,
            struct bblib_precoding_response * response )
```

This function implements precoding procedure.

Parameters

in	*request*	Structure with request data.
out	*response*	Structure containing the output data.

Note

Data_in and data_out need to be aligned to 256 bits.

Returns

0 for success, -1 for error.

7.42 phy_precoding_5gnr.h File Reference

Data Structures

- struct bblib_precoding_5gnr_precoding
- struct bblib_precoding_5gnr_layers
- struct bblib_precoding_5gnr_antennas
- struct bblib_precoding_5gnr_request
- struct bblib_precoding_5gnr_response
- struct bblib_mul_beta_request
- struct bblib_mul_beta_response
- struct bblib_precoding_codebook_fetch_5gnr_request
- struct bblib_precoding_codebook_fetch_5gnr_response

Enumerations

- enum precoding_mode { BBLIB_MIXED_MODE, BBLIB_FULL_FXP_16B_ACC_MODE, BBLIB_FLP_MODE }
- enum bblib_precoding_codebook_config { MAX_N1_NUMBER = 3, MAX_I13_NUMBER = 4, MAX_PRECODING_ANTENNA = 4 }
- enum bblib_precoding_cb_type { **BBLIB_CB_TYPE1_SP** = 0, **BBLIB_CB_TYPE1_MP** = 1, **BBLIB_CB_TYPE2** = 2, **BBLIB_CB_TYPE2_PS** = 3 }
- enum bblib_precoding_cb_mode { **BBLIB_CB_MODE_1** = 1, **BBLIB_CB_MODE_2** = 2 }
- enum bblib_precoding_trans_scheme { **BBLIB_STATIC_PRECODER** = 0, **BBLIB_LAYER_MAPPING_AND_PRECODING** = 1 }

Functions

- int16_t bblib_precoding_5gnr_version (char *version, int buffer_size)

- void bblib_precoding_5gnr (const struct bblib_precoding_5gnr_request *request, struct bblib_precoding_5gnr_response *response)
- void **bblib_precoding_5gnr_c** (const struct bblib_precoding_5gnr_request *request, struct bblib_precoding_5gnr_response *response)
- void **bblib_precoding_5gnr_avx2** (const struct bblib_precoding_5gnr_request *request, struct bblib_precoding_5gnr_response *response)
- void **bblib_precoding_5gnr_avx512** (const struct bblib_precoding_5gnr_request *request, struct bblib_precoding_5gnr_response *response)

- void bblib_precoding_5gnr_fxp (const struct bblib_precoding_5gnr_request *request, struct bblib_precoding_5gnr_response *response)
- void **bblib_precoding_5gnr_fxp_c** (const struct bblib_precoding_5gnr_request *request, struct bblib_precoding_5gnr_response *response)
- void **bblib_precoding_5gnr_fxp_avx2** (const struct bblib_precoding_5gnr_request *request, struct bblib_precoding_5gnr_response *response)
- void **bblib_precoding_5gnr_fxp_avx512** (const struct bblib_precoding_5gnr_request *request, struct bblib_precoding_5gnr_response *response)

- int bblib_mul_beta_fxp (const struct bblib_mul_beta_request *request, struct bblib_mul_beta_response *response)
- int **bblib_mul_beta_fxp_avx512** (const struct bblib_mul_beta_request *request, struct bblib_mul_beta_response *response)

- int bblib_precoding_codebook_fetch_5gnr (struct bblib_precoding_codebook_fetch_5gnr_request *request, struct bblib_precoding_codebook_fetch_5gnr_response *response)

7.42.1 Detailed Description

External API for 5G NR Precoder.

Precoding combines the information from beamforming with several layers of sub-carriers to create a set of antenna outputs. The operation is essentially the multiplication of a complex precoding matrix by a sub-carrier matrix containing layers to form a subcarrier matrix containing antenna outputs.

Precoding receives precode matrix and layers inputs from two different kernels so the parameters are passed in separately. The input subcarriers and the output antenna data will be stored in non-contiguous memory because the predecessor and successor kernels generate their data in different memory locations. Extra kernels could be inserted before and after precoding to combine and split the data appropriately but it is more efficient to roll this operation into precoding.

7.42.2 Enumeration Type Documentation

7.42.2.1 bblib_precoding_cb_mode

```
enum bblib_precoding_cb_mode
```

enum containing the different DL codebook modes

Warning

EXPERIMENTAL: Further optimization is possible, API may change without prior notice.

7.42.2.2 bblib_precoding_cb_type

```
enum bblib_precoding_cb_type
```

enum containing the different DL codebook types \0: type 1 single panel \1: type 1 multi panel \2: type 2 \3: type 2 port selection

Warning

EXPERIMENTAL: Further optimization is possible, API may change without prior notice.

7.42.2.3 bblib_precoding_codebook_config

```
enum bblib_precoding_codebook_config
```

Configuration supported by the code book precoding APIs including max number of Output Antenna.

Warning

EXPERIMENTAL: Further optimization is possible, API may change without prior notice.

Enumerator

MAX_N1_NUMBER	Maximum number of n1 according to 38.214 5.2.2.2
MAX_I13_NUMBER	Maximum number of i1,3 according to 38.214 5.2.2.2
MAX_PRECODING_ANTENNA	Maximum number of antennas supported

7.42.2.4 bblib_precoding_trans_scheme

```
enum bblib_precoding_trans_scheme
```

enum containing the different 3GPP transmission schemes supported by the code book precoding and layer mapping function

Warning

EXPERIMENTAL: Further optimization is possible, API may change without prior notice.

7.42.2.5 precoding_mode

```
enum precoding_mode
```

precoding mode: mixed mode (calculations using flp),full fxp mode or flp mode

Enumerator

BBLIB_MIXED_MODE	Mixed mode: fixed point input and output with floating point calculations. Best fixed point accuracy, but not as fast as BBLIB_FULL_FXP_16B_ACC_MODE. Suitable for codebook and non-codebook precoding.
BBLIB_FULL_FXP_16B_ACC_MODE	Full fixed point mode: fixed point input and output with fixed point calculations. Faster than BBLIB_MIXED_MODE but less accurate due to the use of a 16 bit accumulation. Suitable for codebook precoding.
BBLIB_FLP_MODE	Floating point mode: half-precision layer data, all other data single precision. Suitable for codebook and non-codebook precoding.

7.42.3 Function Documentation

7.42.3.1 bblib_mul_beta_fxp()

```
int bblib_mul_beta_fxp (
            const struct bblib_mul_beta_request * request,
            struct bblib_mul_beta_response * response )
```

Fixed point multiply with beta.

Parameters

in	*request*	Structure defining the input data sequence, length and beta.
out	*response*	Structure defining the output data sequence.

Returns

beta multiplication result, return 0 is success,return -1 is fail.

Note

the maximum input beta should not exceed 4.

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.42.3.2 bblib_precoding_5gnr()

```
void bblib_precoding_5gnr (
            const struct bblib_precoding_5gnr_request * request,
            struct bblib_precoding_5gnr_response * response )
```

Precoding 5gnr. Implements precoding procedure for floating point mode (precoding_mode) BBLIB_FLP_MODE: layer data is half-precision and all other data is single precision floating point numbers.

Parameters

in	*request*	Structure defining the precoding matrix and layers data.
out	*response*	Structure defining the antenna data.

7.42.3.3 bblib_precoding_5gnr_fxp()

```
void bblib_precoding_5gnr_fxp (
            const struct bblib_precoding_5gnr_request * request,
            struct bblib_precoding_5gnr_response * response )
```

Fixed point precoding 5gnr. Implements precoding procedure for fixed point numbers in two modes (precoding_↩ mode)

- BBLIB_MIXED_MODE: fixed point input and output data with floating point calculations;
- BBLIB_FULL_FXP_16B_ACC_MODE: fixed point input and output with fixed point calculations using 16 bit accumulation (less accuracy).

Parameters

in	*request*	Structure defining the precoding matrix and layers data.
out	*response*	Structure defining the antenna data.

Note

In full fxp mode an overflow may occur while adding fxp numbers (16b accumulators are used in format Q16s9), there is no guard to prevent it. The 16b accumulators are used deliberately to increase performance.

7.42.3.4 bblib_precoding_5gnr_version()

```
int16_t bblib_precoding_5gnr_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_precoding_5gnr library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, typically no more than BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.42.3.5 bblib_precoding_codebook_fetch_5gnr()

```
int bblib_precoding_codebook_fetch_5gnr (
            struct bblib_precoding_codebook_fetch_5gnr_request * request,
            struct bblib_precoding_codebook_fetch_5gnr_response * response )
```

fetch the 3GPP 38.214 pre-defined precoding matrix

Parameters

in	*request*	structure defining input
out	*response*	structure defining outputs

Returns

zero on success, negative on error

Warning

EXPERIMENTAL: Further optimization is possible, API may change without prior notice.

7.43 phy_pucch_5gnr.h File Reference

Data Structures

- struct bblib_pucch_seq_gen_request
- struct bblib_pucch_seq_gen_response
- struct bblib_pucch_low_papr_request
- struct bblib_pucch_low_papr_response
- struct bblib_pucch_f0_detect_request
- struct bblib_pucch_f0_detect_response
- struct bblib_phy_pucch_f1_mul_omega_request
- struct bblib_phy_pucch_f1_mul_omega_response
- struct bblib_phy_pucch_f1_mul_payload_request
- struct bblib_phy_pucch_f1_mul_payload_response
- struct bblib_despread_compensate_pucch_f1_request
- struct bblib_despread_compensate_pucch_f1_response

Enumerations

- enum bblib_pucch_5gnr_seq_gen_const { BBLIB_NUM_OF_SUBFRAME_IN_ONE_FRAME = 10, BBLIB_NUM_OF_SLOT_PER_SUBFRAME = 8, BBLIB_NUM_OF_SYMBOL_PER_SLOT = 14 }
- enum bblib_pucch_5gnr_constants { BBLIB_MAX_PUCCH_F0_RX_AP_NUM = 4, BBLIB_PUCCH_SYMB_PER_SLOT = 14, BBLIB_PUCCH_SLOTS_PER_SF = 2 }
- enum bblib_pucch_fullband_sc { FULLBAND_SC_792 = 792 }

Functions

- int16_t bblib_pucch_5gnr_version (char ∗version, int buffer_size)
- void bblib_init_omega_pucch_f1 (void)
- int bblib_pucch_rnd_seq_gen (const struct bblib_pucch_seq_gen_request ∗request, struct bblib_pucch_seq_gen_response ∗response)
- int **bblib_pucch_rnd_seq_gen_avx2** (const struct bblib_pucch_seq_gen_request ∗request, struct bblib_pucch_seq_gen_response ∗response)
- int **bblib_pucch_rnd_seq_gen_avx512** (const struct bblib_pucch_seq_gen_request ∗request, struct bblib_pucch_seq_gen_response ∗response)
- int bblib_pucch_low_papr_seq_gen_5gnr (const struct bblib_pucch_low_papr_request ∗request, struct bblib_pucch_low_papr_response ∗response)
- int bblib_pucch_f0_detect_5gnr (const struct bblib_pucch_f0_detect_request ∗request, struct bblib_pucch_f0_detect_response ∗response)

- int **bblib_pucch_f0_detect_5gnr_avx2** (const struct bblib_pucch_f0_detect_request *request, struct bblib_pucch_f0_detect_response *response)
- int **bblib_pucch_f0_detect_5gnr_avx512** (const struct bblib_pucch_f0_detect_request *request, struct bblib_pucch_f0_detect_response *response)
- int **bblib_pucch_f0_detect_5gnr_cfp** (const struct bblib_pucch_f0_detect_request *request, struct bblib_pucch_f0_detect_response *response)

- void bblib_phy_pucch_f1_mul_omega (const struct bblib_phy_pucch_f1_mul_omega_request *request, struct bblib_phy_pucch_f1_mul_omega_response *response)
- void **bblib_phy_pucch_f1_mul_omega_avx2** (const struct bblib_phy_pucch_f1_mul_omega_request *request, struct bblib_phy_pucch_f1_mul_omega_response *response)
- void **bblib_phy_pucch_f1_mul_omega_avx512** (const struct bblib_phy_pucch_f1_mul_omega_request *request, struct bblib_phy_pucch_f1_mul_omega_response *response)

- void bblib_phy_pucch_f1_mul_payload (const struct bblib_phy_pucch_f1_mul_payload_request *request, struct bblib_phy_pucch_f1_mul_payload_response *response)
- void **bblib_phy_pucch_f1_mul_payload_avx2** (const struct bblib_phy_pucch_f1_mul_payload_request *request, struct bblib_phy_pucch_f1_mul_payload_response *response)
- void **bblib_phy_pucch_f1_mul_payload_avx512** (const struct bblib_phy_pucch_f1_mul_payload_request *request, struct bblib_phy_pucch_f1_mul_payload_response *response)

- void bblib_despread_compensate_pucch_f1 (const struct bblib_despread_compensate_pucch_f1_request *request, struct bblib_despread_compensate_pucch_f1_response *response)
- void **bblib_despread_compensate_pucch_f1_avx2** (const struct bblib_despread_compensate_pucch_f1_request *request, struct bblib_despread_compensate_pucch_f1_response *response)
- void **bblib_despread_compensate_pucch_f1_avx512** (const struct bblib_despread_compensate_pucch_f1_request *request, struct bblib_despread_compensate_pucch_f1_response *response)

7.43.1 Detailed Description

Top level API for 5gNR pucch functions.

Overview: This library contains a collection of common functions used to implement 5gnr pucch as per 38.211.

The main functionality includes pucch format 0 and 1 detection algorithm.

7.43.2 Enumeration Type Documentation

7.43.2.1 bblib_pucch_5gnr_constants

```
enum bblib_pucch_5gnr_constants
```

This configuration sets global constants and macros which are of general use throughout the module.

Enumerator

BBLIB_MAX_PUCCH_F0_RX_AP_NUM	Maximum number of RX antennas supported
BBLIB_PUCCH_SYMB_PER_SLOT	Symbols per slot
BBLIB_PUCCH_SLOTS_PER_SF	Slots per subframe

7.43.2.2 bblib_pucch_5gnr_seq_gen_const

```
enum bblib_pucch_5gnr_seq_gen_const
```

This configuration sets global constants and macros which are of general use throughout 5gnr sequence generation module.

Enumerator

BBLIB_NUM_OF_SUBFRAME_IN_ONE_FRAME	Number of subframes in one frame
BBLIB_NUM_OF_SLOT_PER_SUBFRAME	Number of slots in one subframe
BBLIB_NUM_OF_SYMBOL_PER_SLOT	Number of symbols in one slot

7.43.2.3 bblib_pucch_fullband_sc

```
enum bblib_pucch_fullband_sc
```

Enum containing the supported subcarrier size.

Enumerator

FULLBAND_SC_792	792 max supported subcarriers

7.43.3 Function Documentation

7.43.3.1 bblib_despread_compensate_pucch_f1()

```
void bblib_despread_compensate_pucch_f1 (
            const struct bblib_despread_compensate_pucch_f1_request * request,
            struct bblib_despread_compensate_pucch_f1_response * response )
```

Function for PUCCH format 1 despreading and compensation.

Parameters

in	*request*	Pointer to the request structure containing the input settings.
out	*response*	Pointer to the response structure containing the result.

Returns

none

7.43.3.2 bblib_init_omega_pucch_f1()

```
void bblib_init_omega_pucch_f1 (
            void )
```

Function init Omega in PUCCH F1.

Returns

none

7.43.3.3 bblib_phy_pucch_f1_mul_omega()

```
void bblib_phy_pucch_f1_mul_omega (
            const struct bblib_phy_pucch_f1_mul_omega_request * request,
            struct bblib_phy_pucch_f1_mul_omega_response * response )
```

Function multiply DMRS and UCI symbols with Omega sequence.

Parameters

in	*request*	Pointer to the request structure containing the input settings.
out	*response*	Pointer to the response structure containing the result.

Returns

none

7.43.3.4 bblib_phy_pucch_f1_mul_payload()

```
void bblib_phy_pucch_f1_mul_payload (
            const struct bblib_phy_pucch_f1_mul_payload_request * request,
            struct bblib_phy_pucch_f1_mul_payload_response * response )
```

Function multiply DMRS and UCI symbols with Omega sequence.

Parameters

in	*request*	Pointer to the request structure containing the input settings.
out	*response*	Pointer to the response structure containing the result.

Returns

none

7.43.3.5 bblib_pucch_5gnr_version()

```
int16_t bblib_pucch_5gnr_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_pucch_5gnr library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, length BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

Note

All Kernels shall contain a version function.

7.43.3.6 bblib_pucch_f0_detect_5gnr()

```
int bblib_pucch_f0_detect_5gnr (
            const struct bblib_pucch_f0_detect_request * request,
            struct bblib_pucch_f0_detect_response * response )
```

Function detect format 0 references in an input buffer. The received signal after FFT / RE demapping is correlated with local low PAPR sequence. ACK/SR is detected by selecting maximum correlation value.

Parameters

in	*request*	Pointer to the request structure containing the input settings.
out	*response*	Pointer to the response structure containing the result.

Returns

int: status 0 on success, non 0 on error

7.43.3.7 bblib_pucch_low_papr_seq_gen_5gnr()

```
int bblib_pucch_low_papr_seq_gen_5gnr (
            const struct bblib_pucch_low_papr_request * request,
            struct bblib_pucch_low_papr_response * response )
```

Function to generate the low papr sequence based on 38.211 5.2.2 required by pucch as defined in 6.3.2.2.

Parameters

in	*request*	Pointer to the request structure containing the input settings.
out	*response*	Pointer to the response structure containing the result.

Returns

int: status 0 on success, non 0 on error

7.43.3.8 bblib_pucch_rnd_seq_gen()

```
int bblib_pucch_rnd_seq_gen (
            const struct bblib_pucch_seq_gen_request * request,
            struct bblib_pucch_seq_gen_response * response )
```

Function to generate the 3 Random sequences based on 38.211 5.2.1 required by pucch as defined in 6.3.2.2.

Parameters

in	*request*	Pointer to the request structure containing the input settings.
out	*response*	Pointer to the response structure containing the result.

Returns

int: status 0 on success, non 0 on error

7.44 phy_pucch_cestimate_5gnr.h File Reference

Data Structures

- struct bblib_pucch_cestimate_5gnr_request
- struct bblib_pucch_cestimate_5gnr_response
- struct bblib_pucch_cestimate_5gnr_dmrs_request
- struct bblib_pucch_cestimate_5gnr_dmrs_response

Enumerations

- enum cestimate_pucch_5gnr_constants {
  BBLIB_MAX_PUCCH_RX_AP_NUM = 4, BBLIB_PUCCH_F2_DMRS_PER_RB = 4, BBLIB_PUCCH_F3_DMRS_PER_RB = 12, BBLIB_PUCCH_F4_DMRS_PER_RB = 12,
  BBLIB_PUCCH_DMRS_SYM4_HOPPING = 2 }
- enum cestimate_pucch_5gnr_phy_formats { BBLIB_PUCCH_FORMAT_2 = 2, BBLIB_PUCCH_FORMAT_3 = 3, BBLIB_PUCCH_FORMAT_4 = 4 }

Functions

- int16_t bblib_pucch_cestimate_5gnr_version (char ∗version, int buffer_size)

- int32_t bblib_pucch_cestimate_5gnr (const struct bblib_pucch_cestimate_5gnr_request ∗request, struct bblib_pucch_cestimate_5gnr_response ∗response)
- int32_t **bblib_pucch_cestimate_5gnr_c** (const struct bblib_pucch_cestimate_5gnr_request ∗request, struct bblib_pucch_cestimate_5gnr_response ∗response)
- int32_t **bblib_pucch_cestimate_5gnr_avx2** (const struct bblib_pucch_cestimate_5gnr_request ∗request, struct bblib_pucch_cestimate_5gnr_response ∗response)
- int32_t **bblib_pucch_cestimate_5gnr_avx512** (const struct bblib_pucch_cestimate_5gnr_request ∗request, struct bblib_pucch_cestimate_5gnr_response ∗response)

- int32_t bblib_pucch_cestimate_5gnr_dmrs (const struct bblib_pucch_cestimate_5gnr_dmrs_request ∗request, struct bblib_pucch_cestimate_5gnr_dmrs_response ∗response)
- int32_t **bblib_pucch_cestimate_5gnr_dmrs_c** (const struct bblib_pucch_cestimate_5gnr_dmrs_request ∗request, struct bblib_pucch_cestimate_5gnr_dmrs_response ∗response)
- int32_t **bblib_pucch_cestimate_5gnr_dmrs_avx2** (const struct bblib_pucch_cestimate_5gnr_dmrs_request ∗request, struct bblib_pucch_cestimate_5gnr_dmrs_response ∗response)
- int32_t **bblib_pucch_cestimate_5gnr_dmrs_avx512** (const struct bblib_pucch_cestimate_5gnr_dmrs_request ∗request, struct bblib_pucch_cestimate_5gnr_dmrs_response ∗response)

7.44.1 Detailed Description

External API for PUCCH 5GNR Channel Estimator.

7.44.2 Enumeration Type Documentation

7.44.2.1 cestimate_pucch_5gnr_constants

`enum cestimate_pucch_5gnr_constants`

This configuration sets global constants and macros which are of general use throughout the module.

Enumerator

BBLIB_MAX_PUCCH_RX_AP_NUM	Maximum number of RX antennas supported
BBLIB_PUCCH_F2_DMRS_PER_RB	DMRS per PRB, format 2
BBLIB_PUCCH_F3_DMRS_PER_RB	DMRS per PRB, format 3
BBLIB_PUCCH_F4_DMRS_PER_RB	DMRS per PRB, format 4
BBLIB_PUCCH_DMRS_SYM4_HOPPING	DMRS per PRB for hopping enable & symbol=4 >

7.44.2.2 cestimate_pucch_5gnr_phy_formats

```
enum cestimate_pucch_5gnr_phy_formats
```

PUCCH formats.

Enumerator

BBLIB_PUCCH_FORMAT_2	PUCCH format2
BBLIB_PUCCH_FORMAT_3	PUCCH format3
BBLIB_PUCCH_FORMAT_4	PUCCH format4

7.44.3 Function Documentation

7.44.3.1 bblib_pucch_cestimate_5gnr()

```
int32_t bblib_pucch_cestimate_5gnr (
            const struct bblib_pucch_cestimate_5gnr_request * request,
            struct bblib_pucch_cestimate_5gnr_response * response )
```

5GNR PUCCH Channel Estimator

Parameters

in	*request*	structure
out	*response*	structure

Returns

0 for success, -1 for error.

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.44.3.2 bblib_pucch_cestimate_5gnr_dmrs()

```
int32_t bblib_pucch_cestimate_5gnr_dmrs (
            const struct bblib_pucch_cestimate_5gnr_dmrs_request * request,
            struct bblib_pucch_cestimate_5gnr_dmrs_response * response )
```

5GNR PUCCH Channel Estimator DMRS generator

Parameters

in	*request*	structure
out	*response*	structure

Returns

0 for success, -1 for error.

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.44.3.3 bblib_pucch_cestimate_5gnr_version()

```
int16_t bblib_pucch_cestimate_5gnr_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_pucch_cestimate_5gnr library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, typically no more than BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.45 phy_pucch_equ_5gnr.h File Reference

Data Structures

- struct bblib_pucch_equ_5gnr_request
- struct bblib_pucch_equ_5gnr_response

Enumerations

- enum pucch_equ_config { **MAX_N_ANT** = 4, **EQU_SCALE** = 13 }
- enum equ_pucch_5gnr_phy_formats { BBLIB_EQU_PUCCH_FORMAT_2 = 2, BBLIB_EQU_PUCCH_FORMAT_3 = 3, BBLIB_EQU_PUCCH_FORMAT_4 = 4 }

Functions

- int16_t bblib_pucch_equ_5gnr_version (char *version, int buffer_size)

- int32_t bblib_pucch_equ_5gnr (const struct bblib_pucch_equ_5gnr_request *request, struct bblib_pucch_equ_5gnr_response *response)
- int32_t **bblib_pucch_equ_5gnr_c** (const struct bblib_pucch_equ_5gnr_request *request, struct bblib_pucch_equ_5gnr_response *response)
- int32_t **bblib_pucch_equ_5gnr_avx2** (const struct bblib_pucch_equ_5gnr_request *request, struct bblib_pucch_equ_5gnr_response *response)
- int32_t **bblib_pucch_equ_5gnr_avx512** (const struct bblib_pucch_equ_5gnr_request *request, struct bblib_pucch_equ_5gnr_response *response)

7.45.1 Detailed Description

API for PUCCH Equalization MIMO based on MRC for 5GNR Library.

Overview: The purpose of this kernel is to implement MMSE Equalization for PUCCH for 5GNR. Based on the 38.211 3GPP specification, section 6.3.2.5.

Algorithm Guidance:

1. Multiply complex conjugate of channel estimate result with rx antenna signal for each antenna.
2. Multiply channel estimate result with complex conjugate to get absolute value for each antenna.
3. Invert result of previous step, accumulated across all antenna.
4. Multiply inverted absolute channel estimate result with the result of step one for each antenna.
5. Accumulate each antenna value from step 4 and store result.

7.45.2 Enumeration Type Documentation

7.45.2.1 equ_pucch_5gnr_phy_formats

```
enum equ_pucch_5gnr_phy_formats
```

PUCCH formats.

Enumerator

BBLIB_EQU_PUCCH_FORMAT↩ _2	PUCCH format2
BBLIB_EQU_PUCCH_FORMAT↩ _3	PUCCH format3
BBLIB_EQU_PUCCH_FORMAT↩ _4	PUCCH format4

7.45.3 Function Documentation

7.45.3.1 bblib_pucch_equ_5gnr()

```
int32_t bblib_pucch_equ_5gnr (
            const struct bblib_pucch_equ_5gnr_request * request,
            struct bblib_pucch_equ_5gnr_response * response )
```

Determines the estimated signal based on the input channel state for PUCCH.

Parameters

in	*request*	Pointer to the request structure containing the input settings.
out	*response*	Pointer to the response structure containing the result.

Returns

int: status 0 on success, non 0 on error

7.45.3.2 bblib_pucch_equ_5gnr_version()

```
int16_t bblib_pucch_equ_5gnr_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_equ_pucch_5gnr library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, length BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.46 phy_pucch_focompensation_5gnr.h File Reference

Data Structures

- struct bblib_pucch_fo_compensation_5gnr_request
- struct bblib_pucch_fo_compensation_5gnr_response

Functions

- void bblib_print_pucch_focompensation_5gnr_version ()
- int16_t bblib_pucch_focompensation_5gnr_version (char ∗version, int buffer_size)

- void bblib_pucch_focompensation_5gnr (struct bblib_pucch_fo_compensation_5gnr_request ∗request, struct bblib_pucch_fo_compensation_5gnr_response ∗response)
- void **bblib_pucch_focompensation_5gnr_avx512** (struct bblib_pucch_fo_compensation_5gnr_request ∗p_ce_req, struct bblib_pucch_fo_compensation_5gnr_response ∗p_ce_resp)

7.46.1 Detailed Description

External API for channel estiation and timing advance estimation for 5GNR.

Overview:

The pucch_focompensation_5gnr kernel is a 5G NR PUCCH Format 1 and 3 Frequency offset compensation. This is a generic kernel which can be used to both PUCCH Format 1 and 3.

Algorithm Guidance:

The 5G NR Frequency offset compensation algorithm can be broken down into the following steps:
1.Using estimation offset and stored compensation table to calculating the compensation value of each SC.
2.In order to get final compensated reviced data, Need to multiplex those compensation values with recived original data.

7.46.2 Function Documentation

7.46.2.1 bblib_print_pucch_focompensation_5gnr_version()

```
void bblib_print_pucch_focompensation_5gnr_version ( )
```

printf pucch focompensation 5gnr version

Returns

null.

7.46.2.2 bblib_pucch_focompensation_5gnr()

```
void bblib_pucch_focompensation_5gnr (
            struct bblib_pucch_fo_compensation_5gnr_request * request,
            struct bblib_pucch_fo_compensation_5gnr_response * response )
```

pucch_focompensation_5gnr procedures.

Parameters

in	*request*	Structure containing the input data which need to be 64 bytes alignment.
out	*response*	Structure containing the compensated output data which need 64 byte alignment.

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.46.2.3 bblib_pucch_focompensation_5gnr_version()

```
int16_t bblib_pucch_focompensation_5gnr_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_pucch_focompensation_5gnr library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.47 phy_pusch_foestimate_5gnr.h File Reference

Data Structures

- struct bblib_pusch_fo_estimation_5gnr_request
- struct bblib_pusch_fo_estimation_5gnr_response

Functions

- void bblib_print_pusch_foestimate_5gnr_version ()
- int16_t bblib_pusch_foestimate_5gnr_version (char ∗version, int buffer_size)

- void bblib_pusch_fostimate_5gnr (struct bblib_pusch_fo_estimation_5gnr_request ∗request, struct bblib_pusch_fo_estimation_5gnr_response ∗response)
- void **bblib_pusch_foestimate_5gnr_avx512** (struct bblib_pusch_fo_estimation_5gnr_request ∗p_ce_req, struct bblib_pusch_fo_estimation_5gnr_response ∗p_ce_resp)

7.47.1 Detailed Description

External API for channel estiation and timing advance estimation for 5GNR.

External API for carrier frequency offset estimation for 5GNR.

Overview:

The lib_cestimate_5gnr kernel is a 5G NR channel estimate functions with Wiener filter. This is a generic kernel which can be used to channel estimate in both UL and DL.

Algorithm Guidance:

The 5G NR channel estimator algorithm (Wiener) can be broken down into the following steps:
1.Estimate reference signal channel using Least Square algorithm (per layer per antenna).
2.Generate interpolation weight and do the interpolation in frequency domain for all subcarriers (per layer per antenna).
3.Estimate noise power (per layer per antenna).
4.Two-iteration channel estimation. Repeat step 2 and step 3 (per antenna).
Overview:

The lib_pusch_foestimation_5gnr kernel is a 5G NR carrier frequency offset estimate functions. This is a generic kernel which can be used to carrier frequency offset estimation in UL.

Algorithm Guidance:

The 5G NR carrier frequency offset estimator algorithm (Wiener) can be broken down into the following steps:
1.Estimate reference signal channel using Least Square algorithm (per layer per antenna).
2.Estimate carrier frequency offset (per antenna).

7.47.2 Function Documentation

7.47.2.1 bblib_print_pusch_foestimate_5gnr_version()

```
void bblib_print_pusch_foestimate_5gnr_version ( )
```

printf cestimate 5gnr version

Returns

null.

7.47.2.2 bblib_pusch_foestimate_5gnr_version()

```
int16_t bblib_pusch_foestimate_5gnr_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_pusch_foestimation_5gnr library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.47.2.3 bblib_pusch_fostimate_5gnr()

```
void bblib_pusch_fostimate_5gnr (
            struct bblib_pusch_fo_estimation_5gnr_request * request,
            struct bblib_pusch_fo_estimation_5gnr_response * response )
```

cestimate_5gnr procedures.

Parameters

in	*request*	Structure containing the input data which need to be 64 bytes alignment.
out	*response*	Structure containing the decoding output data which need 64 byte alignment.

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.48 phy_pusch_irc_symbol_processing_5gnr.h File Reference

Data Structures

- struct bblib_pusch_irc_symbol_processing_request
- struct bblib_pusch_irc_symbol_processing_response

Functions

- int16_t bblib_pusch_irc_symbol_processing_version (char *version, int buffer_size)
- int32_t bblib_pusch_irc_symbol_processing (bblib_pusch_irc_symbol_processing_request *request, bblib_pusch_irc_symbol_processing_response *response)
- int32_t **bblib_pusch_irc_symbol_processing_avx512** (bblib_pusch_irc_symbol_processing_request *request, bblib_pusch_irc_symbol_processing_response *response)

7.48.1 Detailed Description

External API for 5GNR PUSCH symbol processing.

Overview: This module implements MMSE IRC MIMO equalization, layer demapping,and LLR dempping. MMSE IRC MIMO equalization supports 1x2, 2x2, 2x4 1x16 2x16 in SU case; 4x16 in 2UE MU case; 8x16 in 4UE MU case. LLR demapping supports QPSK, 16QAM, 64QAM and 256QAM

Algorithm Guidance:

1. MMSE IRC MIMO equalization refers to lib_equalization
2. Layer demapping refers to lib_layerdemapping_5gnr
3. LLR demapping refers to the inline comments

7.48.2 Function Documentation

7.48.2.1 bblib_pusch_irc_symbol_processing()

```
int32_t bblib_pusch_irc_symbol_processing (
            bblib_pusch_irc_symbol_processing_request * request,
            bblib_pusch_irc_symbol_processing_response * response )
```

5GNR PUSCH IRC symbol processing: MMSE IRC MIMO+Layer demapping+LLR demapping

Parameters

in	*request*	Input request structure for PUSCH symbol processing
out	*response*	Output response structure for PUSCH symbol processing

Returns

0 for success, and -1 for error

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.48.2.2 bblib_pusch_irc_symbol_processing_version()

```
int16_t bblib_pusch_irc_symbol_processing_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_pusch_irc_symbol_processing library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.49 phy_pusch_symbol_processing_5gnr.h File Reference

Data Structures

- struct bblib_pusch_symbol_processing_request
- struct bblib_pusch_symbol_processing_response
- struct bblib_layer_llr_demap_request
- struct bblib_layer_llr_demap_response

Functions

- int16_t bblib_pusch_symbol_processing_version (char *version, int buffer_size)

- int32_t bblib_pusch_symbol_processing (bblib_pusch_symbol_processing_request *request, bblib_pusch_symbol_processing_
  *response)
- int32_t **bblib_pusch_symbol_processing_avx512** (bblib_pusch_symbol_processing_request *request, bblib_pusch_symbol_processing_response *response)

- int32_t bblib_layer_llr_demap_processing (bblib_layer_llr_demap_request *request, bblib_layer_llr_demap_response *response)
- int32_t **bblib_layer_llr_demap_processing_avx512** (bblib_layer_llr_demap_request *request, bblib_layer_llr_demap_respon
  *response)

7.49.1 Detailed Description

External API for 5GNR PUSCH symbol processing.

Overview: This module implements MMSE MIMO equalization, layer demapping,and LLR dempping. MMSE MIMO equalization supports 1x2, 1x4, 2x2 and 2x4; and 4x4 in 2UE MU case LLR demapping supports pi/2 BPSK, QPSK, 16QAM and 64QAM

Algorithm Guidance:

1. MMSE MIMO equalization refers to lib_equalization
2. Layer demapping refers to lib_layerdemapping_5gnr
3. LLR demapping refers to the inline comments

7.49.2 Function Documentation

7.49.2.1 bblib_layer_llr_demap_processing()

```
int32_t bblib_layer_llr_demap_processing (
            bblib_layer_llr_demap_request * request,
            bblib_layer_llr_demap_response * response )
```

5GNR PUSCH layer demap and LLR demap processing: Layer demapping+LLR demapping

Parameters

in	*request*	Input request structure for Layer demapping and LLR demapping
out	*response*	Output response structure for Layer demapping and LLR demapping

Returns

0 for success, and -1 for error

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.49.2.2 bblib_pusch_symbol_processing()

```
int32_t bblib_pusch_symbol_processing (
            bblib_pusch_symbol_processing_request * request,
            bblib_pusch_symbol_processing_response * response )
```

5GNR PUSCH symbol processing: MMSE MIMO+TA Compensation+Layer demapping+LLR demapping

Parameters

in	*request*	Input request structure for PUSCH symbol processing
out	*response*	Output response structure for PUSCH symbol processing

Returns

0 for success, and -1 for error

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.49.2.3 bblib_pusch_symbol_processing_version()

```
int16_t bblib_pusch_symbol_processing_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_pusch_symbol_processing library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.50 phy_pusch_symbol_processing_5gnr_avx512.h File Reference

7.50.1 Detailed Description

Macros used for MMSE MIMO detection, with post SINR calculation.

7.51 phy_qr_decomposition_5gnr.h File Reference

Data Structures

- struct bblib_qr_decomp_request
- struct bblib_qr_decomp_response

Functions

- int16_t bblib_qr_decomp_version (char ∗version, int buffer_size)

- void bblib_qr_decomposition (const struct bblib_qr_decomp_request ∗request, struct bblib_qr_decomp_response ∗response)
- void **bblib_qr_decomposition_c** (const struct bblib_qr_decomp_request ∗request, struct bblib_qr_decomp_response ∗response)
- void **bblib_qr_decomposition_avx2** (const struct bblib_qr_decomp_request ∗request, struct bblib_qr_decomp_response ∗response)
- void **bblib_qr_decomposition_avx512** (const struct bblib_qr_decomp_request ∗request, struct bblib_qr_decomp_response ∗response)

7.51.1 Detailed Description

External API for QR Decomposition.

7.51.2 Function Documentation

7.51.2.1 bblib_qr_decomp_version()

```
int16_t bblib_qr_decomp_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_qr_decomposition_operation library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, typically no more than BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.51.2.2 bblib_qr_decomposition()

```
void bblib_qr_decomposition (
            const struct bblib_qr_decomp_request * request,
            struct bblib_qr_decomp_response * response )
```

Decompose a Nx2 matrix into separate Nx2 and 2x2 Q/R matrices using Modified Gram-Schmidt.

Parameters

in	*request*	Pointer to request structure containing the matrix to be decomposed.
in,out	*response*	Pointer to a response structure that will contain the Q and R decomposition results.

7.52 phy_rate_dematching_5gnr.h File Reference

Data Structures

- struct bblib_rate_dematching_5gnr_request
- struct bblib_rate_dematching_5gnr_response

Functions

- void bblib_rate_dematching_5gnr (struct bblib_rate_dematching_5gnr_request *request, struct bblib_rate_dematching_5gnr_re *response)
- void **bblib_rate_dematching_5gnr_c** (struct bblib_rate_dematching_5gnr_request *req, struct bblib_rate_dematching_5gnr_r *resp)
- void **bblib_rate_dematching_5gnr_avx512** (struct bblib_rate_dematching_5gnr_request *req, struct bblib_rate_dematching_5gnr_response *resp)
- int16_t bblib_rate_dematching_5gnr_version (char *version, int buffer_size)
- void bblib_print_rate_dematching_5gnr_version (void)

7.52.1 Detailed Description

External API for 5G rate dematching.

5G rate dematching module performs HARQ combine, De-Interleaver and De-Selection for data channel LDPC decoder.

The lib_rate_dematching_5gnr kernel is a 5G NR Rate Dematching for LDPC code implemented by Harq Combine, bit de-interleaving and bit de-selection. For reducing the interference and matching the bearing of physical channel, the transfer site need to do bit collection, bit interleaving and bit repeat or puncture some bits in order to map to physical resource element. Vice-versa happens in receiver site to do bit de-interleaving, bit de-selection and HARQ combine for retransmission.

It is implemented according to 3GPP TS 38.212 5.4.2 Rate matching for LDPC code The test coverage for this module includes BPSK, QPSK, 16QAM, 64QAM and 256QAM

7.52.2 Function Documentation

7.52.2.1 bblib_rate_dematching_5gnr()

```
void bblib_rate_dematching_5gnr (
            struct bblib_rate_dematching_5gnr_request * request,
            struct bblib_rate_dematching_5gnr_response * response )
```

Implements rate dematching.

Parameters

in	*request*	Structure containing the configuration, input data
out	*response*	Structure containing the output data.

7.52.2.2 bblib_rate_dematching_5gnr_version()

```
int16_t bblib_rate_dematching_5gnr_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_rate_dematching_5gnr library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 when success, -1 when fail

7.53 phy_rate_match.h File Reference

Data Structures

- struct bblib_rate_match_dl_request
- struct bblib_rate_match_dl_response
- struct bblib_rate_match_ul_request
- struct bblib_rate_match_ul_response
- struct bblib_harq_combine_ul_request
- struct bblib_harq_combine_ul_response
- struct bblib_deinterleave_ul_request
- struct bblib_deinterleave_ul_response
- struct bblib_turbo_adapter_ul_request
- struct bblib_turbo_adapter_ul_response

Enumerations

- enum circular_buffer_format { BBLIB_CIRCULAR_BUFFER_WITHOUT_PADDING = 0, BBLIB_FULL_CIRCULAR_BUFFER = 1 }

Functions

- int16_t bblib_rate_match_version (char ∗version, int buffer_size)

- int32_t bblib_rate_match_dl (const struct bblib_rate_match_dl_request ∗request, struct bblib_rate_match_dl_response ∗response)
- int32_t **bblib_rate_match_dl_sse** (const struct bblib_rate_match_dl_request ∗request, struct bblib_rate_match_dl_response ∗response)
- int32_t **bblib_rate_match_dl_avx2** (const struct bblib_rate_match_dl_request ∗request, struct bblib_rate_match_dl_response ∗response)

- int32_t bblib_rate_match_ul (const struct bblib_rate_match_ul_request ∗request, struct bblib_rate_match_ul_response ∗response)
- int32_t **bblib_rate_match_ul_avx2** (const struct bblib_rate_match_ul_request ∗request, struct bblib_rate_match_ul_response ∗response)
- int32_t **bblib_rate_match_ul_avx512** (const struct bblib_rate_match_ul_request ∗request, struct bblib_rate_match_ul_response ∗response)

- int32_t bblib_harq_combine_ul (const struct bblib_harq_combine_ul_request *request, struct bblib_harq_combine_ul_response *response)
- int32_t **bblib_harq_combine_ul_avx2** (const struct bblib_harq_combine_ul_request *request, struct bblib_harq_combine_ul_response *response)
- int32_t **bblib_harq_combine_ul_avx512** (const struct bblib_harq_combine_ul_request *request, struct bblib_harq_combine_ul_response *response)

- int32_t bblib_deinterleave_ul (const struct bblib_deinterleave_ul_request *request, struct bblib_deinterleave_ul_response *response)
- int32_t **bblib_deinterleave_ul_avx2** (const struct bblib_deinterleave_ul_request *request, struct bblib_deinterleave_ul_response *response)
- int32_t **bblib_deinterleave_ul_avx512** (const struct bblib_deinterleave_ul_request *request, struct bblib_deinterleave_ul_response *response)

- int32_t bblib_turbo_adapter_ul (const struct bblib_turbo_adapter_ul_request *request, struct bblib_turbo_adapter_ul_response *response)
- int32_t **bblib_turbo_adapter_ul_avx2** (const struct bblib_turbo_adapter_ul_request *request, struct bblib_turbo_adapter_ul_response *response)
- int32_t **bblib_turbo_adapter_ul_avx512** (const struct bblib_turbo_adapter_ul_request *request, struct bblib_turbo_adapter_ul_response *response)

7.53.1 Detailed Description

External API for LTE Rate Matching, Dematching (HARQ & deinterleaver) functions in LTE.

7.53.2 Enumeration Type Documentation

7.53.2.1 circular_buffer_format

```
enum circular_buffer_format
```

circular buffer format; defines format of circular buffer given as input

Enumerator

BBLIB_CIRCULAR_BUFFER_WITHOUT_PADDING	Cicular buffer without dummy padding bits.
BBLIB_FULL_CIRCULAR_BUFFER	Full circular buffer, i.e. with dummy padding bits as discribed in 3GPP 36.212 subclause 5.1.4.1.1.

7.53.3 Function Documentation

7.53.3.1 bblib_deinterleave_ul()

```
int32_t bblib_deinterleave_ul (
            const struct bblib_deinterleave_ul_request * request,
            struct bblib_deinterleave_ul_response * response )
```

Subblock deinterleaving for LTE.

Use circular_buffer_format in request struct to indicate type of input buffer:

- BBLIB_CIRCULAR_BUFFER_WITHOUT_PADDING - for circular buffer without dummy bits;
- BBLIB_FULL_CIRCULAR_BUFFER for full circular buffer (with dummy padding bits).

Parameters

in	*request*	structure containing configuration information and input data
out	*response*	structure containing kernel outputs

Returns

Success: return 0, else: return -1

7.53.3.2 bblib_harq_combine_ul()

```
int32_t bblib_harq_combine_ul (
            const struct bblib_harq_combine_ul_request * request,
            struct bblib_harq_combine_ul_response * response )
```

HARQ combining for LTE.

Parameters

in	*request*	structure containing configuration information and input data
out	*response*	structure containing kernel outputs

Returns

Success: return 0, else: return -1

7.53.3.3 bblib_rate_match_dl()

```
int32_t bblib_rate_match_dl (
            const struct bblib_rate_match_dl_request * request,
            struct bblib_rate_match_dl_response * response )
```

Downlink rate matching for LTE.

Parameters

in	*request*	Structure containing configuration information and input data.
out	*response*	Structure containing kernel outputs.

Note

bblib_rate_match_dl provides the most appropriate version for the available ISA, the _avx2 etc. version allow direct access to specific ISA implementations.

Returns

Success: return 0, else: return -1.

7.53.3.4 bblib_rate_match_ul()

```
int32_t bblib_rate_match_ul (
            const struct bblib_rate_match_ul_request * request,
            struct bblib_rate_match_ul_response * response )
```

Uplink rate matching for LTE.

Includes HARQ combining, subblock deinterleaving and data formatting for turbo decoding.

Parameters

in	*request*	Structure containing configuration information and input data.
out	*response*	Structure containing kernel outputs.

Returns

Success: return 0, else: return -1.

7.53.3.5 bblib_rate_match_version()

```
int16_t bblib_rate_match_version (
            char * version,
            int buffer_size )
```

Report the version number for the rate match library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, typically no more than BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.53.3.6 bblib_turbo_adapter_ul()

```
int32_t bblib_turbo_adapter_ul (
            const struct bblib_turbo_adapter_ul_request * request,
            struct bblib_turbo_adapter_ul_response * response )
```

Data formatting for turbo decoding for LTE.

Parameters

in	*request*	structure containing configuration information and input data
out	*response*	structure containing kernel outputs

Returns

Success: return 0, else: return -1

7.54 phy_reed_muller.h File Reference

Data Structures

- struct bblib_reed_muller_dec_request
- struct bblib_reed_muller_dec_fxp_request
- struct bblib_reed_muller_dec_response
- struct bblib_reed_muller_fht_request
- struct bblib_reed_muller_fht_fxp_request
- struct bblib_reed_muller_fht_response
- struct bblib_reed_muller_conf_request
- struct bblib_reed_muller_conf_fxp_request

Enumerations

- enum bblib_reed_muller_code_type { BBLIB_RM_32_CODE, BBLIB_RM_20_CODE }

Functions

- int16_t bblib_reed_muller_version (char ∗version, int buffer_size)
- void bblib_print_reed_muller_version ()

- int bblib_reed_muller_dec (const struct bblib_reed_muller_dec_request ∗request, struct bblib_reed_muller_dec_response ∗response)
- int **bblib_reed_muller_dec_c** (const struct bblib_reed_muller_dec_request ∗request, struct bblib_reed_muller_dec_response ∗response)
- int **bblib_reed_muller_dec_avx2** (const struct bblib_reed_muller_dec_request ∗request, struct bblib_reed_muller_dec_response ∗response)
- int **bblib_reed_muller_dec_avx512** (const struct bblib_reed_muller_dec_request ∗request, struct bblib_reed_muller_dec_response ∗response)
- int **bblib_reed_muller_dec_fs** (const struct bblib_reed_muller_dec_request ∗request, struct bblib_reed_muller_dec_response ∗response)
- int **bblib_reed_muller_dec_fs_c** (const struct bblib_reed_muller_dec_request ∗request, struct bblib_reed_muller_dec_response ∗response)
- int **bblib_reed_muller_dec_fs_avx512** (const struct bblib_reed_muller_dec_request ∗request, struct bblib_reed_muller_dec_response ∗response)
- int **bblib_reed_muller_dec_fxp** (const struct bblib_reed_muller_dec_fxp_request ∗request, struct bblib_reed_muller_dec_response ∗response)
- int **bblib_reed_muller_dec_fxp_c** (const struct bblib_reed_muller_dec_fxp_request ∗request, struct bblib_reed_muller_dec_response ∗response)
- int **bblib_reed_muller_dec_fxp_avx2** (const struct bblib_reed_muller_dec_fxp_request ∗request, struct bblib_reed_muller_dec_response ∗response)
- int **bblib_reed_muller_dec_fxp_avx512** (const struct bblib_reed_muller_dec_fxp_request ∗request, struct bblib_reed_muller_dec_response ∗response)
- int **bblib_reed_muller_dec_fs_fxp** (const struct bblib_reed_muller_dec_fxp_request ∗request, struct bblib_reed_muller_dec_response ∗response)
- int **bblib_reed_muller_dec_fs_fxp_c** (const struct bblib_reed_muller_dec_fxp_request ∗request, struct bblib_reed_muller_dec_response ∗response)
- int **bblib_reed_muller_dec_fs_fxp_avx512** (const struct bblib_reed_muller_dec_fxp_request ∗request, struct bblib_reed_muller_dec_response ∗response)

- int bblib_reed_muller_dec_fht (const struct bblib_reed_muller_fht_request ∗request, struct bblib_reed_muller_fht_response ∗response)
- int **bblib_reed_muller_dec_fht_c** (const struct bblib_reed_muller_fht_request ∗request, struct bblib_reed_muller_fht_response ∗response)
- int **bblib_reed_muller_dec_fht_avx2** (const struct bblib_reed_muller_fht_request ∗request, struct bblib_reed_muller_fht_response ∗response)
- int **bblib_reed_muller_dec_fht_fxp** (const struct bblib_reed_muller_fht_fxp_request ∗request, struct bblib_reed_muller_fht_response ∗response)
- int **bblib_reed_muller_dec_fht_fxp_c** (const struct bblib_reed_muller_fht_fxp_request ∗request, struct bblib_reed_muller_fht_response ∗response)

- uint16_t bblib_reed_muller_dec_conf (const struct bblib_reed_muller_conf_request ∗request)
- uint16_t **bblib_reed_muller_dec_conf_c** (const struct bblib_reed_muller_conf_request ∗request)
- uint16_t **bblib_reed_muller_dec_conf_avx2** (const struct bblib_reed_muller_conf_request ∗request)
- uint16_t **bblib_reed_muller_dec_conf_fxp** (const struct bblib_reed_muller_conf_fxp_request ∗request)
- uint16_t **bblib_reed_muller_dec_conf_fxp_c** (const struct bblib_reed_muller_conf_fxp_request ∗request)

7.54.1 Detailed Description

External API for LTE and 5GNR Reed-Muller decoder.

7.54.2 Enumeration Type Documentation

7.54.2.1 bblib_reed_muller_code_type

```
enum bblib_reed_muller_code_type
```

Enum defining different types of reed muller code.

Enumerator

BBLIB_RM_32_CODE	(32, X) RM code will be used.
BBLIB_RM_20_CODE	(20, X) RM code will be used.

7.54.3 Function Documentation

7.54.3.1 bblib_reed_muller_dec()

```
int bblib_reed_muller_dec (
            const struct bblib_reed_muller_dec_request * request,
            struct bblib_reed_muller_dec_response * response )
```

Reed-Muller decoding.

Algorithm will use majority logic decoding scheme for decoding. The decoder output bits are packed into a 16-bit output word (msb first, lsb last). Precomputed encoder and decoder tables will be used.

fxp functions use fixed point input in the format 16s12. In both floating point and fixed point the resulting output is the same.

Parameters

in	*request*	Structure containing input data, length of thr input buffer, number of bits to be decoded and type of the operation.
out	*response*	Structure containing the output data and length of it.

Returns

0 on success, negative on failure.

7.54.3.2 bblib_reed_muller_dec_conf()

```
uint16_t bblib_reed_muller_dec_conf (
            const struct bblib_reed_muller_conf_request * request )
```

Function bblib_reed_muller_dec_conf implements a function that computes confidence measures for the (32, X) and (20, X) Reed Muller decoder outputs.

Algorithm will re-encode the decoded output and then compare against the soft decisions received. Two metrics will be computed to determine if algorithm can determine that a valid Reed-Muller was received. The first metric will be comparing hard decisions against received soft decisions. The second metric is summing soft decisions of the correct soft decisions versus the bad ones.

fxp functions use fixed point input in the format 16s12. In both floating point and fixed point the resulting output is the same.

Parameters

in	*request*	Structure containing input data, length of thr input buffer, decoded_data, number of bits in the decoded data and type of the operation.

Returns

Reed-Muller decoder confidence, (0 = fail, 1 = pass).

7.54.3.3 bblib_reed_muller_dec_fht()

```
int bblib_reed_muller_dec_fht (
            const struct bblib_reed_muller_fht_request * request,
            struct bblib_reed_muller_fht_response * response )
```

Reed-Muller decoding.

Algorithm will use Fast Hadamard Transform decoding scheme for decoding. The decoder output bits are packed into a 16-bit output word (msb first, lsb last).

fxp functions use fixed point input in the format 16s12. In both floating point and fixed point the resulting output is the same.

Parameters

in	*request*	Structure containing input data, length of thr input buffer and number of bits to be decoded.
out	*response*	Structure containing the output data and length of it

Returns

0 on success, negative on failure.

7.54.3.4 bblib_reed_muller_version()

```
int16_t bblib_reed_muller_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_reed_muller library.

Parameters

in	*version*	pointer to a char buffer where the version string should be copied.
in	*buffer_size*	the length of the string buffer, typically no more than BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.55 phy_remapping_ctrlch.h File Reference

Data Structures

- struct Matrix
- struct bblib_pdcch_remapping_request
- struct bblib_pdcch_remapping_response
- struct bblib_pbch_remapping_request
- struct bblib_pbch_remapping_response

Enumerations

- enum BandwidthEnum {
  B1p4M, **B3M**, **B5M**, **B10M**,
  B15M, **B20M** }

Functions

- int16_t bblib_lte_remapping_ctrlch_version (char *version, int buffer_size)

- void bblib_pdcch_remapping (const struct bblib_pdcch_remapping_request *request, struct bblib_pdcch_remapping_response *response)

- void **bblib_pdcch_remapping_c** (const struct bblib_pdcch_remapping_request ∗request, struct bblib_pdcch_remapping_response ∗response)

- void bblib_pbch_remapping (const struct bblib_pbch_remapping_request ∗request, struct bblib_pbch_remapping_response ∗response)
- void **bblib_pbch_remapping_c** (const struct bblib_pbch_remapping_request ∗request, struct bblib_pbch_remapping_response ∗response)

7.55.1 Detailed Description

: Header for phy_remapping_ctrlch.cpp.

7.55.2 Function Documentation

7.55.2.1 bblib_lte_remapping_ctrlch_version()

```
int16_t bblib_lte_remapping_ctrlch_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_lte_remapping_ctrlch library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, typically no more than BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.55.2.2 bblib_pbch_remapping()

```
void bblib_pbch_remapping (
            const struct bblib_pbch_remapping_request * request,
            struct bblib_pbch_remapping_response * response )
```

PBCH RE mapping for a subframe.

Parameters

in	*request*	Structure containing inputs.
out	*response*	Structure containing outputs.

7.55.2.3 bblib_pdcch_remapping()

```
void bblib_pdcch_remapping (
            const struct bblib_pdcch_remapping_request * request,
            struct bblib_pdcch_remapping_response * response )
```

PDCCH RE mapping for a subframe.

Parameters

in	*request*	Structure containing inputs.
out	*response*	Structure containing outputs.

7.56 phy_remapping_pdcch_5gnr.h File Reference

Data Structures

- struct bblib_pdcch_remapping_5gnr_request
- struct bblib_pdcch_remapping_5gnr_response

Enumerations

- enum bblib_remapping_config { BBLIB_REMAPPING_PDCCH_5GNR_MAX_SYMBOL = 3 }

Functions

- int16_t bblib_pdcch_remapping_5gnr_version (char *version, int buffer_size)

- int16_t bblib_pdcch_remapping_5gnr (const struct bblib_pdcch_remapping_5gnr_request *request, struct bblib_pdcch_remapping_5gnr_response *response)
- int16_t **bblib_pdcch_remapping_5gnr_c** (const struct bblib_pdcch_remapping_5gnr_request *request, struct bblib_pdcch_remapping_5gnr_response *response)
- int16_t **bblib_pdcch_remapping_5gnr_avx2** (const struct bblib_pdcch_remapping_5gnr_request *request, struct bblib_pdcch_remapping_5gnr_response *response)
- int16_t **bblib_pdcch_remapping_5gnr_avx512** (const struct bblib_pdcch_remapping_5gnr_request *request, struct bblib_pdcch_remapping_5gnr_response *response)

7.56.1 Detailed Description

: Header for phy_remapping_pdcch_5gnr.cpp.

7.56.2 Enumeration Type Documentation

7.56.2.1 bblib_remapping_config

```
enum bblib_remapping_config
```

Constants used in PDCCH remapping 5gnr.

Enumerator

BBLIB_REMAPPING_PDCCH_5GNR_MAX_SYMBOL	PDCCH remapping 5gnr allow max symbols using.

7.56.3 Function Documentation

7.56.3.1 bblib_pdcch_remapping_5gnr()

```
int16_t bblib_pdcch_remapping_5gnr (
            const struct bblib_pdcch_remapping_5gnr_request * request,
            struct bblib_pdcch_remapping_5gnr_response * response )
```

PDCCH RE mapping for a subframe. In the 3GPP 38.211 section 7.3.2.2.

Parameters

in	*request*	Structure containing inputs.
out	*response*	Structure containing outputs.

Returns

0 on success, otherwise -1.

7.56.3.2 bblib_pdcch_remapping_5gnr_version()

```
int16_t bblib_pdcch_remapping_5gnr_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_pdcch_remapping_5gnr_version library.

Parameters

in	*version*	pointer to a char buffer where the version string should be copied.
in	*buffer_size*	the length of the string buffer, typically no more than BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.57 phy_remapping_pdsch.h File Reference

Data Structures

- struct reMappingInput
- struct bblib_remapping_pdsch_request
- struct bblib_remapping_pdsch_response

Enumerations

- enum enReMapType { RbMapTypeA = 0, RbMapTypeB = 1, RbMapTypeC = 2 }
- enum enSymbolPatternType {
  SymbType1 = 0, SymbType2 = 1, SymbType3 = 2, SymbType4 = 3,
  SymbType5 = 4, SymbType6 = 5 }

Functions

- int16_t bblib_lte_remapping_pdsch_version (char ∗version, int buffer_size)
- int16_t bblib_remapping_pdsch (const struct bblib_remapping_pdsch_request ∗request, struct bblib_remapping_pdsch_respons ∗response)
- int16_t **bblib_remapping_pdsch_c** (const struct bblib_remapping_pdsch_request ∗request, struct bblib_remapping_pdsch_response ∗response)

7.57.1 Detailed Description

External API for remapping PDSCH.

7.57.2 Enumeration Type Documentation

7.57.2.1 enReMapType

`enum enReMapType`

Defined RB map type - RE mapping would be different by type.

Enumerator

RbMapTypeA	All RBnumber >52 or all RBnumber <47 or 47 =<all RBnumber<=52
RbMapTypeB	RBnumber start < 47 and RBnumber end >=47 and RBnumber end<=52 or RBnumber end > 52 and RBnumber start >=47 and RBnumber start<=52.
RbMapTypeC	RBnumber start<47 and RBnumber end>52.

7.57.2.2 enSymbolPatternType

```
enum enSymbolPatternType
```

Defined symbol type - RRE mapping would be different by type.

Enumerator

SymbType1	number = 8 cellID%3==0 or 3 position occupy by RS.
SymbType2	number = 8 cellID%3==1 or 4 position occupy by RS.
SymbType3	number = 8 cellID%3==2 or 5 position occupy by RS.
SymbType4	number = 0 occupy by PDCCH or PBCH or PSS or SSS.
SymbType5	number = 12 all of is for PDSCH.
SymbType6	[needs updating].

7.57.3 Function Documentation

7.57.3.1 bblib_lte_remapping_pdsch_version()

```
int16_t bblib_lte_remapping_pdsch_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_lte_remapping_pdsch library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, typically no more than BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.57.3.2 bblib_remapping_pdsch()

```
int16_t bblib_remapping_pdsch (
            const struct bblib_remapping_pdsch_request * request,
            struct bblib_remapping_pdsch_response * response )
```

Implements PDSCH remapping for LTE.

Parameters

in	*request*	Structure containing the configuration and input data buffers.
out	*response*	Structure containing the output data.

Returns

0 on success, -1 otherwise.

7.58 phy_rx_mimo_mmse.h File Reference

Data Structures

- struct bblib_mmse_mimo_request
- struct bblib_mmse_mimo_response

Functions

- int16_t bblib_mimo_mmse_detection_version (char *version, int buffer_size)
- int32_t bblib_mimo_mmse_detection (bblib_mmse_mimo_request *request, bblib_mmse_mimo_response *response)
- int32_t **bblib_mimo_mmse_detection_avx512** (bblib_mmse_mimo_request *request, bblib_mmse_mimo_response *response)
- int32_t matrix_inv_lemma_4x4 (__m512 ftempARe[4][4], __m512 ftempAIm[4][4], __m512 ftempInvARe[4][4], __m512 ftempInvAIm[4][4], int16_t nFixedBitsSquare)

7.58.1 Detailed Description

External API for MMSE MIMO detection, with post SINR calculation.

Overview: This module implements MMSE MIMO detection in 5GNR, with post SINR calculation. It can support 1TX1R, 1TX2R, 1TX4R, 1TX8R, 2TX2R, 2TX4R, 2TX8R, 4TX4R, 4TX8R, 4TX16RX 8TX8R, 8TX16RX, 16TX16RX antennas.

Algorithm Guidance:

1. Calculate weighting matrix W = inv(H' * H + Sigma2*I) * H', where H is channel transfer function in frequenchy domain among Tx and Rx antennas Sigma2 is noise power
2. Multiply weighting matrix with input signal from Rx antennas: X = W * Y get estimated Tx signal.
3. For post SINR, let gain = real(diag(inv(H'H+sigma2)*H'*H)) post SINR = gain ./ (1-gain)

7.58.2 Function Documentation

7.58.2.1 bblib_mimo_mmse_detection()

```
int32_t bblib_mimo_mmse_detection (
            bblib_mmse_mimo_request * request,
            bblib_mmse_mimo_response * response )
```

MMSE MIMO detection, with post SNR calculation.

Parameters

in	*request*	Input request structure for MMSE MIMO.
out	*response*	Output response structure for MMSE MIMO..

Returns

0 for success, and -1 for error

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.58.2.2 bblib_mimo_mmse_detection_version()

```
int16_t bblib_mimo_mmse_detection_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_mimo_mmse_detection library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.58.2.3 matrix_inv_lemma_4x4()

```
int32_t matrix_inv_lemma_4x4 (
            __m512 ftempARe[4][4],
            __m512 ftempAIm[4][4],
            __m512 ftempInvARe[4][4],
            __m512 ftempInvAIm[4][4],
            int16_t nFixedBitsSquare )
```

matrix inverse for 4x4, using lemma method

Parameters

in	*ftempARe*	is the real part of the input matrix
in	*ftempAIm*	is the imaginary part of the input matrix
in	*nFixedBitsSquare*	is the square value of the decimal digits number for the fixed point input data
out	*ftempInvARe*	is the real part of the inversed matrix
out	*ftempInvAIm*	is the imaginary part of the inversed matrix

Returns

0 for success, and -1 for error

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.59 phy_sample_kernel.h File Reference

Data Structures

- struct bblib_sample_kernel_request
- struct bblib_sample_kernel_response

Functions

- int16_t bblib_sample_kernel_version (char ∗version, int buffer_size)

- int bblib_sample_kernel (const struct bblib_sample_kernel_request ∗request, struct bblib_sample_kernel_response ∗response)
- int **bblib_sample_kernel_c** (const struct bblib_sample_kernel_request ∗request, struct bblib_sample_kernel_response ∗response)
- int **bblib_sample_kernel_avx2** (const struct bblib_sample_kernel_request ∗request, struct bblib_sample_kernel_response ∗response)
- int **bblib_sample_kernel_avx512** (const struct bblib_sample_kernel_request ∗request, struct bblib_sample_kernel_response ∗response)
- int **bblib_sample_kernel_fxp** (const struct bblib_sample_kernel_request ∗request, struct bblib_sample_kernel_response ∗response)
- int **bblib_sample_kernel_fxp_c** (const struct bblib_sample_kernel_request ∗request, struct bblib_sample_kernel_response ∗response)
- int **bblib_sample_kernel_fxp_avx2** (const struct bblib_sample_kernel_request ∗request, struct bblib_sample_kernel_response ∗response)
- int **bblib_sample_kernel_fxp_avx512** (const struct bblib_sample_kernel_request ∗request, struct bblib_sample_kernel_response ∗response)

7.59.1 Detailed Description

This is a sample Kernel module that can be used as a template to create new kernels from or to simply explore a simple SDK kernel to understand the programming paradigms and build structure.

Overview: Each kernel should contain an overview section that details the implementation of that Kernel.

The purpose of the sample kernel is to implement a simple complex multiplication operation to demonstrate how to implement t a kernel in the SDK.

Requirements and Test Coverage:

Detail The requirements and test coverage in this section for the Kernel.

Functional Test Cases:

- Q16s12 Fixed point implementation of complex multiplication
- 32bit floating point implementation of complex multiplication

Functional tests are compared with a Matlab model.

Performance Test: cycle count measurements for fixed and floating point with 1200 input symbols

Algorithm Guidance:

Detail any algorithm guidance here in this section.

The Kernel implements a standard complex multiplication on two input buffers and places the result in output buffer.

7.59.2 Function Documentation

7.59.2.1 bblib_sample_kernel()

```
int bblib_sample_kernel (
            const struct bblib_sample_kernel_request * request,
            struct bblib_sample_kernel_response * response )
```

sample kernel procedures.

Parameters

in	*request*	Structure containing the input buffers and lengths.
out	*response*	Structure containing the output buffer.

Returns

0 if successful, negative on error

Note

Each kernel shall have a generic API that will choose the implementation based on the CPU type that is running. Along with the generic API a plain C implementation is mandatory for all Kernels in the SDK. The C implementation is useful for debugging and also understanding of the algorithm implemented. The C implementation shall use the _c appended to the API name. Other ISA specific implementations shall then be defined using the _sse, _avx2, _avx512 appended to the same API name. 5G Specific kernels should use the _5gnr appendix to the function name. Generally all fxp implementation should use the _fxp in the API name while flp does not.

7.59.2.2 bblib_sample_kernel_version()

```
int16_t bblib_sample_kernel_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_sample_kernel library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, length BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

Note

All Kernels shall contain a version function.

7.60 phy_scramble.h File Reference

Data Structures

- struct bblib_scramble_request
- struct bblib_scramble_response
- struct bblib_descramble_request
- struct bblib_descramble_response

Functions

- int16_t bblib_lte_scramble_version (char ∗version, int buffer_size)

- int32_t bblib_scramble (const struct bblib_scramble_request *request, struct bblib_scramble_response *response)
- int32_t **bblib_scramble_c** (const struct bblib_scramble_request *request, struct bblib_scramble_response *response)
- int32_t **bblib_scramble_avx2** (const struct bblib_scramble_request *request, struct bblib_scramble_response *response)
- int32_t **bblib_scramble_avx512** (const struct bblib_scramble_request *request, struct bblib_scramble_response *response)

- int32_t bblib_descramble (const struct bblib_descramble_request *request, struct bblib_descramble_response *response)
- int32_t **bblib_descramble_c** (const struct bblib_descramble_request *request, struct bblib_descramble_response *response)
- int32_t **bblib_descramble_avx2** (const struct bblib_descramble_request *request, struct bblib_descramble_response *response)
- int32_t **bblib_descramble_avx512** (const struct bblib_descramble_request *request, struct bblib_descramble_response *response)

7.60.1 Detailed Description

External API for scrambling and descrambling functions.

7.60.2 Function Documentation

7.60.2.1 bblib_descramble()

```
int32_t bblib_descramble (
            const struct bblib_descramble_request * request,
            struct bblib_descramble_response * response )
```

Descrambling procedure.

Parameters

in	*request*	Structure containing the input data, selected data type, the length of input data and the value of init.
out	*response*	Structure containing the output data and length of it.

Note

This function is only used for eNobe UL channel.
refers to 3GPP TS 136.211.

Returns

0 for success, -1 for error

7.60.2.2 bblib_lte_scramble_version()

```
int16_t bblib_lte_scramble_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_lte_scramble library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.60.2.3 bblib_scramble()

```
int32_t bblib_scramble (
            const struct bblib_scramble_request * request,
            struct bblib_scramble_response * response )
```

Scrambling procedure.

Parameters

in	*request*	Structure containing the input data, data length and the initial value of init.
out	*response*	Structure containing the output data and length of it.

Note

This function is used for eNobe DL channels and UE UL channel.
refers to 3GPP TS 36.211.

Returns

0 for success, -1 for error.

7.61 phy_scramble_5gnr.h File Reference

Data Structures

- struct bblib_scramble_5gnr_request
- struct bblib_scramble_5gnr_response

Functions

- int16_t bblib_scramble_5gnr_version (char *version, int buffer_size)

- int32_t bblib_scramble_5gnr (const struct bblib_scramble_5gnr_request *request, struct bblib_scramble_5gnr_response *response)
- int32_t **bblib_scramble_5gnr_c** (const struct bblib_scramble_5gnr_request *request, struct bblib_scramble_5gnr_response *response)
- int32_t **bblib_scramble_5gnr_avx2** (const struct bblib_scramble_5gnr_request *request, struct bblib_scramble_5gnr_response *response)
- int32_t **bblib_scramble_5gnr_avx512** (const struct bblib_scramble_5gnr_request *request, struct bblib_scramble_5gnr_response *response)
- int32_t **bblib_scramble_5gnr_snc** (const struct bblib_scramble_5gnr_request *request, struct bblib_scramble_5gnr_response *response)

- int32_t bblib_descramble_5gnr (const struct bblib_scramble_5gnr_request *request, struct bblib_scramble_5gnr_response *response)
- int32_t **bblib_descramble_5gnr_c** (const struct bblib_scramble_5gnr_request *request, struct bblib_scramble_5gnr_response *response)
- int32_t **bblib_descramble_5gnr_avx2** (const struct bblib_scramble_5gnr_request *request, struct bblib_scramble_5gnr_response *response)
- int32_t **bblib_descramble_5gnr_avx512** (const struct bblib_scramble_5gnr_request *request, struct bblib_scramble_5gnr_response *response)
- int32_t **bblib_descramble_5gnr_snc** (const struct bblib_scramble_5gnr_request *request, struct bblib_scramble_5gnr_response *response)

7.61.1 Detailed Description

External API for scrambling and descrambling functions.

7.61.2 Function Documentation

7.61.2.1 bblib_descramble_5gnr()

```
int32_t bblib_descramble_5gnr (
            const struct bblib_scramble_5gnr_request * request,
            struct bblib_scramble_5gnr_response * response )
```

Descrambling procedure Implements 4G/5G descrambling as defined in TS36.211/TS38.211 Descrambling takes a sequence of 8 bit LLRs and scrambles them LLR by LLR by sign flipping based on a scramble code sequence.

Scramble sequence c(n) defined as c(n) = (x1(n + Nc) + x2(n + Nc))mod2

Where x1(n + 31) = (x1(n + 3) + x1(n))mod2 x2(n + 31) = (x2(n + 3) + x2(n + 2) + x2(n + 1) + x2(n))mod2

Parameters

in	*request*	Structure containing the input data, selected data type, the length of input data and the value of init.
out	*response*	Structure containing the output data and length of it.

Note

This function is only used for eNobe UL channel.

Returns

0 for success, -1 for error

7.61.2.2 bblib_scramble_5gnr()

```
int32_t bblib_scramble_5gnr (
            const struct bblib_scramble_5gnr_request * request,
            struct bblib_scramble_5gnr_response * response )
```

Scrambling procedure Implements 4G/5G scramble code generation as defined in TS36.211/TS38.211 Scrambling takes a sequence of bits and scrambles them bit by bit by XORing with a scramble code sequence.

Scramble sequence c(n) defined as c(n) = (x1(n + Nc) + x2(n + Nc))mod2

Where x1(n + 31) = (x1(n + 3) + x1(n))mod2 x2(n + 31) = (x2(n + 3) + x2(n + 2) + x2(n + 1) + x2(n))mod2

Parameters

in	*request*	Structure containing the input data, data length and the Cinit value
out	*response*	Structure containing the output data and length of it.

Note

This function is used for eNobe DL channels and UE UL channel.

Returns

0 for success, -1 for error.

7.61.2.3 bblib_scramble_5gnr_version()

```
int16_t bblib_scramble_5gnr_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_scramble_5gnr library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.62 phy_srs_cestimate_5gnr.h File Reference

Data Structures

- struct bblib_srs_cestimate_5gnr_request
- struct bblib_srs_cestimate_5gnr_response

Enumerations

- enum bblib_srs_measurement_config {
BBLIB_SRS_MAX_SYMBOL = 4, BBLIB_SRS_MAX_RX_ANT = 64, BBLIB_SRS_MAX_PORT_PER_UE = 4, BBLIB_SRS_MIN_CARRIERS = 24,
BBLIB_SRS_PRBS_PER_BLOCK = 4, BBLIB_SRS_MAX_BLOCKS = 110, BBLIB_SRS_EST_FACTOR = 24, BBLIB_SRS_COMB2_RS_PER_RB = 6,
BBLIB_SRS_COMB4_RS_PER_RB = 3, BBLIB_SRS_MAX_UE_NUM = 12, BBLIB_SRS_COMB2_MAX_UE_NUM = 8, BBLIB_SRS_CE_SCALE = 12,
BBLIB_SRS_SRS_COMB2 = 2, BBLIB_SRS_SRS_COMB4 = 4, BBLIB_SRS_SC_PER_PRB = 12,
BBLIB_SRS_SC_CE_NOISE_EST_FACTOR = ((RX_DATA_FIXED_POINT-1)∗2),
BBLIB_SRS_MIN_RB_NUM = 4 }

Functions

- void bblib_srs_cestimate_5gnr (const struct bblib_srs_cestimate_5gnr_request ∗request, struct bblib_srs_cestimate_5gnr_response ∗response)
- void **bblib_srs_cestimate_5gnr_avx512** (const struct bblib_srs_cestimate_5gnr_request ∗request, struct bblib_srs_cestimate_5gnr_response ∗response)
- void **bblib_srs_cestimate_dft_5gnr** (const struct bblib_srs_cestimate_5gnr_request ∗request, struct bblib_srs_cestimate_5gnr_response ∗response)
- void **bblib_srs_cestimate_dft_5gnr_avx512** (const struct bblib_srs_cestimate_5gnr_request ∗request, struct bblib_srs_cestimate_5gnr_response ∗response)
- int16_t bblib_srs_cestimate_5gnr_version (char ∗version, int buffer_size)

7.62.1 Detailed Description

External API for 5G srs CE.

7.62.2 Enumeration Type Documentation

7.62.2.1 bblib_srs_measurement_config

```
enum bblib_srs_measurement_config
```

srs measurement parameters

Enumerator

BBLIB_SRS_MAX_SYMBOL	Max number of SRS symbols
BBLIB_SRS_MAX_RX_ANT	Max number of receiving antennas
BBLIB_SRS_MAX_PORT_PER_UE	Max number of ports per UE
BBLIB_SRS_MIN_CARRIERS	Min carriers of SRS occupied
BBLIB_SRS_PRBS_PER_BLOCK	RB numbers of each SRS block
BBLIB_SRS_MAX_BLOCKS	Max block numbers of SRS bandwidth
BBLIB_SRS_EST_FACTOR	Srs power estimation factor
BBLIB_SRS_COMB2_RS_PER_RB	Carrier number for each SRS RB when Comb = 2
BBLIB_SRS_COMB4_RS_PER_RB	Carrier number for each SRS RB when Comb = 4
BBLIB_SRS_MAX_UE_NUM	Max UE number
BBLIB_SRS_COMB2_MAX_UE_NUM	Max UE number of Comb2 scenario
BBLIB_SRS_CE_SCALE	Srs CE scale factor
BBLIB_SRS_SRS_COMB2	Srs subcarrier interval
BBLIB_SRS_SRS_COMB4	Srs subcarrier interval
BBLIB_SRS_SC_PER_PRB	Carrier number for each RB
BBLIB_SRS_SC_CE_NOISE_EST_FACTOR	CE_NOISE_EST_FACTOR
BBLIB_SRS_MIN_RB_NUM	srs min RB numbers>

7.62.3 Function Documentation

7.62.3.1 bblib_srs_cestimate_5gnr()

```
void bblib_srs_cestimate_5gnr (
            const struct bblib_srs_cestimate_5gnr_request * request,
            struct bblib_srs_cestimate_5gnr_response * response )
```

Implements SRS CE Massive MIMO.

Parameters

in	*request*	Structure containing the configuration, input data, lengths for different data types.
out	*response*	Structure containing the output data.

7.62.3.2 bblib_srs_cestimate_5gnr_version()

```
int16_t bblib_srs_cestimate_5gnr_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_srs_ce_5gnr library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 when success, -1 when fail

7.63 phy_ta_compensation_5gnr.h File Reference

Data Structures

- struct bblib_ta_compensation_request
- struct bblib_ta_compensation_response

Functions

- void bblib_ta_compensation_print_version ()
- int16_t bblib_ta_compensation_version_5gnr (char *version, int buffer_size)

- int32_t bblib_ta_compensation_5gnr (const struct bblib_ta_compensation_request *request, struct bblib_ta_compensation_response *response)
- int32_t **bblib_ta_compensation_5gnr_avx512** (const struct bblib_ta_compensation_request *request, struct bblib_ta_compensation_response *response)

7.63.1 Detailed Description

LTE ta_compensation_5gnr for the pusch.

The ta_compensation kernel is a module for adjust IQ data according to TA from channel estimation.

7.63.2 Function Documentation

7.63.2.1 bblib_ta_compensation_5gnr()

```
int32_t bblib_ta_compensation_5gnr (
            const struct bblib_ta_compensation_request * request,
            struct bblib_ta_compensation_response * response )
```

ta_compensation procedure.

Parameters

in	*request*	Structure containing the input symbols data length.
out	*response*	Structure containing the soft-bits and the number of outputs.

Returns

ta_compensation result, return 0 is success,return -1 is fail.

Note

Input and output buffers have to be 64 bytes aligned.
Only subset of the request parameters is used by each order of ta_compensation.

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.63.2.2 bblib_ta_compensation_print_version()

```
void bblib_ta_compensation_print_version ( )
```

printf ta_compensation version

Returns

null.

7.63.2.3 bblib_ta_compensation_version_5gnr()

```
int16_t bblib_ta_compensation_version_5gnr (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_lte_ta_compensation library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, has to be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.64 phy_tbcc.h File Reference

Data Structures

- struct bblib_tbcc_encoder_request
- struct bblib_tbcc_encoder_response

Functions

- int16_t bblib_lte_tbcc_version (char ∗version, int buffer_size)
- void bblib_print_tbcc_version ()

- int32_t bblib_tbcc_encoder (const struct bblib_tbcc_encoder_request ∗request, struct bblib_tbcc_encoder_response ∗response)
- int32_t **bblib_tbcc_encoder_avx2** (const struct bblib_tbcc_encoder_request ∗request, struct bblib_tbcc_encoder_response ∗response)
- int32_t **bblib_tbcc_encoder_c** (const struct bblib_tbcc_encoder_request ∗request, struct bblib_tbcc_encoder_response ∗response)

7.64.1 Detailed Description

API Definition for the convolutional coding functions used for 3GPP TS 36.212 Transport channels specifically 5.↩
1.3.1.

7.64.2 Function Documentation

7.64.2.1 bblib_lte_tbcc_version()

```
int16_t bblib_lte_tbcc_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_tbcc library.

Parameters

out	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, typically no more than BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.64.2.2 bblib_tbcc_encoder()

```
int32_t bblib_tbcc_encoder (
            const struct bblib_tbcc_encoder_request * request,
            struct bblib_tbcc_encoder_response * response )
```

This function implements the Covolutional encoder as per 3GPP TS 36.212 Section 5.1.3.1.

Parameters

request	Input data container.
response	Output data container.

Returns

0 on success, negative on error.

7.65 phy_turbo.h File Reference

Data Structures

- struct bblib_turbo_decoder_request
- struct bblib_turbo_decoder_response
- struct bblib_turbo_encoder_request
- struct bblib_turbo_encoder_response

Functions

- int16_t bblib_lte_turbo_version (char ∗version, int buffer_size)

- int32_t bblib_turbo_encoder (const struct bblib_turbo_encoder_request ∗request, struct bblib_turbo_encoder_response ∗response)

- int32_t **bblib_lte_turbo_encoder_sse** (const struct bblib_turbo_encoder_request *request, struct bblib_turbo_encoder_response *response)
- int32_t **bblib_lte_turbo_encoder_avx2** (const struct bblib_turbo_encoder_request *request, struct bblib_turbo_encoder_response *response)
- int32_t **bblib_lte_turbo_encoder_avx512** (const struct bblib_turbo_encoder_request *request, struct bblib_turbo_encoder_response *response)

- int32_t bblib_turbo_decoder (const struct bblib_turbo_decoder_request *request, struct bblib_turbo_decoder_response *response)
- int32_t **bblib_lte_turbo_decoder_64windows_avx512** (const struct bblib_turbo_decoder_request *request, struct bblib_turbo_decoder_response *response)
- int32_t **bblib_lte_turbo_decoder_32windows_avx2** (const struct bblib_turbo_decoder_request *request, struct bblib_turbo_decoder_response *response)
- int32_t **bblib_lte_turbo_decoder_16windows_sse** (const struct bblib_turbo_decoder_request *request, struct bblib_turbo_decoder_response *response)
- int32_t **bblib_lte_turbo_decoder_16windows_3iteration_sse** (const struct bblib_turbo_decoder_request *request, struct bblib_turbo_decoder_response *response)
- int32_t **bblib_lte_turbo_decoder_8windows_sse** (const struct bblib_turbo_decoder_request *request, struct bblib_turbo_decoder_response *response)

7.65.1 Detailed Description

External API for LTE turbo coder/decoder.

7.65.2 Function Documentation

7.65.2.1 bblib_lte_turbo_version()

```
int16_t bblib_lte_turbo_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_lte_turbo library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, has to be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.65.2.2 bblib_turbo_decoder()

```
int32_t bblib_turbo_decoder (
            const struct bblib_turbo_decoder_request * request,
            struct bblib_turbo_decoder_response * response )
```

Turbo decoder implementation for different windows sizes as defined in TS.36.212.

Parameters

request	Input data container.
response	Output data container.

Returns

Number of half iterations on success, negative on failure

7.65.2.3 bblib_turbo_encoder()

```
int32_t bblib_turbo_encoder (
            const struct bblib_turbo_encoder_request * request,
            struct bblib_turbo_encoder_response * response )
```

Turbo encoder implementation as defined in TS.36.212.

Parameters

request	Input data container.
response	Output data container.

Returns

0 on success, non-zero on failure.

7.66 phy_viterbi_decoder.h File Reference

Data Structures

- struct bblib_lte_viterbi_decoder_request
- struct bblib_lte_viterbi_decoder_response

Functions

- int16_t bblib_lte_viterbi_version (char *version, int buffer_size)

- int32_t bblib_lte_viterbi_decoder (const struct bblib_lte_viterbi_decoder_request *request, struct bblib_lte_viterbi_decoder_response *response)
- int32_t **bblib_lte_viterbi_decoder_avx2** (const struct bblib_lte_viterbi_decoder_request *request, struct bblib_lte_viterbi_decoder_response *response)
- int32_t **bblib_lte_viterbi_decoder_c** (const struct bblib_lte_viterbi_decoder_request *request, struct bblib_lte_viterbi_decoder_response *response)
- int32_t **bblib_lte_viterbi_decoder_sse** (const struct bblib_lte_viterbi_decoder_request *request, struct bblib_lte_viterbi_decoder_response *response)
- int32_t **bblib_lte_viterbi_decoder_avx512** (const struct bblib_lte_viterbi_decoder_request *request, struct bblib_lte_viterbi_decoder_response *response)

7.66.1 Detailed Description

Header file of the Viterbi Decoder.

7.66.2 Function Documentation

7.66.2.1 bblib_lte_viterbi_decoder()

```
int32_t bblib_lte_viterbi_decoder (
            const struct bblib_lte_viterbi_decoder_request * request,
            struct bblib_lte_viterbi_decoder_response * response )
```

Viterbi decoding implementation.

Parameters

request	Input data container.
response	Output data container.

Returns

0 for pass (1st iteration), 1 for pass (2nd iteration), -1 for fail.

7.66.2.2 bblib_lte_viterbi_version()

```
int16_t bblib_lte_viterbi_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_lte_viterbi library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, typically no more than BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.67 phy_zc_sequence_gen.h File Reference

Data Structures

- struct bblib_zc_sequence_gen_request
- struct bblib_zc_sequence_gen_response

Enumerations

- enum {
k_maxNumLayers = 8, k_numSubCarrPerRB = 12, k_numSubFramesPerFrame = 50, k_maxRBPerCarr = 275,
k_maxNumDMRS = 2 }
- enum sym_type { ZC_GEN_LTE, ZC_GEN_5GNR }

Functions

- int16_t bblib_zc_sequence_gen_version (char ∗version, int buffer_size)

- int bblib_zc_sequence_gen (const struct bblib_zc_sequence_gen_request ∗request, struct bblib_zc_sequence_gen_response ∗response)
- int **bblib_zc_sequence_gen_c** (const struct bblib_zc_sequence_gen_request ∗request, struct bblib_zc_sequence_gen_response ∗response)
- int **bblib_zc_sequence_gen_avx2** (const struct bblib_zc_sequence_gen_request ∗request, struct bblib_zc_sequence_gen_response ∗response)
- int **bblib_zc_sequence_gen_avx512** (const struct bblib_zc_sequence_gen_request ∗request, struct bblib_zc_sequence_gen_response ∗response)

7.67.1 Detailed Description

External API for Zadoff-Chu sequence generator.

Overview:

The lib_zc_sequence_gen kernel is a Zadoff-Chu sequence generator which supports both LTE, based on 3GPP TS 36.211 Release 14.2.0, sections 5.5.1 and 5.5.2, and 5G NR, based on 3GPP TS 38.211, sections 5.5.1 and 6.4.1, implementations.

This kernel can be used to generate the UL reference signal for both LTE and 5G NR, however it does not initially support precoding of reference signals as part of the kernel.

For LTE, the kernel has support for up to 4 layers with 1 DMRS. For 5G NR, the kernel has support for up to 8 layers, 1 DMRS for up to 4 layers and 2 DMRS for up to 8 layers.

Algorithm Guidance:

Currently, the kernel only supports a DMRS-add-pos value of 0.

7.67.2 Enumeration Type Documentation

7.67.2.1 anonymous enum

```
anonymous enum
```

Some standard static values for the sequence generator.

Enumerator

k_maxNumLayers	maximum number of layers supported.
k_numSubCarrPerRB	number of sub carriers per resource block.
k_numSubFramesPerFrame	number of subframes per frame.
k_maxRBPerCarr	maximum number of resource blocks supported.
k_maxNumDMRS	maximum number of DMRS symbols supported

7.67.2.2 sym_type

```
enum sym_type
```

Enum describing which spec to follow when generating the sequence.

Enumerator

ZC_GEN_LTE	generate Zadoff-Chu sequence for LTE.
ZC_GEN_5GNR	generate Zadoff-Chu sequence for 5GNR.

7.67.3 Function Documentation

7.67.3.1 bblib_zc_sequence_gen()

```
int bblib_zc_sequence_gen (
            const struct bblib_zc_sequence_gen_request * request,
            struct bblib_zc_sequence_gen_response * response )
```

Generates demodulation reference signals using a Zadoff-Chu sequence generator.

Parameters

in	*request*	Pointer to request structure containing the configuration settings required to generate the reference signal.
out	*response*	Pointer to a response structure containing the generated reference signals.

Returns

0 if output array was populated, otherwise -1.

7.67.3.2 bblib_zc_sequence_gen_version()

```
int16_t bblib_zc_sequence_gen_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_zc_sequence_gen library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, typically no more than BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.68 phy_zf_matrix_gen.h File Reference

Data Structures

- struct bblib_zf_matrix_gen_request
- struct bblib_zf_matrix_gen_response

Enumerations

- enum zf_matrix_gen_constants { BBLIB_ZF_MAX_RX_ANT_NUM = 64, BBLIB_ZF_MAX_TX_LAYER_NUM = 16, BBLIB_ZF_RX_DATA_FIXED_POINT = 13, BBLIB_ZF_W_LEFT_SHIFT = (BBLIB_ZF_RX_DATA_↩ FIXED_POINT∗2) }

Functions

- int16_t bblib_zf_matrix_gen_version (char ∗version, int buffer_size)

- int32_t bblib_zf_matrix_gen (const bblib_zf_matrix_gen_request ∗request, bblib_zf_matrix_gen_response ∗response)
- int32_t **bblib_zf_matrix_gen_avx512** (const bblib_zf_matrix_gen_request ∗request, bblib_zf_matrix_gen_response ∗response)

7.68.1 Detailed Description

External API for zf weight matrix generation in 5GNR.

Overview: This module implements zf weight matrix generation in 5GNR. It can support 32∗n and n∗32 configuration, n=1∼8. It can also support UL 8x64 and DL 64x16 configuration

Algorithm Guidance:

1. Calculate UL weight matrix W = inv(H' ∗ H) ∗ H',
2. Or calculate DL weight matrix W = H' ∗ inv(H ∗ H') where H is channel status in frequenchy domain

7.68.2 Enumeration Type Documentation

7.68.2.1 zf_matrix_gen_constants

```
enum zf_matrix_gen_constants
```

Constants used in zf weight matrix generation.

Enumerator

BBLIB_ZF_MAX_RX_ANT_NUM	MAX number of Rx antennas
BBLIB_ZF_MAX_TX_LAYER_NUM	MAX number of Tx layers
BBLIB_ZF_RX_DATA_FIXED_POINT	Fixed point of Rx data
BBLIB_ZF_W_LEFT_SHIFT	ZF W left shift

7.68.3 Function Documentation

7.68.3.1 bblib_zf_matrix_gen()

```
int32_t bblib_zf_matrix_gen (
            const bblib_zf_matrix_gen_request * request,
            bblib_zf_matrix_gen_response * response )
```

zf matrix generation.

Parameters

in	*request*	Input request structure .
out	*response*	Output response structure.

Returns

0 for success, and -1 for error

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

7.68.3.2 bblib_zf_matrix_gen_version()

```
int16_t bblib_zf_matrix_gen_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_zf_matrix_gen library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, must be at least BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.69 sdk_version.h File Reference

Functions

- int16_t bblib_sdk_version (char **buffer, const char **version, int buffer_size)
- int16_t bblib_common_version (char *version, int buffer_size)
- void bblib_print_common_version ()

7.69.1 Detailed Description

This file stores the SDK version number reported by the libraries.

7.69.2 Function Documentation

7.69.2.1 bblib_common_version()

```
int16_t bblib_common_version (
            char * version,
            int buffer_size )
```

Report the version number for the bblib_common library.

Parameters

in	*version*	Pointer to a char buffer where the version string should be copied.
in	*buffer_size*	The length of the string buffer, length BBLIB_SDK_VERSION_STRING_MAX_LEN characters.

Returns

0 if the version string was populated, otherwise -1.

7.69.2.2 bblib_sdk_version()

```
int16_t bblib_sdk_version (
            char ** buffer,
            const char ** version,
            int buffer_size )
```

Fill in the buffer_size long string array pointed by buff with the version string pointed by version.

Parameters

buffer	Output buffer.
version	Version string.
buffer_size	Size of the buffer.

Returns

0 if success, else -1.

7.70 singular_value_decomp.h File Reference

Data Structures

- struct bblib_singular_value_decomp_request
- struct bblib_singular_value_decomp_response

Functions

- int32_t bblib_singular_value_decomp (struct bblib_singular_value_decomp_request *request, struct bblib_singular_value_decomp_response *response)
- int32_t **bblib_singular_value_decomp_avx512** (struct bblib_singular_value_decomp_request *request, struct bblib_singular_value_decomp_response *response)

7.70.1 Detailed Description

External API for 5GNR Singular Value Decomposition.

Overview:

The lib_singular_value_decomp kernel is for channel matrix decomposition, which will be used in beamforming matrix generation

7.70.2 Function Documentation

7.70.2.1 bblib_singular_value_decomp()

```
int32_t bblib_singular_value_decomp (
            struct bblib_singular_value_decomp_request * request,
            struct bblib_singular_value_decomp_response * response )
```

This function implements singular value decomposition with AVX512 intrinsics.

Parameters

in	*request*	Input struct of singular value decomposition
out	*response*	Output struct of singular value decomposition

Returns

0 for success, and -1 for error

Warning

EXPERIMENTAL: Further optimization is possible, API may change in future release without prior notice.

Index