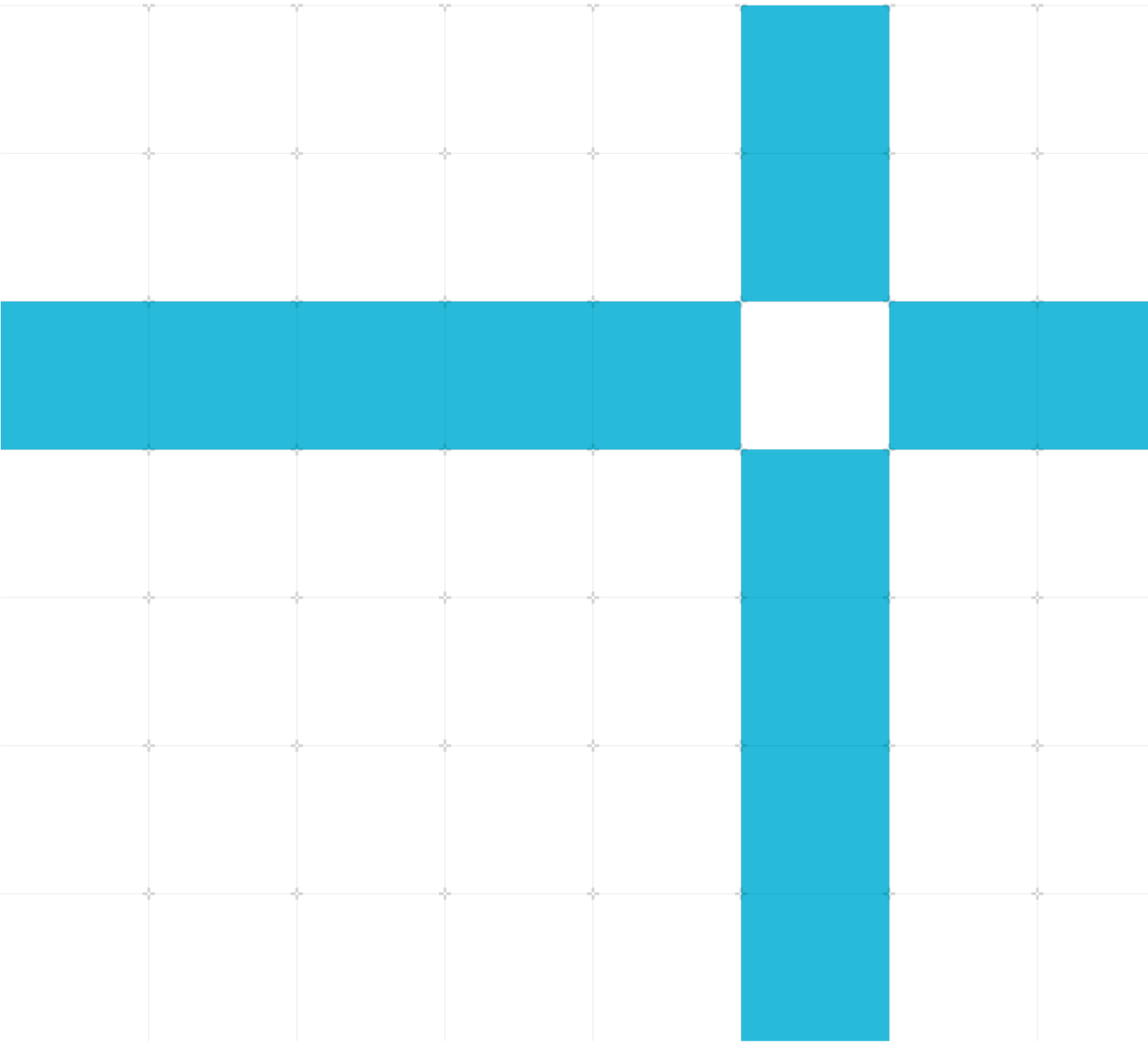
Arm RAN Acceleration Library

Product revision: 24.07

Release Notes

Non-Confidential

Document ID:
107561



Arm RAN Acceleration Library

Release Notes

Non-confidential proprietary notice

Arm Limited. Company 02557590 registered in England.
110 Fulbourn Road, Cambridge, England CB1 9NJ.
(LES-PRE-20349)

Confidentiality status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product status

The information in this document is Final, that is for a developed product.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on Arm RAN Acceleration Library, create a ticket on **https://support.developer.arm.com**.

To provide feedback on the document, fill the following survey:
https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

To report offensive language in this document, email **terms@arm.com**.

Contents

1 Release overview

The following sections describe the product to which these release notes relate and its quality status at time of release.

1.1 Product description

The Arm RAN Acceleration Library (ArmRAL) contains a set of functions for accelerating telecommunications applications such as, but not limited to, 5G Radio Access Networks (RANs). These functions are optimized for Arm AArch64-based processors.

ArmRAL provides:

- Vector functions

- Matrix functions

- Lower physical layer (Lower PHY) support functions

- Upper physical layer (Upper PHY) support functions

- Distributed Unit – Radio Unit (DU-RU) Interface support functions

ArmRAL includes functions that operate on 16-bit signed integers and 16-bit and 32-bit floating-point values.

1.2 Release status

This is the 24.07 release of ArmRAL.

These deliverables are being released under the terms of the agreement between Arm and each licensee (the "Agreement"). All planned verification and validation is complete.

The release is suitable for volume production under the terms of the Agreement.

1.3 Licensing information

Use of ArmRAL is subject to a BSD-3-Clause license, the text of which can be found in the `LICENSE.md` file in your product installation. We will receive inbound contributions under the same license.

If you require a different license than BSD-3-Clause for compatibility with your end product, please get in contact.

2 Release contents

ArmRAL releases contain documentation and source files.

The following sections describe:

- Cloning the product's git repository from Arm's GitLab
- The contents of this release
- Any changes since the previous release
- Any known issues and limitations that exist at the time of this release

2.1 Cloning the source repository

To access this release, clone the following repository using HTTPS:

```
git clone -b armral-24.07 https://git.gitlab.arm.com/networking/ral
```

2.2 Deliverables

The downloaded product includes the following deliverables:

- ArmRAL 24.07

- Release Notes (this document)

- Documentation

Product documentation is available on the Arm Developer website.

Documentation, errata and release notes might change between product releases. For the latest documentation bundle, check the product download page.

Arm tests its PDFs only in Adobe Acrobat and Acrobat Reader. Arm cannot guarantee the quality of this document when used with any other PDF reader.
A suitable PDF reader can be downloaded from **Adobe**.

2.3 Differences from previous release

The following subsections describe differences from the previous release of ArmRAL

2.3.1 Additions and functionality changes

This section describes new features or any technical changes to features or components in this release.

- For complex 32-bit floating point matrix multiplication, we recommend that you use `armral_cmplx_matmul_f32` for all cases. This function calls existing optimized special cases with minimal overhead and has new optimizations for larger cases.

- Renamed `armral_cmplx_mat_mult_aah_f32` to `armral_cmplx_matmul_aah_f32`. All arguments are in the same order and have the same meaning.

- Replaced `armral_cmplx_mat_mult_ahb_f32` with `armral_cmplx_matmul_ahb_f32`.
Note that the meanings of the parameters `m`, `n` and `k` differ between the old function and the new; a call to the old function of the form

```
armral_cmplx_mat_mult_ahb_f32(dim1, dim2, dim3, a, b, c);
```

becomes

```
armral_cmplx_matmul_ahb_f32(dim2, dim3, dim1, a, b, c);
```

- Replaced `armral_cmplx_mat_mult_i16` with `armral_cmplx_matmul_i16`.
Note that the meanings of the parameters `m`, `n` and `k` differ between the old function and the new; a call to the old function of the form

```
armral_cmplx_mat_mult_i16(dim1, dim2, dim3, a, b, c);
```

becomes

```
armral_cmplx_matmul_i16(dim1, dim3, dim2, a, b, c);
```

- Replaced `armral_cmplx_mat_mult_i16_32bit` with `armral_cmplx_matmul_i16_32bit`.
Note that the meanings of the parameters `m`, `n` and `k` differ between the old function and the new; a call to the old function of the form

```
armral_cmplx_mat_mult_i16_32bit(dim1, dim2, dim3, a, b, c);
```

becomes

```
armral_cmplx_matmul_i16_32bit(dim1, dim3, dim2, a, b, c);
```

- Replaced `armral_cmplx_mat_mult_f32` with `armral_cmplx_matmul_f32`.
Note that the meanings of the parameters `m`, `n` and `k` differ between the old function and the new; a call to the old function of the form

```
armral_cmplx_mat_mult_f32(dim1, dim2, dim3, a, b, c);
```

becomes

```
armral_cmplx_matmul_f32(dim1, dim3, dim2, a, b, c);
```

- The following functions now use unsigned integers in their interfaces to represent the lengths of vectors and the dimensions of matrices:

- `armral_cmplx_vecdot_f32`
- `armral_cmplx_vecdot_f32_2`
- `armral_cmplx_vecdot_i16`

- armral_cmplx_vecdot_i16_2
- armral_cmplx_vecdot_i16_32bit
- armral_cmplx_vecmul_f32
- armral_cmplx_vecmul_f32_2
- armral_cmplx_vecmul_i16
- armral_cmplx_vecmul_i16_2
- armral_corr_coeff_i16
- armral_svd_cf32
- armral_svd_cf32_noalloc
- armral_svd_cf32_noalloc_bufer_size

- Added the CMake option ARMRAL_ENABLE_WEXTRA to add the compiler flag -Wextra when building the library and tests.

2.3.2 Performance improvements

This section describes any features or components with improved performance.

- Performance improvements for the following routines:

- armral_cmplx_matmul_f32. For complex 32-bit floating point matrix multiplication, we recommend that you use this function for all cases. This function calls existing optimized special cases with minimal overhead and has new optimizations for larger cases.
- armral_turbo_decode_block and armral_turbo_decode_block_noalloc. These functions now operate internally on 16-bit integer values rather than 16-bit or 32-bit floating point values.

2.3.3 Changes to simulation programs

This section describes any changes, new features or components added to the channel simulation programs in this release.

- There are no changes to the channel simulation programs in this release.

2.3.4 Resolved issues

This section describes any known issues resolved in the current release.

- Documentation is now installed by the make install target, if it has been built.

- Corrected documentation for armal_cmplx_mat_inverse_batch_f32 and armal_cmplx_mat_inverse_batch_f32_pa to clarify that these functions have no restriction on batch sizes.- All functions in the library now use unsigned integers in their interfaces to represent the lengths of vectors and the dimensions of matrices. Previously some functions used signed integers and others used unsigned.

2.4 Known limitations

This section describes any known limitations of the current release.

- There are no known limitations in this release.

3 Support

If you have any issues with the installation, content, or use of this release, raise a question on the **Developer Community Forum.** Arm will respond as soon as possible.

3.1 Tools

To build ArmRAL you will need:

- A C/C++ compiler, such as GCC. ArmRAL has been tested with GCC 7.5.0, 8.5.0, 9.5.0, 10.5.0, 11.4.0, 12.3.0, and 13.2.0.

If you are cross-compiling, you need a cross-toolchain compiler that targets AArch64. You can download open-source cross-toolchain builds of the GCC compiler on the Arm Developer website.
The variant to use for an AArch64 GNU/Linux target is `aarch64-none-linux-gnu`.

- CMake version 3.3.0 or higher.

Additionally:

- To run the benchmarks, you must have the Linux utility tool `perf` installed and a recent version of Python 3. ArmRAL has been tested with Python 3.8.5.
- To build a local version of the documentation, you must have Doxygen installed. ArmRAL has been tested with Doxygen version 1.8.17.
- To generate code coverage HTML pages, you must have `gcovr` installed. The library has been tested with `gcovr` version 4.2.

ArmRAL runs on AArch64 cores, however to use the convolutional encoder, CRC, and sequence generator functions, you must run on a core that supports the AArch64 PMULL extension. If your machine supports the PMULL extension, `pmull` is listed under the "Features" list given in the `/proc/cpuinfo` file.

4 Release history

A full release history (with release notes) for ArmRAL is available on the Arm Developer website.

5 Conventions

The following subsections describe conventions used in Arm documents.

5.1 Glossary

The Arm Glossary is a list of terms that are used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the **Arm Glossary** for more information.