TASKING®

TASKING Performance Libraries For Infineon AURIX MCUs

L A P A C K

L -A P -A C -K

L A P A -C -K

L -A P -A -C K

L A -P -A C K

L -A -P A C -K

TABLE OF CONTENTS

LAPACK AND BLAS LIBRARIES — HIGH LEVEL OVERVIEW

The TASKING® LAPACK Performance Libraries provides the Linear Algebra PACKage (LAPACK) and the Basic Linear Algebra Subroutines (BLAS) in the form of a highly-optimized and highly-tested ISO-C99 compliant source code library.

Linear algebra is a central theme in mathematics. It concerns linear equations and their representations through matrices and vectors and is relevant for every engineering discipline. The mathematical complexity of today's Advanced Driver Assist Systems (ADAS) has raised computation to a level that is best served from a set of widely used, well tested, accurate, and fast de facto standard libraries such as the LAPACK and the BLAS. Application areas that benefit from the availability of such library include: Kinematic and dynamic vehicle modelling; Object detection and classification using either classic algorithms or artificial intelligence (AI); Sensor data fusion; Free space detection; Drive path planning; and others. The TASKING library eases the implementation of the automated driving functions that are specified in the Euro NCAP 2025 'In Pursuit of Vision Zero' roadmap.

The TASKING library provides a full implementation of the LAPACK and the BLAS for single precision arithmetic. It facilitates a seamless port of existing existing LAPACK/BLAS based software to embedded devices, and it is compatible with the output of the MathWorks MATLAB[1] and Simulink autocoders, which makes it easy to deploy the library in an existing software development flow.

The TASKING Performance Libraries for Tricore AURIX are highly-optimized for the Infineon TriCore AURIX and AURIX 2nd Generation microcontrollers, and have been tuned and verified using the TASKING TriCore Compiler. The floating point peak performance of AURIX 2G devices with 6 TriCore cores is 1.8GFlop/sec, and the inner loops of the BLAS functions typically perform at or close to 300MFlop/sec per core. The library has been created using ASPICE level-2 compliant processes, and the resulting code has been verified and validated using the de facto standard "LAPACK Test Suite". As such the library is suited for use in safety related systems up to ASIL-B.

The library comes with full C source code, pre-compiled binary files, makefiles, user documentation, and examples. The library can be (re)build with a TASKING VX-toolset for TriCore (not included), targeting AURIX and AURIX 2G devices, or can be build with any other ISO-C99 compliant compiler targeting any (embedded) device. In the latter case the performance may be suboptimal, and the result is not pre-verified.

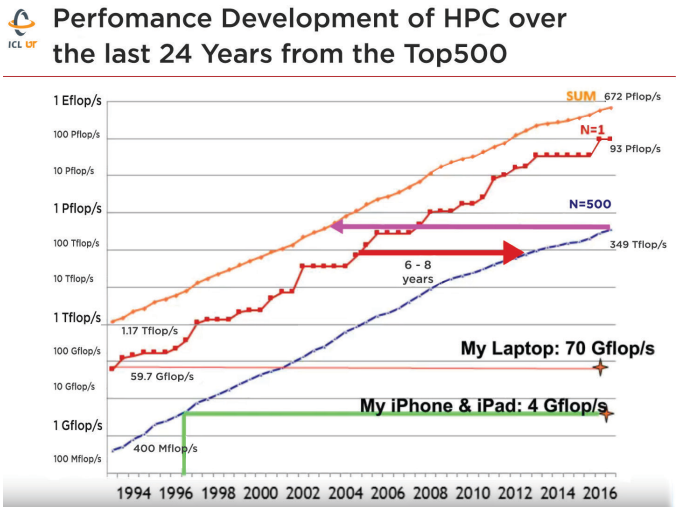
LAPACK AND BLAS LIBRARIES - DETAILED INTRODUCTION

Numerical Libraries for Embedded Applications

The mathematical complexity of embedded applications is rapidly rising. This trend is best witnessed by taking the cell phone as example. A modern cell phone can execute complex algorithms in real time such as recognizing faces and classifying image objects. It can do so because of its high computational performance.

[1]MATLAB started its life in the late 1970s as an interactive calculator built on top of LINPACK and EISPACK. LAPACK is the modern replacement for LINPACK and EISPACK, and the current versions of MATLAB are build on top of the LAPACK. Code generated by the MATLAB® and Simulink® autocoders (formerly know as Real-Time Workshop®) can be executed on embedded devices and use the LAPACK and the BLAS interfaces to call into device specific optimized implementations such as the TASKING LAPACK Performance Libraries.

The figure on the right shows the performance development of the top 500 supercomputers over the past 24 years. The orange line shows the accumulated performance of the fastest 500 computers; The red line shows the performance of the world's fastest computer, and the blue line shows the performance of the slowest computer in the top 500. The performance is measured by executing a set of linear algebra functions that are specifically optimized for execution on the computer being benchmarked. In the 1990s the LAPACK and the BLAS were used for this purpose[2]. The graph has been annotated with entries for today's laptops and cell phones, and is often used to forecast the performance trend of embedded computer systems. The performance of today's embedded devices match the performance of the supercomputers from the mid 1990s, which were predominantly based on a "multicore shared memory with local caches" architecture, which resembles the architecture of today's high-end embedded devices. This is what makes the LAPACK and the BLAS well suited for implementation on today's embedded devices.



The peak single precision floating point performance of the AURIX hardware is 600 MFlop/s which is achieved by executing a sequence of multiply-accumulate instructions, counting as two Flops each, without loads and stores. BLAS functions load operands from RAM, store the computed result back to RAM, and do not use multiply-accumulate instructions only. This limits the theoretical peak performance of the hardware to 300 MFlop/s.

The graphs below show the performance of some commonly used functions where the library was built with TASKING VX-toolset for TriCore v6.2r2 and measurements were done on a TC298TE processor at 300MHz. Results for other functions or other parameter values can be obtained from TASKING upon request.

In the automotive industry segment computational performance must be combined with functional safety. This affects both the hardware and the software.

Only a few microcontrollers are designed to support the higher ASILs, such as Infineon's TriCore AURIX and AURIX 2nd Generation microcontrollers.

There are many mathematical libraries but almost none of these are suited for use in safety critical embedded systems. A range of requirements need to be taken into consideration when selecting such library: ease-of-use, conformance to industry practices and standards, portability to the required device architecture(s), run-time performance, continued support from developers, and specialized optimization in code for specific application scenarios.

Under these constraints the LAPACK and the BLAS open source mathematical libraries look promising. This is a highly optimized numerical library designed for use on multicore shared memory systems with local caches. It is designed and implemented by the most renowned scientists and is maintained by various academic institutions, national laboratories and industry partners. It has been proven in use for many years, in application domains such as astrophysics, computational fluid dynamics, vibration analysis, noise reduction, data mining, deep learning, signal analysis, computer vision, and more.

---

[2]Successors of LAPACK/BLAS such as ScaLAPACK, PLASMA and MAGMA share the LAPACK and the BLAS interfaces, but the algorithms used and their implementations are optimized for subsequent supercomputer architectures suchs as parallel distributed memory machines, and GPU accelerated Multicore Architectures.

The library is a de facto industry standard, it is the basis of mathematical libraries such as the Intel Math Kernel Library (MKL), the ARM performance Libraries and the Mathworks MATLAB/Simulink software. However there is one problem: this library is only available in Fortran, a language that is typically not supported by compilers for embedded devices.

For all of the above reasons TASKING has taken the initiative to transfer the LAPACK and the BLAS source code from Fortran to ISO-C99 and make the result available as a library for use in embedded systems.

Open Source LAPACK and BLAS Functionality[3]

LAPACK is written in Fortran 90 and provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems. The associated matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur) are also provided, as are related computations such as reordering of the Schur factorizations and estimating condition numbers. Dense and banded matrices are handled, but not general sparse matrices. In all areas, similar functionality is provided for real and complex matrices, in both single and double precision.

There are three classes of LAPACK routines:

- Driver routines solve a complete problem, such as solving a system of linear equations or computing the eigenvalues of a real symmetric matrix. Users are encouraged to use a driver routine if there is one that meets their requirements.
- Computational routines, also called simply LAPACK routines, perform a distinct computational task, such as computing the LU decomposition of an m-by-n matrix or finding the eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the QR algorithm.
- Auxiliary routines are all the other subroutines called by the driver routines and computational routines.

The approach to achieving high efficiency is based on the use of a standard set of Basic Linear Algebra Subprograms, which can be optimized for each computing environment. Nearly all of the parallelism in the LAPACK routines is contained in the BLAS. LAPACK algorithms use block matrix operations, such as matrix multiplication, in the innermost loops. These block operations can be optimized to account for the memory hierarchy. LAPACK requires that highly optimized block matrix operations be already implemented on each machine.

The BLAS are routines that provide standard building blocks for performing basic vector and matrix operations. The Level 1 BLAS perform scalar, vector, and vector-vector operations, the Level 2 BLAS perform matrix-vector operations, and the Level 3 BLAS perform matrix-matrix operations.

Port to TriCore AURIX

The TASKING Performance Libraries for Tricore AURIX are based on the http://www.netlib.org Fortran version 3.7.0.

The Fortran code has been translated to ISO-C99 using a Fortran-to-C translator after which the code has been hand optimized. The ISO-C99 implementation is created in such a way that updates of the original Fortran version can easily be incorporated in future releases.

The LAPACK Performance Libraries are comprised of three separate physical libraries:

- The Linear Algebra PACKage (LAPACK).
  This is the top-level library that provides the user entry points which are build on top of the underlying Basic Linear Algebra Subprograms that perform most of the computations.

[3]As described on http://www.netlib.org/lapack and http://www.netlib.org/blas websites.

- The Basic Linear Algebra Subroutines (BLAS).
  This library contains functions that provide standard building blocks for performing basic vector and matrix operations.
- The Fortran-to-C Support Functions (F2C).
  The F2C library is a run-time library that provides a ISO-C99 implementations of various Fortran I/O and intrinsic functions.

The TASKING library is single threaded and supports all single precision LAPACK functions for real numbers utilizing the AURIX hardware floating point unit. AURIX devices do not have hardware support for double precision floating point arithmetic, therefore it is foreseen that future releases of the library for Tricore will not support 64-bit floating point. Support for complex numbers is scheduled for a future releases.

TASKING's implementation of the BLAS is highly optimized. For example the restrict keyword has been added to all pointer parameters of the BLAS function declarations, which notifies the compiler that the underlying data object is only accessed via that specific pointer. This enables the compiler to apply many optimizations that would otherwise be unsafe. To optimally exploit the memory bandwidth of the AURIX architecture the ratio between floating point instructions versus load/store instructions has been increased by using Load Double-word and Store Double-world instructions rather than multiple Load and Store Word instructions, loop constructs have been rewritten when needed to allow such optimization. Best performance is obtained when the code is compiled with TASKING VX-toolset for TriCore v6.2r2 and the performance of most BLAS functions is close to the maximum performance of the AURIX hardware which makes further tuning using handcrafted assembly superfluous.

The TASKING library is shipped with the C source files, makefiles, and precompiled object files, and can be (re)build with a TASKING VX-toolset for TriCore (not included). With the included makefiles the libraries will be built for AURIX (tc1.6.x) and AURIX 2G (tc1.6.2). The library can also be rebuild with any third party ISO-C99 compiler, targeting any (embedded) device.

An HTML-based API reference is included for easy access by means of a browser. The information is derived from Doxygen comments in the source files. There is also a C header file declaring all supported LAPACK and BLAS functions and data types for inclusion in application C modules.

Verification and Use in Safety Critical Systems

The Tricore AURIX port of the library and test suites was done using ASPICE level-2 compliant processes.

The netlib.org LAPACK and BLAS software is distributed with elaborate test suites and test instructions. The TASKING library has been verified utilizing these test suites and instructions. Version v1.0r2 of the library has been compiled with the TASKING VX-toolset for TriCore v6.2r2.

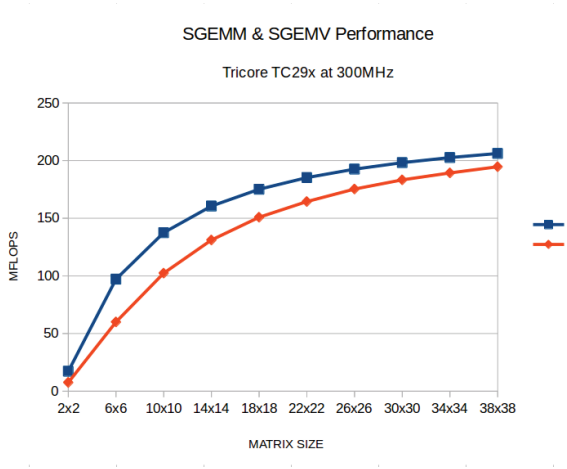
Performance data

The results for a selection of commonly used functions and input parameters are listed here. Results for other functions or other parameter values can be obtained from TASKING upon request.

Measurements were done on core 0 of a TC298TE processor. The execution speed results are listed in millions of floating point operations per second - MFlop/s. The maximum floating point performance that can be attained with the TC1.6P core at 300 MHz is 600 MFlop/s. This value corresponds to the execution of a continuous flow of floating point multiply-accumulate instructions (counting as two Flops each) without loads and stores. As BLAS functions need to load floating point operands from RAM and store the computed result back to RAM, and also since BLAS functions do not only use multiply-accumulate instructions, the maximum performance for BLAS functions is significantly less than the theoretical 600 MFlop/s and in best case limited to 300 MFlop/s.

Measurements were done using the TASKING C compiler for TriCore v6.2r2. Other compilers may produce different results. A binary version of the libraries built with the development version of the compiler can be obtained from TASKING upon request.

LAPACK Function	Parameters	TASKING Library v1.0r1	TASKING Library v1.0r2	Performance gained
sgemm	size=30x30, not transposed, α=1, β=0	134 MFlop/s	206 MFlop/s	+54%
sgemv	size=30x30, not transposed, α=1, β=1	131 MFlop/s	189 MFlop/s	+44%
sdot	size=85	140 MFlop/s	150 MFlop/s	+7%
ssymm	size=25x25, side=L, uplo=U, α=1, β=0	58 MFlop/s	107 MFlop/s	+84%



Licensing

Contact TASKING for licensing information.

Evaluate TASKING Library for Free

Please contact TASKING sales (support@tasking.com) if you want to evaluate the TASKING Performance Libraries for Tricore AURIX.

TASKING®

www.tasking.com