Gaisler Driver Library

GRDRV Product Overview

Product Overview

RELEASED JANUARY 2026

Frontgrade Gaisler AB | Kungsgatan 12 | SE-411 19 | Göteborg | Sweden | frontgrade.com/gaisler

Table of Contents

1. Introduction

The purpose of this document is to give an overview of the Gaisler Driver Library (GRDRV), highlighting its main features and capabilities.

This overview document is aimed at the GRDRV version for the GR765/NOEL-V. The GR765 ASIC is not yet available and this document will be updated when updating the software for the GR765 ASIC. This software has been developed has part of the "RISC-V Microprocessor Hardware Software Ecosystem" project, initiated and funded by the European Space Agency, contract `4000143337/23/NL/CRS`.

GRDRV has been developed according to ECSS standards [ECSS-E-ST-40C] and [ECSS-Q-ST-80C].

This document has been prepared by Frontgrade Gaisler AB, Göteborg, Sweden.

2. Applicable and reference documents

[ECSS-E-ST-40C]	ECSS - Space engineering, ECSS-E-ST-40C, 6 March 2009
[ECSS-Q-ST-80C]	ECSS - Space product assurance, ECSS-Q-ST-80C, 15 February 2017
[GRDRV-UM-1]	GRDRV User Manual, GRDRV-UM-1, Issue 2.0

3. Terms, definitions and abbreviated terms

AHB	Advanced High-performance Bus
ASIC	Application Specific Integrated Circuit
BC	Bus Controller
BM	Bus Monitor
BSP	Board Support Package
CAN	Controller Area Network
DMA	Direct Memory Access
ECSS	European Cooperation for Space Standardization
EDAC	Error Detection and Correction
FIFO	First In, First Out
FPGA	Field-Programmable Gate Array
FTADDR	Autonomous DDR2/DDR3 Controller with EDAC
GPIO	General-Purpose Input/Output
GR765	Radiation hardened fault-tolerant octa-core SPARC LEON and RISC-V microprocessor and embedded FPGA
GRLIB	IP Library from Frontgrade Gaisler
GRMON	Hardware monitor LEON and NOEL-V systems
GRSPW2	SpaceWire implementation by Frontgrade Gaisler
ICD	Interface Control Document
PROM	Programmable read-only memory
RT	Remote Terminal
RTEMS	Open source Real Time Operating System (RTOS)
SMP	Symmetric multiprocessing
SPI	Serial Peripheral Interface
SRS	Software Requirement Specification
SSS	Software System Specification
SoC	System on Chip
SpaceFibre	Successor of SpaceWire with higher data rate
SpaceWire	A standard governing serial communication between satellite components
UART	Universal Asynchronous Receiver/Transmitter

4. Delivery Overview

This chapter provides an overview of the delivery contents for the GRDRV software package.

4.1. Software Contents

The GRDRV software package is delivered as a collection of source code files, header files, unit & validation tests, samples and build scripts. The main components of the delivery include:

Table 4.1. GRDRV Software Delivery Contents

Component	Description
Source Code	The complete source code for the GRDRV library, organized into modules and components.
Header Files	Public and private header files defining the interfaces and data structures used by the GRDRV library.
Unit Tests	A suite of unit tests to verify the functionality and correctness of the software components.
Timing tests	Performance measurement tests to evaluate the timing characteristics of the software components.
Validation Tests	A set of validation tests to ensure the library meets the specified requirements.
Build Scripts	Makefiles and processing scripts to facilitate the compilation of the library and associated tests as well as test and coverage report generation.
Examples	RTEMS sample applications demonstrating the usage of the GRDRV library.

4.2. Documentation Contents

GRDRV is delivered with ECSS documentation covering software requirements, specifications, design, and validation. The list of documents is as follows:

Table 4.2. GRDRV Documentation Delivery Contents

ECSS Name	ECSS Acronym	Description
Software System Specification	SSS	High-level description of the GRDRV software. Also describres the system requirements.
Software Design Document	SDD	Description of GRDRV architectural design and the software detailed design.
Software Requirement Specification	SRS	Describes the functional and non-functional requirements applicable to GRDRV.
Interface Control Document	ICD	Describes all the external interfaces.
Software Unit/Integration Test Plan	SUITP	Plans for unit testing of the GRDRV software components.
Software Validation Plan	SValP	Describes the approach to the implementation of the validation process for GRDRV.
Software Validation Specification	SVS	Describes the testing, analysis, inspection and review of design specifications.
Software Unit/Integration Test Report	SUITR	Results and analysis of the unit tests.
Software Validation Test Report	SVTR	Results and analysis of the validation tests.
Software User Manual	SUM	Guide for users on how to utilize the GRDRV library.
Software Release Document	SRelD	Summary of the software release, including changes compared to previous releases, known problems, limitations or restrictions with respect to its approved baseline.

5. Software Overview

GRDRV is a library of drivers for GRLIB peripherals present in the GR765/NOEL-V SoC. The drivers are written in C and implemented in standard C11 while also following a tailored version of the MISRA C:2012 guidelines for safety-critical software development.

While the current version of the library is targeted for the GR765 in NOEL-V mode, it is designed to be minimally dependent on the underlying hardware or RTOS (accessed through a run-time abstraction layer), making it easily portable to other platforms with similar peripherals.

Drivers for the following peripherals are provided:

- AHB Status Register: Driver for logging correctable and non-correctable error responses.
- APB UART: Driver for transfering arbitrarily sized data buffers over UART.
- CAN-FD Controller: Driver for CAN controller with Flexible Data-rate support.
- Clock-gate Unit: Driver to manage clock gated cores to reduce system power consumption.
- General Purpose I/O Port: Driver for general purpose input/output (GPIO) pins.
- FTADDR Memory Controller: Driver for accessing memory scrubbing and error counters to help identifying memory errors.
- MIL-STD-1553B Controller: Driver for the MIL-STD-1553B data bus, including bus controller (BC), remote terminal (RT), and bus monitor (BM) roles.
- SpaceFibre Controller: Driver for SpaceFibre communication with virtual channel and broadcast channel support.
- SpaceWire DMA Controller: Driver for SpaceWire communication via the SpaceWire router.
- SpaceWire Router: Driver for configuring the SpaceWire router.
- SPI Master Controller: Driver for a Serial Peripheral Interface (SPI) master controller with standard, dual, and quad support.
- Watchdog: Driver used to kick watchdog and handle graceful shutdown.

5.1. Key Features

- Developed in accordance with European Space Agency software engineering standards [ECSS-E-ST-40C] and [ECSS-Q-ST-80C], tailored software criticality category B.
- RTEMS compliant with minimal reliance on operating system support.
- All operations are non-blocking.
- Interrupt operation is supported and optional.

5.2. Software Licensing

A software license for GRDRV can be acquired from Frontgrade Gaisler. The license includes the software in source code, unit tests, detailed documentation on requirements, specifications, implementation and support hours.

For more information on licensing options, contact sales@gaisler.com [mailto:sales@gaisler.com].

5.3. Board Support

Currently, GRDRV is supported on any NOEL-V based platform, including the GR765 FPGA development boards. See Chapter 8 for more details.

5.4. Operating System Support

GRDRV is designed to be portable across different real-time operating systems (RTOS) through a run-time abstraction layer. The current version has been validated on top of RTEMS 6.2 SMP with a NOEL64IMAFD BSP.

5.5. Limitations

At the time of this release, the GR765 target hardware is still in development and device address mapping may change before the final release. This will require updating the descriptors provided to the drivers. All validation testing has been performed on FPGAs.

Another limitation is that the SPI controller implementation in the FPGA loses two data bits for every transferred word when using it in loopback mode in dual or quad configuration. The validation test can detect this and ignore

those two bits. A message is generated in the test report to state this fact. When the problem has been solved the validation test will automatically start testing all bits in the words.

6. Software unit testing

GRDRV is delivered with a comprehensive unit testsuite that validates the functionality of each driver in isolation using simulated registers.

The validation testsuite requires RTEMS 6 SMP with a NOEL64IMAFD BSP running on on simulated hardware, either on TSIM3-GR765, QEMU, or on the target hardware using GRMON 4. See Chapter 8 for more details.

The modular design of the drivers allows for unit-testing each drivers independently from the rest of the library. An automated test framework built on top of the RTEMS Test Framework [https://docs.rtems.org/docs/main/eng/test-framework.html] is used to execute the tests and generate reports on test results and code coverage.

The coverage reports contain source code annotated with coverage information for each line and branch in the code, making it easy to identify untested parts of the code.

As of this release, a code coverage of 100% has been achieved for all drivers in the library.

Timing tests are also included in the unit testsuite to measure the execution time of all software components.

7. Software validation testing

GRDRV is delivered with a comprehensive validation testsuite that is aimed at validating requirements for every drivers.

The validation testsuite requires RTEMS 6 SMP with a NOEL64IMAFD BSP running on a NOEL-V based platform with the peripherals connected in loopback mode where applicable. See Section 7.1 and Chapter 8 for more details.

As of this release, a 100% requirement coverage has been achieved for all drivers in the library.

7.1. Required hardware configuration

The following chapter describes the required hardware configuration for each driver to be able to run their validation tests.

Each driver validation test is self contained and can be run independently from the others, meaning that only the hardware configuration required for the specific driver under test is necessary.

7.1.1. AHB Status Register

Access to a PROM memory with EDAC support is required to be able to trigger correctable and non-correctable error responses.

7.1.2. APB UART

Two UART peripherals must be connected in loopback.

7.1.3. CAN-FD Controller

Two CAN-FD controllers must be connected in loopback.

7.1.4. Clock-gate Unit

A GRDMAC2 peripheral is required to test the clock-gating functionality.

7.1.5. General Purpose I/O Port

The GPIO pins should be connected as follows, starting from Pin[2]: `Pin[n] <-> Pin[n+1]`

7.1.6. FTADDR Memory Controller

A L2 cache big enough to fit the full test binary must be present and enabled to be able to run the memory scrubbing and error counter tests.

A GRDMAC2 peripheral and an AHBRAM are also required to test memory initialization.

7.1.7. MIL-STD-1553B Controller

The test requires two GR1553B controllers connected together.

7.1.8. SpaceFibre Controller

The test requires that two SpaceFibre controllers (GRSPFI0 and GRSPFI1) are connected and that GRSPFI0 has a FIFO connection to a SpaceWire router. The devices need to support at least three virtual channels.

7.1.9. SpaceWire DMA Controller

The test requires two SpaceWire DMA controllers, GRSPW0 and GRSPW1, connected to a SpaceWire router.

7.1.10. SpaceWire Router

The test requires three SpaceWire DMA controllers, GRSPW0, GRSPW1 and GRSPW2 and two ports (Port 1 and Port 8) on the SpaceWire router connected either internally or externally.

7.1.11. SPI Master Controller

The test requires that the two SPI controllers SPI0 and SPI1 are connected together and that the first slave signal from SPI0 is connected to the slave input of SPI1.

7.1.12. Watchdog

Two GPIO peripherals are be required.

The WDOG signal needs to be connected to GPIO0 pin 0. The WDOG reset signal needs to be connected to GPIO1 pin 1.

8. Equipment requirements

This chapter is a summary of the tools and equipment needed to use and develop with GRDRV. The [GR-DRV-UM-1] describes how to use the tools for building and testing.

8.1. Building the target software

For building the software, a computer equipped with GNU/Linux is supported. The following environments have been tested:

- OpenSuse Leap 15.5 & 15.6
- Ubuntu 24.04 LTS

The following software tools are required to compile and use the drivers:

- RTEMS 6 RISC-V Cross Compiler built with RTEMS Source Builder 6.2
- RTEMS 6 SMP noel64imafd BSP kernel built with RTEMS 6.2
- GNU Make
- LCOV 2.0-1
- Python >= 3.6 along with packages:
  - `clang`
  - `lcov_cobertura`

These tools are also necessary to run the unit tests, generate code coverage reports and run the validation testsuite.

8.2. Running the unit tests

The unit testsuite uses simulated registers to validate the driver functionalities. Therefore, it is not necessary to use the target hardware to run the tests.

The unit tests can be run either on TSIM3-GR765 [https://www.gaisler.com/products/tsim3-gr765] or on QEMU [https://www.qemu.org] as well as on the target hardware, using GRMON 4.

The unit testsuite also comes with a timing test for each software component. It is recommended to run these tests on the target hardware to get accurate timing measurements.

8.3. Running the validation tests

The validation testsuite requires the hardware to be present and, in the case of communication peripherals, connected in loopback mode. See Section 7.1 for more details on the required hardware setup for each driver.

The validation testsuite has been developed and tested on the GR-CPCIS-XCKU [https://www.gaisler.com/products/gr-cpcis-xcku] development board programmed with a prototype GR765 NOEL-V system.

GRMON 4 is necessary to load and run the validation tests on the target hardware.

Frontgrade Gaisler AB
Kungsgatan 12
411 19 Göteborg
Sweden
frontgrade.com/gaisler
sales@gaisler.com
T: +46 31 7758650
F: +46 31 421407