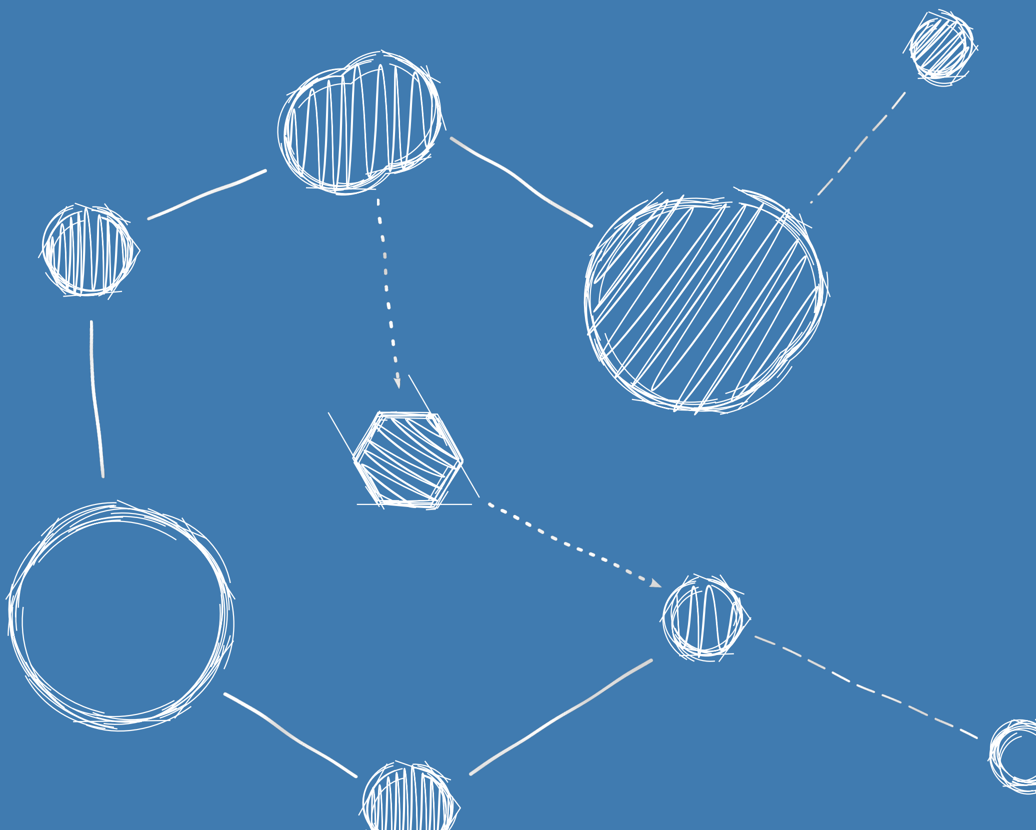
institut für astrophysik
UNIVERSITÄTSSTERNWARTE WIEN

universität wien

LEAN OS – AN OPERATING SYSTEM FOR THE SSDP

Test Plan



Test Plan

Reference: LEANOS-UVIE-TP-001

Version: Issue 1.0, June 1, 2017

Prepared by: Armin Luntzer[1]

Checked by: Roland Ottensamer[1], Christian Reimers[1]

Approved by: Franz Kerschbaum[1]

[1] Department of Astrophysics, University of Vienna

Contents

List of Requirements

Revision History

Revision	Date	Author(s)	Description
0.1	23.07.2017	AL	initial version
0.2	01.07.2016	AL	incorporate changes from review items
1.0	01.06.2017	FK	document approved

1. Introduction

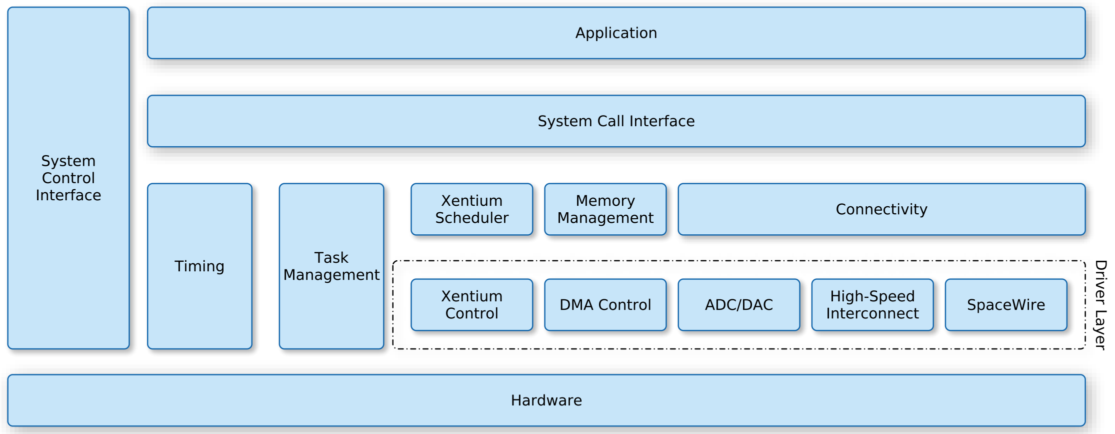


Figure 1.1: The logical model of LeanOS. Here, the "hardware" layer represents both the hardware and the hardware abstraction layer of the software.

LeanOS[3][1][2] is an operating system specifically designed to support hardware devices with drivers and data processing on the Xentium Digital Signal Processors (DSPs) of the Scalable Sensor Data Processor (SSDP).

In LeanOS, the SSDP hardware is accessed in multiple layers of abstraction (see Figure 1.1). Typical Central Processing Unit (CPU) tasks such as thread/task management and timer operation are used as part of the operating system kernel and are also accessible by user applications via a system call interface. Other functional hardware components of the SSDP such as the Network On Chip (NoC) Direct Memory Access (DMA) have their own driver modules. These are in turn used by the Xentium scheduler and other higher level modules in the operating system. Configuration of and access to the latter from user space is done via a system call interface. The system control interface serves as an intermediate between all layers of the operating system, where system or module states and hardware modes or usage statistics may be exported by individual components for external (user) access.

The compliance of LeanOS with its requirements and its operational environment are partially verified by testing.

1.1 Purpose of the Document

This document defines test procedures for the LeanOS operating system. In particular, it defines the:

- types of tests
- environment in which a test is executed
- coverage of the test
- entity responsible for performing the test

1.2 Incremental Testing

LeanOS is not envisaged to be implemented in one final form. Instead, it is marked with version numbers for each release. This means that testing is version-dependent and is be applied in increments by executing a sub-set of tests only as modifications are made.

1.3 Test Sequences

Tests defined in this document are executed by automated test sequences in the form of software programs that control the execution of a test, check the correctness of the results and generate a summary report of the test case.

1.4 Items Under Test

Components and software modules of the operating system are tested on an unit-level where applicable. System level tests are tailored to the various configuration options available. and shall be accompanied by configuration files so the proper build may generated.

R-IUT-0001	**Short Text**	**Software Requirement**
	Hardware constraints	The test plan specified by the present requirements shall apply to all parts of LeanOS.

1.5 Operational Constraints

Each test shall specify the minimum hardware platform on which it must be performed. Unit-level test of platform-independent components may be executed on any compatible hardware. Platform-dependent tests may require specific hardware models (e.g. the SSDP for testing of the

NoC) or just any compatible processor model (e.g. LEON3 for testing the operation of the Memory Management Unit (MMU)).

R-OPC-0001	Short Text	Software Requirement
	Hardware constraints	The execution of the tests defined in this document shall comply with all the operational constraints applicable to the hardware used in the tests.

2. Applicable and Reference Documents

[1] *LeanOS Architectural Design Document*. 2017.

[2] *LeanOS Software Requirements Specification*. 2017.

[3] Armin Luntzer et al. *A Lightweight Operating System for the SSDP*. 2016.
URL: `https://indico.esa.int/indico/event/102/session/28/contribution/26`.

3. Terms, Definitions and Abbreviated Items

3.1 Acronyms

CPU	Central Processing Unit
DMA	Direct Memory Access
DSP	Digital Signal Processor
ILP	Instruction Level Parallelism
MMU	Memory Management Unit
MPPB	Massively Parallel Processor Breadboarding system
NoC	Network On Chip
RAM	Random-Access Memory
SMP	Symmetric Multiprocessing
SoC	System On Chip
SSDP	Scalable Sensor Data Processor
VLIW	Very Long Instruction Word

3.2 Glossary

Central Processing Unit (CPU)

The Central Processing Unit is the electronic circuitry that interprets instructions of a computer program and performs control logic, arithmetic, and input/output operations specified by the instructions. It maintains high-level control of peripheral components, such as memory and other devices.

Digital Signal Processor (DSP)

A Digital Signal Processor is a specialised processor with its architecture targeting the operational needs of digital signal processing.

Direct Memory Access (DMA)

Direct Memory Access is a feature of a computer system that allows hardware subsystems to access main system Random-Access Memory (Random-Access Memory (RAM)) directly, thereby bypassing the Central Processing Unit (CPU).

Instruction Level Parallelism (ILP)

Instruction-level parallelism (ILP) is a measure of how many instructions in a computer program can be executed simultaneously by the CPU.

LEON2

The LEON2 is a synthesisable VHDL model of a 32-bit processor compliant with the SPARC V8 architecture. It is highly configurable and particularly suitable for System On Chip (SoC) designs. Its source code is available under the GNU LGPL license

LEON3

The LEON3 is an updated version of the LEON2, changes include Symmetric Multiprocessing (Symmetric Multiprocessing (SMP)) support and a deeper instruction pipeline

LEON3-FT

The LEON3-FT is a fault-tolerant version of the LEON3. Changes to the base version include autonomous error handling, cache locking and different cache replacement strategies.

Massively Parallel Processor Breadboarding system (MPPB)

The Massively Parallel Processor Breadboarding system is a proof-of-concept design for a space-hardened, fault-tolerant multi-DSP system with various subsystems to build a powerful digital signal processing system with a high data throughput. Its distinguishing features are the Network On Chip (NoC) and the Xentium DSPs controlled by a LEON2 processor. It was developed under ESA contract 21986 by Recore Systems B.V.

Memory Management Unit (MMU)

A Memory Management Unit performs address space translation between physical and virtual memory pages and protects unprivileged access to certain memory regions.

Network On Chip (NoC)

A Network On Chip is a communication system on an integrated circuit that applies (packet based) networking to on-chip communication. It offers improvements over more conventional bus interconnects and is more scalable and power efficient in complex System On Chip (SoC) desgins.

Random-Access Memory (RAM)

Random-Access Memory is a type of memory where each memory cell may be accessed directly via their memory addresses.

Scalable Sensor Data Processor (SSDP)

The Scalable Sensor Data Processor (SSDP) is a next generation on-board data processing mixed-signal ASIC, envisaged to be used in future scientific payloads requiring high-performance on-board processing capabilities. It is built opon a heterogeneous multicore architecture, combining two Xentium DSP cores with a general-purpose LEON3-FT control processor in a Network On Chip (NoC).

Symmetric Multiprocessing (SMP)

Symmetric Multiprocessing denotes computer architectures, where two or more identical processors are connected to the same periphery and are controlled by the same operating system instance.

System On Chip (SoC)

A System On Chip is an integrated circuit that combines all components of a computer or other electronic system into a single chip.

Very Long Instruction Word (VLIW)

Very Long Instruction Word is a processor architecture design concept that exploits Instruction Level Parallelism (ILP). This approach allows higher performance at a smaller silicone footprint compared to serialised instruction processors, as no instruction re-ordering logic to exploit superscalar capabilities of the processor must be integrated on the chip, but requires either code to be tuned manually or a very sophisticated compiler to exploit the full potential of the processor.

Xentium

The Xentium is a high performance Very Long Instruction Word (VLIW) DSP core. It operates 10 parallel execution slots supporting 32/40 bit scalar and two 16-bit element vector operations.

4. Types of Tests

In order to satisfy the verification and validation needs of the test campaign, the following types of tests are defined for LeanOS:

- unit and integration tests
- hardware-software tests

4.1 Unit and Integration Tests

LeanOS is developed as a collection of individual, interacting components. A unit test verifies the behaviour of a single component, while an integration test verifies the behaviour of a number of interacting components.

Unit and integration tests are white-box type tests that executes a component with a series of input conditions and verify the output or action for the expected response. These type of tests are typically independent from the hardware platform.

R-UIT-0001	**Short Text**	**Software Requirement**
	Unit/Integ. Tests	Unit and Integration Tests shall be defined and executed for LeanOS components.
Comment	Tests can make use of the modified *kselftest* framework provided in the LeanOS source tree.	

R-UIT-0002	**Short Text**	**Software Requirement**
	Test Suite	A unit and integration test suite shall be provided as a set of applications.

R-UIT-0003	**Short Text**	**Software Requirement**
	Automatic Test Execution	A single mechanism shall be provided that automatically runs all unit and integration tests in sequence.
Comment	The *kselftest* framework in the LeanOS source tree provides a *Makefile* setup that can be used for this purpose.	

R-UIT-0004	Short Text	Software Requirement
	Test Results	The test suite shall generate feedback as a human-readable report describing the outcome of each unit and integration test.

R-UIT-0005	Short Text	Software Requirement
	Automatic Testing	A feedback mechanism shall be provided that allows an external automated testing mechanism to execute and detect the results of unit and integration tests.

Comment	This is very useful for automated bisection of code versions, e.g. by using the *git bisect run* … functionality.

4.2 Hardware-Software Tests

These tests differ from unit and integration tests in that a particular hardware platform is required for testing. All system-level integration tests fall into this category.

R-HST-0001	Short Text	Software Requirement
	Hardware Tests	Hardware-software tests shall be defined and executed for LeanOS components with the objective of verifying the behaviour of components which directly interact with the relevant target hardware.

R-HST-0002	Short Text	Software Requirement
	Test Suite	A hardware-software test suite shall be provided as a set of applications.

R-HST-0003	Short Text	Software Requirement
	Test Results	The hardware-software tests shall generate feedback as a human-readable report describing the outcome of test.

R-HST-0004	Short Text	Software Requirement
	Automatic Testing	A feedback mechanism shall be provided that allows an external automated testing mechanism to execute and detect the results of hardware-software tests.

Comment	This is very useful for automated bisection of code versions, e.g. by using the *git bisect run* … functionality.

4.3 Acceptance Tests

Acceptance tests are performed by the user on the integrated processing unit. Their objective is to exercise the selected LeanOS configuration while running in its operational environment. The operational environment for LeanOS is defined by the user according to their use case.

R-ACT-0001	Short Text	Software Requirement
	Acceptance Tests	Acceptance tests and their procedures shall be defined and executed for LeanOS.

R-ACT-0002	Short Text	Software Requirement
	Use Cases	The configuration and operational environment for an acceptance test of LeanOS shall be defined by the user according to their use case.

5. Test Environments

Test beds in which LeanOS tests are performed:

- general-purpose platform for regular unit and integration tests
- LEON2 and/or LEON3 platforms for hardware-software tests
- SSDP and/or MPPB for system-level hardware-software tests

The test environments required for each category of test are described in the following sections.

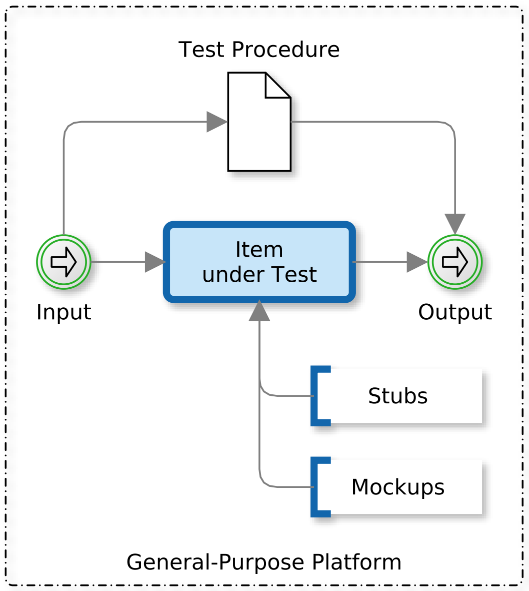


Figure 5.1: Unit and integration test environment. Stubs and Mockups are stand-ins for external dependencies with defined behaviour. A set of input data is fed into the item under test as specified in the test procedure.

5.1 Unit and Integration Test Environment

The test environment for unit and integration tests is shown in Figure 5.1.

The item under test is either one or a set of components of LeanOS. If needed, stub and mockup functions, which simulate the expected behaviour of external dependcies, are defined to create a representative environment in which the item under test is running.

Tests are controlled by a test sequence, which generates inputs for the item under test and collects and evaluates the generated output.

R-UITE-0001	Short Text	Software Requirement
	Unit Test Environment	Unit and integration tests shall be performed in an environment as defined in Figure 5.1.

5.2 Hardware-Software Test Environment

The hardware-software test environments are detailed on a per-test basis.

R-HSTE-0001	Short Text	Software Requirement
	HW/SW Test Environment	The required hardware-software test environment shall be defined in the specification of each individual test.

6. Test Coverage

This chapter defines the coverage of tests in the environments defined previously.

6.1 Source Code Coverage

Source code coverage must be achieved in unit and integration tests. A component is considered covered if full statement, decision and condition coverage is achieved.

If full coverage cannot be achieved, for example due to operational constraints, deviations are acceptable, provided a justification is given.

R-SCC-0001	**Short Text**	**Software Requirement**
	Statement Coverage	Unit and integration tests shall offer full statement coverage for LeanOS code.

R-SCC-0002	**Short Text**	**Software Requirement**
	Decision Coverage	Unit and integration tests shall offer full decision coverage for LeanOS code.

R-SCC-0003	**Short Text**	**Software Requirement**
	Condition Coverage	Unit and integration tests shall offer full condition coverage for LeanOS code.

R-SCC-0004	**Short Text**	**Software Requirement**
	Reduced Coverage	If the coverage specified in the previous requirements cannot be achieved due to operational contraints, justifications for deviations shall be provided for each of the individual conditions where the degree of coverage is below the specified goal.