

# ARINC 653 Conformance Document

---

LynxOS-178

DOC-2205-00



Product names mentioned in this document are trademarks of their respective manufacturers and are used here for identification purposes only.

© 2015 - 2019 Lynx Software Technologies, Inc. All rights reserved.

© 2004 - 2014 LynxWorks, Inc. All rights reserved.

U.S. Patents 5,469,571; 5,594,903; 6,075,939; 7,047,521

Printed in the United States of America.

All rights reserved. No part of the document may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photographic, magnetic, or otherwise, without the prior written permission of Lynx Software Technologies, Inc.

Lynx Software Technologies, Inc. makes no representations, express or implied, with respect to this documentation or the software it describes, including (with no limitation) any implied warranties of utility or fitness for any particular purpose; all such warranties are expressly disclaimed. Neither Lynx Software Technologies, Inc., nor its distributors, nor its dealers shall be liable for any indirect, incidental, or consequential damages under any circumstances.

(The exclusion of implied warranties may not apply in all cases under some statutes, and thus the above exclusion may not apply. This warranty provides the purchaser with specific legal rights. There may be other purchaser rights which vary from state to state within the United States of America.)

# Table of Contents

<b>PREFACE</b> .....	<b>III</b>
Typographical Conventions .....	iii
Technical Support.....	iv
How to Submit a Support Request .....	iv
Where to Submit a Support Request.....	iv
<b>OVERVIEW</b> .....	<b>1</b>
Introduction .....	1
General .....	1
Implementation Conformance .....	1
Definitions.....	1
Implementation Structure.....	7
Issues with Mixing POSIX and ARINC .....	7
Implementation-Specific Details.....	8
File System Naming Details.....	8
File System Features Limitation .....	9
Configuration .....	9
Installing ARINC 653 to LynxOS-178.....	10
Static Installation of ARINC 653 Support .....	10
Dynamic Installation of ARINC 653 Support.....	11
Configuring LynxOS-178 for ARINC 653 Support .....	11
Configuring the Kernel .....	12
Service Requirements.....	16
Service Request Categories (P1:3.1, P2:3.1).....	16
Return Code Data Type (P1:3.1.1, P2:3.1.1).....	17
Partition Management Services.....	17
Process Management Services .....	19
Time Management Services .....	31
Inter-partition Communication Services .....	34
Intra-partition Communication Services .....	48
Blackboard Services (P1:3.7.2.2).....	54
Semaphore Services (P1:3.7.2.3) .....	59
Event Services (P1:3.7.2.4).....	64
Memory Blocks (P2:3.9).....	68

Health Monitoring (P1:3.8).....69

File System Services (P2:3.2.5) .....72

---

# *Preface*

---

## Typographical Conventions

The typefaces used in this manual, summarized below, emphasize important concepts. All references to filenames and commands are case-sensitive and should be typed accurately.

Kind of Text	Examples
Body text; <i>italicized</i> for emphasis, new terms, and book titles	Refer to the <i>ARINC 653 Conformance Document</i>
Environment variables, filenames, functions, methods, options, parameter names, path names, commands, and computer data	<code>ls -l myprog.c /dev/null</code>
Commands that need to be highlighted within body text or commands that must be typed as is by the user are <b>bolded</b> .	<code>login: <b>myname</b></code> <code># <b>cd /usr/home</b></code>
Text that represents a variable, such as a filename or a value that must be entered by the user, is <i>italicized</i> .	<code>cat &lt;filename&gt;</code> <code>mv &lt;file1&gt; &lt;file2&gt;</code>
Blocks of text that appear on the display screen after entering instructions or commands	Loading file /tftpboot/shell.kdi into 0x4000 ..... File loaded. Size is 1314816  © 2015 Lynx Software Technologies, Inc. All rights reserved.
Keyboard options, button names, and menu sequences	<b>Enter, Ctrl-C</b>

---

## Technical Support

Lynx Software Technologies handles support requests from current support subscribers. For questions regarding Lynx Software Technologies products, evaluation CDs, or to become a support subscriber; our knowledgeable sales staff will be pleased to help you. Please visit us at:

<http://www.lynx.com/training-support/contact-support/>

### How to Submit a Support Request

When you are ready to submit a support request, please include *all* of the following information:

- First name, last name, your job title
- Phone number, e-mail address
- Company name, address
- Product version number
- Target platform (for example, PowerPC)
- Board Support Package (BSP), Current Service Pack Revision, Development Host OS version
- Detailed description of the problem that you are experiencing:
- Is there a requirement for a US Citizen or Green Card holder to work on this issue?
- Priority of the problem - Critical, High, Medium, or Low?

---

### Where to Submit a Support Request

Support, Europe	tech_europe@lynx.com +33 1 30 85 93 96
Support, worldwide except Europe	support@lynx.com +1 800-327-5969 or +1 408-979-3940 +81 33 449 3131 [for Japan]
Training and Courses	USA: training-usa@lynx.com Europe: training-europe@lynx.com USA: +1 408-979-4353 Europe: +33 1 30 85 06 00

---

# Overview

---

## Introduction

This document states the conformance of LynxOS-178 to the ARINC 653 APEX Interface defined by the ARINC 653-1 standard and a subset of ARINC 653-2 standard. It describes the functionality of individual ARINC 653-1 and 653-2 services and the deviations from the ARINC 653-1 and 653-2 standards.

Sections of this document that have counterparts in the ARINC 653-1 and 653-2 standards are accompanied with the corresponding section numbers with prefixes P1 or P2 respectively.

Throughout this document, the term ARINC 653-1 is used as an abbreviation for the *ARINC Specification 653P1-3: Avionics Application Standard Software Interface Part1 – Required Services*, published November 15, 2010 and the term ARINC 653-2 is used as an abbreviation for the *ARINC Specification 653P2-2: Avionics Application Standard Software Interface Part2 – Extended Services*, published June 29, 2012. The term ARINC 653 is used to refer to both ARINC 653-1 and 653-2 specifications.

---

## General

### Implementation Conformance

#### Definitions

#### APEX Interface

Application/Executive Interface, the term used to designate the OS services in the ARINC 653-1 and 653-2 standards.

## ARINC

Aeronautical Radio, Inc., an organization issuing the specifications for the airline electronic equipment.

## BSP

Board Support Package.

## FIFO

First-In, First-Out; the order of request servicing in which first-come is first-served (as opposed to the priority-based servicing).

## HM

Health Monitoring.

## KDI

Kernel Downloadable Image.

## OS

Operating system.

## PID

Process Identifier, an integer number uniquely identifying a process within the operating system.

## VCT

Virtual machine Configuration Table.

## VM

Virtual Machine.

## Aperiodic

Occurs predictably, but not at regular time intervals.



## Application

That software consisting of tasks or processes that perform a specific function on the aircraft.

An application may be composed of one or more partitions.

## Application Partition

An ARINC 653-1-compliant partition.

## Blackboard

Message-based intra-partition communication service in which every new message overwrites the previous one.

## Buffer

Message-based intra-partition communication service in which messages are queued.

## Conformance

Providing all services and characteristics described in a specification or standard.

## Deadline

A time by which a process must have completed a certain activity.

## Debug

The process of locating, analyzing, and correcting suspected errors (ARINC Report 652).

## Default

A value or state that is used when contrary values or actions are not available.

## Directory

Directory is a Filesystem cataloging structure which contains references to other computer files, and possibly other directories. Files are organized by storing related files in the same directory.

## Dormant

A process that is available to execute, but is not currently executing or waiting to execute.

## Error

- With respect to software, a mistake in requirements, design, or code.
- An undesired system state that exists either at the boundary or at an internal point in the system; it may be experienced by the user as failure when it is manifested at the boundary. For software, it is the programmer action or omission that results in a fault.

## Event

Semaphore-based intra-partition communication service having two states: UP and DOWN. A process can set either of these states or wait for the event to change into the UP state.

## Failure

The inability of a system or system component to perform a required function within specified limits. A failure may be produced when a fault is encountered.

## File

File is a resource for storing information, which is available to a computer program and is based on a durable storage. Files are organized into

one-dimensional arrays of bytes associated with control information (such as the name or size of file).

## Filesystem (FS)

Filesystem The file system is a general purpose, abstract way of managing data storage. It is used to control how data is stored and retrieved.

## I-node

An index node, informally referred to as an i-node, is a data structure used to represent a filesystem object, which can be one of various things including a file or a directory. Each i-node stores the attributes and disk block location(s) of the filesystem object's data.

### Inter-partition

Refers to any communication conducted between partitions.

### Interrupt

A suspension of a task, such as the execution of a computer program, caused by an event external to that task and performed in such a way that the task can be resumed (ARINC Report 652).

### Intra-partition

Refers to any communication conducted between processes within the partition.

### Memory Block

Memory blocks provide a means for a partition to access blocks of memory that are within in the module's memory space.

### Message

A package of data transmitted between partitions, or between partitions and external entities.

Messages are sent and received by partitions via sampling and queuing port services.

### Partition

A group of processes contending for the dedicated (partitioned) set of resources, such as memory, CPU time, and so on.

### Periodic Process

A process that is marked as ready to run at fixed periods of time.

### Priority Queue

Queuing system in which entries that are assigned highest priority by their originators are processed first.

### Process

A programming unit contained within a partition that executes concurrently with other processes of the same partition. A process is the same as a task (ARINC Report 651).

### Queuing Port

Message-based inter-partition communication service in which messages are queued.

### Real Time Operating System (RTOS)

An operating system that satisfies the requirements for temporal and spatial partitioning.

### Release Point

The moment at which a periodic process is marked as ready to run.

### Sampling Port

Message-based inter-partition communication service in which every new message overwrites the previous one.

### Stack

Area of memory, allocated to a process, utilized on a last-in, first-out (LIFO) basis.

### Suspended

Process in waiting state. Execution has been temporarily halted awaiting completion of another activity or occurrence of an event.

### System Partition

A partition that requires interfaces outside the ARINC 653-1-defined services, but is still constrained by robust spatial and temporal partitioning. A system partition may perform functions such as managing communication from hardware devices or fault management schemes. System partitions are optional and are specific to the core module implementation.

## Task

A programming unit contained within a partition that executes concurrently with other processes of the same partition. A task is the same as a process (ARINC Report 651).

## Terminated

A process in dormant state. Execution has ended and cannot be resumed.

## Volume

A “volume” is an allocation of and/or access permissions for a portion of a storage device (e.g., RAM disk, FLASH). The volume name is a virtual representation that maps to an entry point of the physical storage device (e.g., disk or directory).

When used in different installations, the volume name may remain unchanged but the corresponding physical storage device may vary.

## Implementation Structure

The ARINC 653 APEX Interface to applications consists of the  
`$(ENV_PREFIX)/lib/libarinc653.a` library and the  
`$(ENV_PREFIX)/usr/include/arinc653/arinc653.h` header file.

## Issues with Mixing POSIX and ARINC

Note that the implementation of ARINC 653 service requests uses POSIX interfaces wherever appropriate. Moreover, behavior of some POSIX functions is adjusted to address ARINC 653 requirements. Therefore, the following warning is due:

```
MIXING ARINC653 AND POSIX INTERFACES IN THE ARINC  
APPLICATION MAY LEAD TO UNDEFINED BEHAVIOR.
```

Also, the process must exit only as a result of the `STOP/STOP_SELF` requests. Exiting by performing `return` from the `arinc653_main()` function will cause undefined behavior. POSIX applications may use only a restricted set of ARINC 653 APIs.

That said, it is possible to use ARINC 653 inter-partition communication interfaces from POSIX applications. To do so, the application shall invoke ARINC 653 API in a normal way.

## Implementation-Specific Details

The ARINC 653 standard defines the following constants as implementation-dependent:

Constant Name	Constant Value	Scope
SYSTEM_LIMIT_NUMBER_OF_PARTITIONS	32	Module Scope
SYSTEM_LIMIT_NUMBER_OF_MESSAGES	512	Module Scope
SYSTEM_LIMIT_MESSAGE_SIZE	8192	Module Scope
SYSTEM_LIMIT_NUMBER_OF_PROCESSES	128	Partition Scope
SYSTEM_LIMIT_NUMBER_OF_SAMPLING_PORTS	512	Partition Scope
SYSTEM_LIMIT_NUMBER_OF_QUEUEING_PORTS	512	Partition Scope
SYSTEM_LIMIT_NUMBER_OF_BUFFERS	256	Partition Scope
SYSTEM_LIMIT_NUMBER_OF_BLACKBOARDS	256	Partition Scope
SYSTEM_LIMIT_NUMBER_OF_SEMAPHORES	256	Partition Scope
SYSTEM_LIMIT_NUMBER_OF_EVENTS	256	Partition Scope
SYSTEM_LIMIT_FILE_SIZE	0x40000000	Module Scope

## File System Naming Details

The implementation of the ARINC 653-2 file system services relies on the POSIX file system implemented in LynxOS-178. Thus, ARINC 653-2 file system properties mostly leveraged from POSIX:

- The volumes are mounted on directories. There is no explicit ability to address a volume in the filename.
- The names of files and directories are case-sensitive.
- The delimiter between directories and file in the path is “/”.
- In addition to the delimiter and null character ('\0'), the following symbols are prohibited in the name of a file or directory: '#', '@', '!', '\$', '%', '&', '(', ')', '+', '=', ':', '}', '{', ']', '[', '~', '^', '\', ';', '>', '<', '?', '\"'
- The name of a directory or file shall be between 1 and 63 characters (excluding delimiters) and the path length shall be less than 511 characters.

## File System Features Limitation

In the current ARINC 653 implementation ARINC File System is implemented atop of existing LynxOS-178 filesystem. As the result the following features are not supported:

- Creation time (`CREATION_TIME`) is implemented using change time (`st_ctime`) feature of the Lynx-178 filesystem. Change time gets updated every time the size of file or other properties get changed.
- Number of changes of a file (`NB_OF_CHANGE`) and number of write errors (`NB_OF_WRITE_ERRORS`) are not supported as there is no corresponding functionality in LynxOS-178 filesystem.

## Configuration

The support for ARINC 653 APEX interfaces is implemented on top of the existing LynxOS-178 services. Therefore, the resource limits configured for a certain LynxOS-178 VM should not be more restrictive than the limits specified in the ARINC 653 device configuration file. This applies to the resources set in both the VCT and `uparam.h` files.

## VM 0 Issues

Due to the special nature of VM0 in LynxOS-178, the resource limits for it cannot be set in the VCT. Instead, its resource limits are set in the `uparam.h` file. By default, there are only 10 processes reserved for VM0. To allow more ARINC 653-1 processes to be created, either any other partition should be used or the default values should be increased. There are similar restrictions on other resources in VM0.

## Number of Threads and Processes

Concerning the limits on the number of threads and processes for a certain partition, please keep in mind that each partition needs 6 extra user threads in addition to those specified in the ARINC 653-1 configuration.

## Number of Timeouts

The timeout is a kernel resource representing a delayed action. The LynxOS-178 kernel allocates 1 timeout for each process and 1 timeout for each thread in every VM. For VM0, the number of timeouts is configured in the `uparam.h` file.

The ARINC 653-1 process uses timeouts for sending and receiving inter-partition queuing messages.

Therefore, at most one timeout can be used by the ARINC 653 process.

The timeout pool can be exhausted. In case more timeouts are required for some VM, that VM's number of threads should be increased accordingly.

### Thread Priorities

The implementation of the ARINC 653-1 support uses internal priorities in the range of 0 to 247, which are used as follows:

2-248	Priorities reserved for use by the ARINC 653-1 processes
0-1	Temporary lowered priorities, used internally
249-255	Temporarily boosted priorities, used internally

Worth noting is that the preemption lockout is implemented by raising the priority of the current process to 250 (thread has locked preemption). A deadlock can occur if the process invokes services of a driver that passes a request to a kernel thread with lower priority (and priority inheritance is not used).

At this moment, the `ttydriver` is known to have such a problem.

## Installing ARINC 653 to LynxOS-178

By default, the ARINC 653 support is installed in a dummy configuration. The ARINC 653 support device driver can be installed either dynamically or statically.

### Static Installation of ARINC 653 Support

The `$ENV_PREFIX/sys/devices/arinc653info.c` file describes the ARINC 653 configuration being linked into the kernel.

1. Edit the `$ENV_PREFIX/sys/devices/arinc653info.c` file as described in “Configuring the ARINC 653 Support Device Driver” on page 14.
2. In the `$ENV_PREFIX/sys/devices` directory execute the `make all` command.
3. Rebuild the kernel.



## Dynamic Installation of ARINC 653 Support

The `$ENV_PREFIX/sys/devices/arinc653info.c` file describes the ARINC 653-1 configuration being linked into the kernel.

1. Edit the `$ENV_PREFIX/sys/devices/arinc653info.c` file as described in “Configuring the VCT” on page 13.
2. In the `$ENV_PREFIX/sys/drivers.rsc/arinc653` directory execute the `make all` command.

Use the `arinc653.dldd` file as a character device driver and the `arinc653.info` file as the device driver information file.

## Configuring LynxOS-178 for ARINC 653 Support

The support for ARINC 653 APEX interfaces is implemented on top of the existing LynxOS-178 services. The configuration for a certain target is split in several places:

- Configuring the kernel
- Configuring the VCT
- Configuring the ARINC 653 support device driver
- Configuring the ARINC 653 File System Refer to subsections below for more information.

## Configuring the Kernel

Kernel configuration is adjusted by editing the `uparam.h` file and rebuilding the kernel. The following configurable parameters may need to be altered to support a specific ARINC 653 configuration:

**Table A -1: System-Wide Settings**

Setting	Description
NPROC	Total number of processes in the system. In addition to the explicitly configured ARINC 653-1 processes, the implementation needs 6 extra threads per partition. Note that the error handler thread should also be accounted for.

**Table A -1: System-Wide Settings (Continued)**

NTHREADS	Total number of threads in the system. Each process needs 1 thread, and there need to be 6 extra threads per partition.
USR_NFDS	The maximum number of files that a single process can open.

**Table A -2: Settings for Partition 0**

Setting	Description
NUMTOUTS	The number of kernel timeouts for partition 0. Refer to “Number of Timeouts” on page 9.
VMZERO_NPROC VMZERO_NTHREADS	<p>The number of processes and threads reserved for partition 0. If the partition 0 is used for ARINC 653, these numbers should be set as follows:</p> <ul style="list-style-type: none"> <li>• VMZERO_NPROC: (number of processes in partition 0) + 4</li> <li>• VMZERO_NTHREADS: VMZERO_NPROC + 1 + (number of kernel threads created by the installed drivers) + (number of ARINC 653-1 partitions * 2)</li> </ul> <p>Command line arguments for each process are specified in the VCT.</p>
NFILES	The number of files allocated in file table reserved for partition 0. If the partition 0 is used for ARINC 653, it should be big enough to cover all open files and shared memory regions.
NINODES	The number of inode entries allocated for partition 0. If the partition 0 is used for ARINC 653, it should be big enough to cover all open files.
NMOUNTS	The number of volumes that can be mounted in the partition 0.

## Configuring the VCT

For partitions other than partition 0, resource limits are configured using the VCT (settings for partition 0 are ignored). The following configurable parameters may need to be altered:

**Table A -3: System-Wide Settings**

Setting	Description
NumOfProcessesLim	The number of processes in a partition. Should be at least two more than the number of ARINC 653-1 processes.
NumOfThreadsLim	The number of threads in a partition. Should be one more than the NumOfProcessesLim value.
CommandLine	The command line to be executed. In ARINC 653 partitions, it should be specified as the path to the <code>arinc653_init</code> application.
NumOfOpenFdsPerVmLim	The number of the files allocated. The same as <code>NFILES</code> parameters described in Table 1-2.
FsNumOfInodesLim	The number of the inodes allocated. The same as <code>NINODES</code> parameters described in Table 1-2.

## Configuring the ARINC 653 Support Device Driver

The ARINC 653 support device driver is configured using the device information file (`sys/devices/arinc653info.c`). Type definitions for structures mentioned below are described in the `sys/dheaders/arinc653info.h` file.

**Table A -4: Configuring Device Driver**

Structure / Array	Contents
<code>arinc653_infostructure</code>	<ul style="list-style-type: none"> <li>• The number and configuration of sampling ports for each partition</li> <li>• The number and configuration of queuing ports for each partition</li> </ul> <p>The number and configuration of inter-partition shared memory blocks</p>

**Table A -4: Configuring Device Driver**

<code>arinc653_info_channels</code> array	<p>For each channel the following information is specified:</p> <ul style="list-style-type: none"> <li>• Channel name</li> <li>• Maximum message size</li> </ul> <p>Channel mode (<code>ARINC653_INFO_CHANNEL_QUEUEING</code>, <code>ARINC653_INFO_CHANNEL_SAMPLING</code>) Flow control mode (always set to <code>ARINC653_INFO_CHANNEL_FLOW_CONTROL_BLOCK</code>) Broadcast (always set to <code>ARINC653_INFO_CHANNEL_UNICAST</code>)</p>
<code>arinc653_info_ports</code> array	<p>For each port the following information is specified:</p> <ul style="list-style-type: none"> <li>• Number of partition port belongs to</li> <li>• Port name</li> <li>• Port direction (<code>SOURCE</code>, <code>DESTINATION</code>)</li> <li>• Port maximum number of messages</li> <li>• Port channel</li> </ul>
<code>arinc653_info_mblock</code> array	<p>For each memory block the following information is specified:</p> <ul style="list-style-type: none"> <li>• Address of the memory block to be used</li> <li>• Size of memory block in bytes</li> <li>• Memory block VM read-only access mode mask</li> <li>• Memory block VM read-write access mode mask</li> <li>• Memory block name</li> </ul>

Note that ports with the same channel index refer to the same channel. For sampling ports, that means every source port will be able to update the message in that channel and every destination port will be able to read this message. For queuing ports, every source port will be able to queue a message into the channel and every destination port can be used to fetch messages from the channel.

### Sample Driver Configuration.

Below is an example of ARINC 653 device info file. This file describes the following resources:

- memory block MBLOCK0 which has size of 4096 bytes and is available for read-write for partition (VM) 0 and for read-only for partition (VM) 1. Partition code may obtain virtual address on which block is mapped into its address space by using GET\_MEMORY\_BLOCK\_STATUS() API call.
- module local inter-partition communication channel "CHANNEL-0" which operates in queuing mode and has max message size 64. Channels themselves are not visible to partitions
- source queuing port QS1 available for partition (VM) 0 which has FIFO size of 1 message.
- destination queuing port QD1 available for partition (VM) 1 which has FIFO size of 1 message.

This example allows partition 0 to send messages to partition 1 via port QS1 and partition 1 may receive these messages via port QD1. Also, partition 0 may write data to virtual address of MBLOCK0 in its address space obtained by GET\_MEMORY\_BLOCK\_STATUS() call and partition 1 may see this data at virtual address in its address space obtained by the same GET\_MEMORY\_BLOCK\_STATUS() API call.

```
#include <arinc653info.h>

#if !defined(DLDD) || defined(ARINC653_INFO_MBLOCKS)
static const struct arinc653_info_mblock arinc653_info_mblocks[] = {
{
    .name = "MBLOCK0",
    .addr = ARINC653_MBLOCK_ADDR_NONE,
    .size = 4096,
    .rwmask = ARINC653_ACCESS_VM_ALLOWED(0),
    .romask = ARINC653_ACCESS_VM_ALLOWED(1)
}
}
#if defined(DLDD)
, {}
#endif
};
#endif

#if !defined(DLDD) || defined(ARINC653_INFO_CHANNELS)
static const struct arinc653_info_channel arinc653_info_channels[] = {
{
    .name = "CHANNEL-0",
    .max_message_size = 64,
    .max_nb_messages = 1,
    .mode = ARINC653_INFO_CHANNEL_QUEUING
}
}
#if defined(DLDD)
, {}
#endif
};
#endif

#if !defined(DLDD) || defined(ARINC653_INFO_PORTS)
static const struct arinc653_info_port arinc653_info_ports[] = {
```

```
{
    .vm = 0,
    .name = "QS1",
    .direction = SOURCE,
    .max_nb_messages = 1,
    .channel = "CHANNEL-0"
}, {
    .vm = 1,
    .name = "QD1",
    .direction = DESTINATION,
    .max_nb_messages = 1,
    .channel = "CHANNEL-0"
}
}
#endif defined(DLDD)
, {}
#endif
};
#endif

#if !defined(DLDD) || defined(ARINC653_INFO_GENERAL)
const struct arinc653_info arinc653_info = {
#if !defined(DLDD)
    .mblocks = arinc653_info_mblocks,
    .nmblocks =
sizeof(arinc653_info_mblocks)/sizeof(arinc653_info_mblocks[0]),
    .channels = arinc653_info_channels,
    .nchannels =
sizeof(arinc653_info_channels)/sizeof(arinc653_info_channels[0]),
    .ports = arinc653_info_ports,
    .nports = sizeof(arinc653_info_ports)/sizeof(arinc653_info_ports[0])
#endif
};
#endif
```

## Service Requirements

### Service Request Categories (P1:3.1, P2:3.1)

This section describes the service requests corresponding to the functions described in Section 2 of the ARINC 653-1 standard and a subset of functions described in Section 2 of ARINC 653-2 standard. The requests are grouped into the following major categories:

- Partition Management
- Process Management
- TimeManagement
- Memory Management
- Inter-partition Communication
- Intra-partition Communication

- Health Monitoring
- File System
- Memory Block

Each service request has a brief description of its functionality and states the conformance of the service to the ARINC 653 standard.

Deviations from the standard are discussed where appropriate.

## Return Code Data Type (P1:3.1.1, P2:3.1.1)

LynxOS-178 provides return codes as defined in Section 3.1.1 of the ARINC 653-1 standard and Section 3.1.1 of the ARINC 653-2 standard.

## Partition Management Services

This section contains services related to partition management.

### CREATE\_PARTITION

#### Description

The CREATE\_PARTITION call which existed in earlier LynxOS-178 versions is obsolete and shall be not used. An `ioctl` command is kept for compatibility with previous ARINC 653 library implementation.

#### Diagnostics

Return Code Value	Error Condition
OK	Successful completion

### GET\_PARTITION\_STATUS (P1:3.2.2.1)

#### Synopsis

```
void GET_PARTITION_STATUS(  
/*out*/ PARTITION_STATUS_TYPE *PARTITION_STATUS,  
/*out*/ RETURN_CODE_TYPE      *RETURN_CODE)
```

#### Description

The GET\_PARTITION\_STATUS service request is used to obtain the status of the current partition. On success, it returns partition status structure

`PARTITION_STATUS` containing information about operating mode, duration, period, lock level, identifier, and start condition of the current partition.

#### Diagnostics

Return Code Value	Error Condition
<code>NO_ERROR</code>	Successful completion
<code>INVALID_PARAM</code>	A NULL pointer passed for <code>PARTITION_STATUS</code> (Development mode only).
<code>INVALID_MODE</code>	No ARINC 653 partition associated with current thread (Development mode only).

#### Conformance

This service is implemented according to the ARINC 653-1 standard.

### SET\_PARTITION\_MODE (P1:3.2.2.2)

#### Synopsis

```
void SET_PARTITION_MODE(  
    /*in */ OPERATING_MODE_TYPE OPERATING_MODE,  
    /*out*/ RETURN_CODE_TYPE     *RETURN_CODE)
```

#### Description

The `SET_PARTITION_MODE` service request is used to set the operating mode of the current partition to one of the following modes:

`NORMAL`/`IDLE`/`COLD_START`/`WARM_START`. Use `SET_PARTITION_MODE` to set operating mode to `NORMAL` after the application portion of the initialization of the partition is complete.

The service is also expected to be used for setting the partition back to `IDLE` (partition shutdown), and to `COLD_START` or `WARM_START` (partition restart), when a serious fault is detected and processed.

#### Diagnostics

Return Code Value	Error Condition
<code>NO_ERROR</code>	Successful completion.
<code>INVALID_PARAM</code>	<code>OPERATING_MODE</code> does not represent an existing mode.



NO_ACTION	OPERATING_MODE is normal and current mode is NORMAL.
INVALID_MODE	OPERATING_MODE is WARM_START and current mode is COLD_START.
INVALID_MODE	No ARINC 653 partition associated with current thread (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## Process Management Services

This section contains services related to process management.

### GET\_PROCESS\_ID (P1:3.3.2.1)

#### Synopsis

```
void GET_PROCESS_ID(
/*in */ PROCESS_NAME_TYPE PROCESS_NAME,
/*out*/ PROCESS_ID_TYPE *PROCESS_ID,
/*out*/ RETURN_CODE_TYPE *RETURN_CODE)
```

#### Description

The GET\_PROCESS\_ID service request allows a process to obtain a process identifier by specifying the process name PROCESS\_NAME.

#### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_CONFIG	There is no current partition process named PROCESS_NAME.
INVALID_PARAM	A NULL pointer passed for PROCESS_NAME (Development mode only).
INVALID_PARAM	A NULL pointer passed for PROCESS_ID (Development mode only).
INVALID_MODE	No ARINC 653 partition associated with current thread (Development mode only).

## Conformance

This service is implemented according to the ARINC 653-1 standard.

### GET\_PROCESS\_STATUS (P1:3.3.2.2)

#### Synopsis

```
void GET_PROCESS_STATUS(  
    /*in */ PROCESS_ID_TYPE      PROCESS_ID,  
    /*out*/ PROCESS_STATUS_TYPE  *PROCESS_STATUS,  
    /*out*/ RETURN_CODE_TYPE     *RETURN_CODE)
```

#### Description

The `GET_PROCESS_STATUS` service request returns the current status of the specified process. The status structure `PROCESS_STATUS` contains process attributes, current priority value, deadline time, and process state. The current operating status of each of the individual processes of a partition is available to all processes within that partition.

#### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_PARAM	PROCESS_ID does not identify an existing process.
INVALID_MODE	No ARINC 653 partition associated with current thread (Development mode only).
INVALID_PARAM	A NULL pointer to process status passed (Development mode only).

## Conformance

This service is implemented according to the ARINC 653-1 standard.

### CREATE\_PROCESS (P1:3.3.2.3)

#### Synopsis

```
void CREATE_PROCESS(  
    /*in */ PROCESS_ATTRIBUTE_TYPE  *ATTRIBUTES,  
    /*out*/ PROCESS_ID_TYPE         *PROCESS_ID,  
    /*out*/ RETURN_CODE_TYPE        *RETURN_CODE)
```

#### Description

The `CREATE_PROCESS` service request creates a process and returns an identifier that denotes the created process.

The number of processes which can be created within an ARINC partition is limited by the maximum number of threads which can be created within VM (please note that each partition may require up to 6 extra threads for internal use). Also, not more than `MAX_NUMBER_OF_PROCESSES` can be created within ARINC partition. Consistency among process parameters and partition parameters is checked. The `CREATE_PROCESS` service request takes as input the process attributes structure `ATTRIBUTES` that contains desired name, base priority, period, time capacity, and deadline policy. The service creates a process and returns the identifier `PROCESS_ID` that denotes the created process.

### Diagnostics

Return Code Value	Error Condition
<code>NO_ERROR</code>	Successful completion
<code>INVALID_CONFIG</code>	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"> <li><code>ATTRIBUTES.NAME</code> does not identify a process name known by the configuration.</li> <li><code>ATTRIBUTES.PERIOD</code> is not consistent with the other parameters.</li> </ul>
<code>INVALID_PARAM</code>	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"> <li><code>ATTRIBUTES.BASE_PRIORITY</code> is out of range.</li> <li><code>ATTRIBUTES.TIME_CAPACITY</code> is out of range.</li> </ul>
<code>INVALID_MODE</code>	The operating mode is <code>NORMAL</code> .
<code>NO_ACTION</code>	Process is already created.
<code>INVALID_MODE</code>	No ARINC 653 partition associated with current thread (Development mode only).
<code>INVALID_PARAM</code>	A <code>NULL</code> pointer is passed as <code>ATTRIBUTES</code> or <code>PROCESS_ID</code> (Development mode only).
<code>INVALID_PARAM</code>	A <code>NULL</code> pointer is passed as process <code>ENTRY_POINT</code> (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard with the following exceptions:

- The error when `RETURN_CODE` is set to `INVALID_PARAM` if `ATTRIBUTES.STACK_SIZE` is out of range will never occur.

Rationale for deviations:

The stack size is adjusted by ARINC library if requested size is less than system minimum stack size.

## SET\_PRIORITY (P1:3.3.2.4)

### Synopsis

```
void SET_PRIORITY(  
/*in */ PROCESS_ID_TYPE PROCESS_ID,  
/*in */ PRIORITY_TYPE NEW_PRIORITY,  
/*out*/ RETURN_CODE_TYPE *RETURN_CODE)
```

### Description

The `SET_PRIORITY` service request changes the current priority of the process `PROCESS_ID`. The process is placed as the newest process with the priority `NEW_PRIORITY` in the `READY` state. Process rescheduling is performed after this service request only when the process whose priority is changed is in the `READY` or `RUNNING` state.

### Diagnostics

Return Code Value	Error Condition
<code>NO_ERROR</code>	Successful completion.
<code>INVALID_PARAM</code>	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"><li>• <code>PROCESS_ID</code> does not identify an existing process.</li><li>• <code>PRIORITY</code> is out of range.</li></ul>
<code>INVALID_MODE</code>	The specified process is in the <code>DORMANT</code> state.
<code>INVALID_MODE</code>	No ARINC 653 partition associated with current thread. (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## SUSPEND\_SELF (P1:3.3.2.5)

### Synopsis

```
void SUSPEND_SELF(
    /*in */ SYSTEM_TIME_TYPE    TIME_OUT,
    /*out*/ RETURN_CODE_TYPE    *RETURN_CODE)
```

### Description

The SUSPEND\_SELF service request suspends the execution of the current process, if aperiodic, until the RESUME service request is issued or the specified TIME\_OUT value expires.

### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_MODE	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"> <li>• Preemption is disabled or process is error handler process.</li> <li>• Process is periodic.</li> </ul>
INVALID_PARAM	TIME_OUT is out of range.
TIMED_OUT	TIME_OUT has elapsed.
INVALID_MODE	No ARINC 653 partition associated with current thread. (Development mode only).
INVALID_CONFIG	Failed to suspend a process with timeout. Not enough system resources (timeouts) (Development mode only).
INVALID_PARAM	Failed to suspend process with timeout. Invalid timeout value (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## SUSPEND (P1:3.3.2.6)

### Synopsis

```
void SUSPEND(
    /*in */ PROCESS_ID_TYPE    PROCESS_ID,
    /*out*/ RETURN_CODE_TYPE    *RETURN_CODE)
```

### Description

The `SUSPEND` service request allows the current process to suspend the execution of any aperiodic process except itself until the suspended process is resumed by another process. If the process `PROCESS_ID` is pending in a queue at the time it is suspended, it is not removed from that queue. When it is resumed, it will continue pending unless it has been removed from the queue (either by occurrence of a condition or expiration of a `TIME_OUT` or a reset of the queue) before the end of its suspension.

A process may suspend any other process asynchronously. Though this practice is not recommended, there may be partitions that require it.

### Diagnostics

Return Code Value	Error Condition
<code>NO_ERROR</code>	Successful completion.
<code>NO_ACTION</code>	Specified process has already been suspended.
<code>INVALID_PARAM</code>	<code>PROCESS_ID</code> does not identify an existing process or identifies itself.
<code>INVALID_MODE</code>	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"><li>• The state of the specified process is <code>DORMANT</code>.</li><li>• The specified process is periodic.</li><li>• Preemption is disabled and <code>PROCESS_ID</code> identifies the process which locked preemption</li></ul>
<code>INVALID_MODE</code>	No ARINC 653 partition associated with current thread (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

### RESUME (P1:3.3.2.7)

#### Synopsis

```
void RESUME(  
    /*in */ PROCESS_ID_TYPE    PROCESS_ID,  
    /*out*/ RETURN_CODE_TYPE    *RETURN_CODE)
```

#### Description

The `RESUME` service request allows the current process to resume another previously suspended process `PROCESS_ID`. The resumed process will become

READY if it is not waiting on a resource (delay, semaphore, period, event, message).  
A periodic process cannot be suspended, so it cannot be resumed.

#### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
NO_ACTION	Identified process is not a suspended process.
INVALID_PARAM	PROCESS_ID does not identify an existing process or identifies itself.
INVALID_MODE	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"> <li>• The state of the specified process is DORMANT.</li> <li>• PROCESS_ID identifies a periodic process.</li> </ul>
INVALID_MODE	No ARINC 653 partition/process associated with current thread (Development mode only).

#### Conformance

This service is implemented according to the ARINC 653-1 standard.

### STOP\_SELF (P1:3.3.2.8)

#### Synopsis

```
void STOP_SELF(void)
```

#### Description

The STOP\_SELF service request allows the current process to stop itself. If the current process is not the error handler process, the partition is placed in the unlocked condition.

No return code is returned to the requesting process procedure.

Note: this function may return without any action if calling thread does not represent an ARINC process.

#### Diagnostics

Return Code Value	Error Condition
None	

## Conformance

This service is implemented according to the ARINC 653-1 standard.

## STOP (P1:3.3.2.9)

### Synopsis

```
void STOP(  
    /*in */ PROCESS_ID_TYPE    PROCESS_ID,  
    /*out*/ RETURN_CODE_TYPE    *RETURN_CODE)
```

### Description

The `STOP` service request makes the process `PROCESS_ID` ineligible for processor resources until another process issues the `START` service request.

This procedure allows the current process to abort the execution of any process except itself. When a process aborts another process that is currently pending in a queue, the aborted process is removed from the queue.

Note: In development mode, this function may raise a HM error if failed to cancel thread which represents the specified ARINC process.

### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_PARAM	PROCESS_ID does not identify an existing process or identifies itself.
NO_ACTION	The state of the specified process is DORMANT.
INVALID_MODE	No ARINC 653 partition/process associated with current thread (Development mode only).

## Conformance

This service is implemented according to the ARINC 653-1 standard.

## START (P1:3.3.2.10)

### Synopsis

```
void START(  
    /*in */ PROCESS_ID_TYPE    PROCESS_ID,  
    /*out*/ RETURN_CODE_TYPE    *RETURN_CODE)
```



## Description

The `START` service request initializes all attributes of the process `PROCESS_ID` to their default values and resets the runtime stack of the process. If the partition is in the `NORMAL` mode, the process's deadline expiration time and next release point are calculated.

## Diagnostics

Return Code Value	Error Condition
<code>NO_ERROR</code>	Successful completion.
<code>INVALID_PARAM</code>	<code>PROCESS_ID</code> does not identify an existing process.
<code>NO_ACTION</code>	The state of the specified process is not <code>DORMANT</code> .

## Conformance

This service is implemented according to the ARINC 653-1 standard with the following exceptions:

- The error when `RETURN_CODE` is set to `INVALID_CONFIG` if `DEADLINE_TIME` calculation is out of range will never occur.

## Rationale for deviations:

`DEADLINE_TIME` calculation cannot overflow.

## DELAYED\_START (P1:3.3.2.11)

### Synopsis

```
void DELAYED_START(
/*in */ PROCESS_ID_TYPE    PROCESS_ID,
/*in */ SYSTEM_TIME_TYPE   DELAY_TIME,
/*out*/ RETURN_CODE_TYPE   *RETURN_CODE)
```

## Description

The `DELAYED_START` service request initializes all the attributes of the process `PROCESS_ID` to their default values, resets the runtime stack of the process, and places the process into the `WAITING` state (that is, the specified process goes from the `DORMANT` state to the `WAITING` state). If the partition is in the `NORMAL` operating mode, the process release point is calculated with the specified `DELAY_TIME`, and the process deadline expiration time is also calculated.

This procedure allows the current process to start the execution of another process during runtime.

## Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_PARAM	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"> <li>• PROCESS_ID does not identify an existing process.</li> <li>• DELAY_TIME is greater than or equal to the period of the specified process.</li> <li>• Infinite DELAY_TIME value is specified.</li> </ul>
NO_ACTION	The state of the specified process is not DORMANT.
INVALID_MODE	No ARINC 653 partition associated with current thread (Development mode only).

## Conformance

This service is implemented according to the ARINC 653-1 standard with the following exceptions:

- The error when RETURN\_CODE is set to INVALID\_CONFIG if DEADLINE\_TIME calculation is out of range will never occur.

Rationale for deviations:

DEADLINE\_TIME calculation cannot overflow.

- The error when RETURN\_CODE is set to INVALID\_PARAM if DELAY\_TIME calculation is out of range will never occur.

Rationale for deviations:

DELAY\_TIME calculation cannot overflow.

## LOCK\_PREEMPTION (P1:3.3.2.12)

## Synopsis

```
void LOCK_PREEMPTION(
/*out*/ LOCK_LEVEL_TYPE    *LOCK_LEVEL,
/*out*/ RETURN_CODE_TYPE   *RETURN_CODE)
```

## Description

The `LOCK_PREEMPTION` service request increments the lock level of the partition and disables process rescheduling for the partition. On success, it returns the new lock level value `LOCK_LEVEL`.

#### Diagnostics

Return Code Value	Error Condition
<code>NO_ERROR</code>	Successful completion.
<code>INVALID_CONFIG</code>	The <code>LOCK_LEVEL</code> value is higher than or equal to <code>MAX_LOCK_LEVEL</code> .
<code>NO_ACTION</code>	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"> <li>• <code>PROCESS_ID</code> does not identify an existing process: Calling process is error handler Preemption is disabled</li> </ul>
<code>INVALID_PARAM</code>	A <code>NULL</code> pointer passed as <code>LOCK_LEVEL</code> parameter (Development mode only).
<code>INVALID_MODE</code>	No ARINC 653 partition/process associated with current thread (Development mode only).

#### Conformance

This service is implemented according to the ARINC 653-1 standard.

### UNLOCK\_PREEMPTION (P1:3.3.2.13)

#### Synopsis

```
void UNLOCK_PREEMPTION(
/*out*/ LOCK_LEVEL_TYPE    *LOCK_LEVEL,
/*out*/ RETURN_CODE_TYPE   *RETURN_CODE)
```

#### Description

The `UNLOCK_PREEMPTION` service request decrements the current lock level of the partition. The process rescheduling function is performed only when the lock level becomes zero. On success, it returns the new lock level value `LOCK_LEVEL`.

#### Diagnostics

Return Code Value	Error Condition
<code>NO_ERROR</code>	Successful completion.

NO_ACTION	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"><li>• <code>PROCESS_ID</code> does not identify an existing process: Calling process is error handler. Preemption is disabled. <code>LOCK_LEVEL</code> indicates unlocked.</li></ul>
INVALID_PARAM	A NULL pointer passed as <code>LOCK_LEVEL</code> parameter (Development mode only).
INVALID_MODE	No ARINC 653 partition/process associated with current thread (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## GET\_MY\_ID (P1:3.3.2.14)

### Synopsis

```
void GET_MY_ID(  
/*out*/ PROCESS_ID_TYPE    *PROCESS_ID,  
/*out*/ RETURN_CODE_TYPE   *RETURN_CODE)
```

### Description

The `GET_MY_ID` service request returns the process identifier of the current process.

### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_MODE	Current process has no ID.
INVALID_PARAM	A NULL pointer passed as <code>PROCESS_ID</code> parameter (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## Time Management Services

This section contains services related to time management.

## TIMED\_WAIT (P1:3.4.2.1)

### Synopsis

```
void TIMED_WAIT(
/*in */ SYSTEM_TIME_TYPE    DELAY_TIME,
/*out*/ RETURN_CODE_TYPE    *RETURN_CODE)
```

### Description

The `TIMED_WAIT` service request suspends execution of the requesting process for a minimum amount of elapsed time `DELAY_TIME`. A `DELAY_TIME` of zero allows round-robin scheduling of processes of the same priority.

### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_MODE	Preemption is disabled or process is error handler process.
INVALID_PARAM	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"> <li>• <code>DELAY_TIME</code> is out of range.</li> <li>• <code>DELAY_TIME</code> is infinite.</li> </ul>
INVALID_MODE	No ARINC 653 partition/process associated with current thread (Development mode only).
INVALID_CONFIG	Not enough system resources (timeouts) (Development mode only).
INVALID_PARAM	Invalid timeout value (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## PERIODIC\_WAIT (P1:3.4.2.2)

### Synopsis

```
void PERIODIC_WAIT(
/*out*/ RETURN_CODE_TYPE *RETURN_CODE)
```

### Description

The `PERIODIC_WAIT` service request suspends execution of the requesting process until the next release point in the processor time line that corresponds to the period of the process.

## Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_MODE	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"><li>• Preemption is disabled or process is error handler process.</li><li>• The calling process is not periodic.</li></ul>
INVALID_MODE	No ARINC 653 partition/process associated with current thread (Development mode only).

## Conformance

This service is implemented according to the ARINC 653-1 standard.

## GET\_TIME (P1:3.4.2.3)

### Synopsis

```
void GET_TIME(  
    /*out*/ SYSTEM_TIME_TYPE    *SYSTEM_TIME,  
    /*out*/ RETURN_CODE_TYPE    *RETURN_CODE)
```

### Description

The service `GET_TIME` requests the value of the system clock. The system clock is the value of a clock common to all processors in the module. On success, it returns system time `SYSTEM_TIME` in nanoseconds.

## Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_PARAM	A NULL pointer passed for <code>SYSTEM_TIME</code> (Development mode only).

## Conformance

This service is implemented according to the ARINC 653-1 standard.

## REPLENISH (P1:3.4.2.4)

### Synopsis

```
void REPLENISH(
/*in */ SYSTEM_TIME_TYPE    BUDGET_TIME,
/*out*/ RETURN_CODE_TYPE    *RETURN_CODE)
```

### Description

The REPLENISH service request updates the deadline of the requesting process with a specified BUDGET\_TIME value. Postponing a periodic process's deadline past its next release point is not allowed.

### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_PARAM	BUDGET_TIME is out of range.
INVALID_MODE	The new deadline time would exceed the next release point.
NO_ACTION	Calling process is Error Handler or operating mode is not NORMAL.
INVALID_MODE	No ARINC 653 partition/process associated with current thread (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## Inter-partition Communication Services

This section contains services responsible for communication between processes residing in different partitions.

### Sampling Port Services (P1:3.6.2.1, P2:3.6.2)

A sampling port is a communication object allowing a partition to access a channel of communication configured to operate in sampling mode. Each new occurrence of a message overwrites the previous one. Messages have a fixed length. A refresh period attribute applies to reception ports. A validity output parameter indicates whether the age of the read message is consistent with the required refresh period attribute of the port. The REFRESH\_PERIOD attribute indicates the maximum

acceptable age of a valid message from the time it was received in the port. A port must be created during the initialization phase before it can be used.

The Sampling Port services introduced by ARINC 653-1 are extended by ARINC 653-2 to provide greater flexibility to the application when reading sampling port messages. In particular, it allows to read the message only if it has been updated since the last time it was read or to return a message only if it has been updated since a given reference time.

## CREATE\_SAMPLING\_PORT (P1:3.6.2.1.1)

### Synopsis

```
void CREATE_SAMPLING_PORT(  
/*in */ SAMPLING_PORT_NAME_TYPE    SAMPLING_PORT_NAME,  
/*in */ MESSAGE_SIZE_TYPE          MAX_MESSAGE_SIZE,  
/*in */ PORT_DIRECTION_TYPE        PORT_DIRECTION,  
/*in */ SYSTEM_TIME_TYPE           REFRESH_PERIOD,  
/*out*/ SAMPLING_PORT_ID_TYPE      *SAMPLING_PORT_ID,  
/*out*/ RETURN_CODE_TYPE           *RETURN_CODE)
```

### Description

The `CREATE_SAMPLING_PORT` service request is used to create a sampling port with the name `SAMPLING_PORT_NAME`. The parameter `MAX_MESSAGE_SIZE` defines the maximum size of the message that can be communicated through the port. An identifier `SAMPLING_PORT_ID` is assigned to the port by the OS and returned to the calling process. `PORT_DIRECTION` is either `SOURCE` or `DESTINATION`. For a `SOURCE` port, the `REFRESH_PERIOD` parameter is ignored. At creation, the port is empty. The partition initialization process can create as many sampling ports as the preallocated memory space will support, but not more than `MAX_NUMBER_OF_SAMPLING_PORTS`.

Note: in development mode this API will return without any action if value of specified `RETURN_CODE` parameter is `NULL`.

### Diagnostics

Return Code Value	Error Condition
<code>NO_ERROR</code>	Successful completion.
<code>NO_ACTION</code>	The port named <code>SAMPLING_PORT_NAME</code> is already created.



INVALID_CONFIG	<p>The return code corresponds to one of the following conditions:</p> <ul style="list-style-type: none"> <li>• <code>SAMPLING_PORT_NAME</code> does not identify a sampling port known by the configuration.</li> <li>• <code>MAX_MESSAGE_SIZE</code> is out of range or not compatible with the configuration.</li> <li>• <code>PORT_DIRECTION</code> is invalid or not compatible with the configuration.</li> <li>• <code>REFRESH_PERIOD</code> is out of range.</li> </ul>
INVALID_MODE	Operating mode is NORMAL.
INVALID_PARAM	A NULL pointer is specified as <code>SAMPLING_PORT_ID</code> parameter (Development mode only).
INVALID_MODE	No ARINC 653 partition associated with current thread (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## WRITE\_SAMPLING\_MESSAGE (P1:3.6.2.1.2)

### Synopsis

```

void WRITE_SAMPLING_MESSAGE(
/*in */ SAMPLING_PORT_ID_TYPE    SAMPLING_PORT_ID,
/*in */ MESSAGE_ADDR_TYPE        MESSAGE_ADDR,
/*in */ MESSAGE_SIZE_TYPE        LENGTH,
/*out*/ RETURN_CODE_TYPE         *RETURN_CODE)

```

### Description

The `WRITE_SAMPLING_MESSAGE` service request is used to write a message of size `LENGTH` stored in `MESSAGE_ADDR` to the sampling port `SAMPLING_PORT_ID`. The message overwrites the previous one.

### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_PARAM	<code>SAMPLING_PORT_ID</code> does not identify an existing sampling port.
NVALID_CONFIG	<code>LENGTH</code> is not compatible with the configuration of the specified port.

INVALID_MODE	SAMPLING_PORT_ID is not configured to operate as a source.
INVALID_PARAM	A NULL pointer passed for MESSAGE_ADDR (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

**NOTE:** WRITE\_SAMPLING\_MESSAGE takes LENGTH as an input parameter. In case sampling port is used to pass text messages, the developer has to keep in mind that LENGTH has to be large enough to account for the terminating zero symbol \0. This is due to the way C language handles strings. For example, the following call to WRITE\_SAMPLING\_MESSAGE is valid:

```
WRITE_SAMPLING_MESSAGE(InPortId, (MESSAGE_ADDR_TYPE)"TRUE", 5, &ReturnCode);
```

This note does not apply to binary messages.

## READ\_SAMPLING\_MESSAGE (P1:3.6.2.1.3)

### Synopsis

```
void READ_SAMPLING_MESSAGE(
/*in */SAMPLING_PORT_ID_TYPE      SAMPLING_PORT_ID,
/*in */MESSAGE_ADDR_TYPE          MESSAGE_ADDR,
/*out*/MESSAGE_SIZE_TYPE          *LENGTH,
/*out*/VALIDITY_TYPE              *VALIDITY,
/*out*/RETURN_CODE_TYPE           *RETURN_CODE)
```

### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
NO_ACTION	Sampling port is empty.
INVALID_PARAM	SAMPLING_PORT_ID does not identify an existing sampling port.
INVALID_MODE	SAMPLING_PORT_ID is not configured to operate as a DESTINATION.
INVALID_PARAM	A NULL pointer passed for MESSAGE_ADDR (Development mode only).

INVALID_PARAM	A NULL passed for LENGTH (Development mode only).
INVALID_PARAM	A NULL passed for VALIDITY (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## GET\_SAMPLING\_PORT\_ID (P1:3.6.2.1.4)

### Synopsis

```
void GET_SAMPLING_PORT_ID(
/*in */ SAMPLING_PORT_NAME_TYPE    SAMPLING_PORT_NAME,
/*out*/ SAMPLING_PORT_ID_TYPE      *SAMPLING_PORT_ID,
/*out*/ RETURN_CODE_TYPE           *RETURN_CODE)
```

### Description

The GET\_SAMPLING\_PORT\_ID service allows a process to obtain a sampling port identifier SAMPLING\_PORT\_ID by specifying the port name SAMPLING\_PORT\_NAME.

Note: in development mode this API will return without any action if value of specified RETURN\_CODE parameter is NULL.

### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_CONFIG	There is no current sampling port named SAMPLING_PORT_NAME.
INVALID_PARAM	A NULL pointer passed for SAMPLING_PORT_ID (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## GET\_SAMPLING\_PORT\_STATUS (P1:3.6.2.1.5)

### Synopsis

```
void GET_SAMPLING_PORT_STATUS(
/*in */ SAMPLING_PORT_ID_TYPE      SAMPLING_PORT_ID,
/*out*/ SAMPLING_PORT_STATUS_TYPE  *SAMPLING_PORT_STATUS,
/*out*/ RETURN_CODE_TYPE           *RETURN_CODE)
```

## Description

The `GET_SAMPLING_PORT_STATUS` service returns the current status structure `SAMPLING_PORT_STATUS` of the sampling port `SAMPLING_PORT_ID`. The status structure provides information on maximum message size, port direction, refresh period, and last message validity.

## Diagnostics

Return Code Value	Error Condition
<code>NO_ERROR</code>	Successful completion.
<code>INVALID_PARAM</code>	<code>SAMPLING_PORT_ID</code> does not identify an existing sampling port.
<code>INVALID_PARAM</code>	A NULL pointer passed for <code>SAMPLING_PORT_STATUS</code> (Development mode only).

## Conformance

This service is implemented according to the ARINC 653-1 standard.

## READ\_UPDATED\_SAMPLING\_MESSAGE (P2:3.6.2.1)

### Synopsis

```
void READ_UPDATED_SAMPLING_MESSAGE (
/*in */ SAMPLING_PORT_ID_TYPE      SAMPLING_PORT_ID,
/*in */ MESSAGE_ADDR_TYPE          MESSAGE_ADDR,
/*inout*/ MESSAGE_SIZE_TYPE        *LENGTH,
/*out*/ UPDATED_TYPE              *UPDATED,
/*out*/ RETURN_CODE_TYPE           *RETURN_CODE)
```

### Description

The `READ_UPDATED_SAMPLING_MESSAGE` service request is used to read a message from the specified sampling port only if it has been updated since the last request. The message is not read if a new message was not received at the port. An updated output parameter indicates whether the message was updated.

This service does not utilize the `REFRESH_PERIOD` logic (as in `READ_SAMPLING_MESSAGE`). The O/S updates the message read indication for the port (set to “not consumed”) when new data is copied into the port.

## Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
NO_ACTION	Sampling port is empty
NO_ACTION	Sampling port contains old message
INVALID_PARAM	SAMPLING_PORT_ID does not identify an existing sampling port
INVALID_PARAM	A NULL pointer passed for MESSAGE_ADDR (Development mode only).
INVALID_PARAM	A NULL passed for LENGTH (Development mode only).
INVALID_PARAM	A NULL passed for UPDATED (Development mode only).
INVALID_MODE	SAMPLING_PORT_ID is not configured to operate as a Destination

## Conformance

This service is implemented according to the ARINC 653-2 standard.

## GET\_SAMPLING\_PORT\_CURRENT\_STATUS (P2:3.6.2.2)

### Synopsis

```
void GET_SAMPLING_PORT_CURRENT_STATUS (
/*in */ SAMPLING_PORT_ID_TYPE          SAMPLING_PORT_ID,
/*out*/ SAMPLING_PORT_CURRENT_STATUS_TYPE *SAMPLING_PORT_STATUS,
/*out*/ RETURN_CODE_TYPE               *RETURN_CODE)
```

### Description

This service returns a parameter of `SAMPLING_PORT_CURRENT_STATUS_TYPE` which is based on `SAMPLING_PORT_STATUS_TYPE` with the following differences:

- Addition of a time stamp that indicates when the message was completely written into destination sampling port by the O/S.
- Replaces `VALIDITY` flag by `MESSAGE_AGE` flag. The `MESSAGE_AGE` flag indicates whether or not the message has been in the destination sampling port (without being replaced) for longer than the port's period.
- Addition of `UPDATED` state (`EMPTY_PORT`, `CONSUMED_MESSAGE`, `NEW_MESSAGE`).

## Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_PARAM	SAMPLING_PORT_ID does not identify an existing sampling port
INVALID_PARAM	A NULL pointer passed for SAMPLING_PORT_STATUS (Development mode only).

## Conformance

This service is implemented according to the ARINC 653-2 standard.

## READ\_SAMPLING\_MESSAGE\_CONDITIONAL(P2:3.6.2.3)

## Synopsis

```
void READ_SAMPLING_MESSAGE_CONDITIONAL (
/*in */ SAMPLING_PORT_ID_TYPE      SAMPLING_PORT_ID,
/*in */ SYSTEM_TIME_TYPE           REF_TIME_STAMP,
/*in */ MESSAGE_ADDR_TYPE          MESSAGE_ADDR,
/*out*/ MESSAGE_SIZE_TYPE          *LENGTH,
/*out*/ SYSTEM_TIME_TYPE           *TIME_STAMP,
/*out*/ RETURN_CODE_TYPE           *RETURN_CODE)
```

## Description

The READ\_SAMPLING\_MESSAGE\_CONDITIONAL service will read the message if the timestamp of the message is greater than (not greater than or equal to) REF\_TIME\_STAMP. TIME\_STAMP is the timestamp of the message read.

## Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
NO_ACTION	Sampling port is empty
NO_ACTION	Sampling port contains “old” message this depends on the READ_CONDITION input parameter
INVALID_PARAM	SAMPLING_PORT_ID does not identify an existing sampling port
INVALID_PARAM	REF_TIME_STAMP is invalid
INVALID_MODE	SAMPLING_PORT_ID is not configured to operate as a destination

INVALID_PARAM	A NULL pointer passed for MESSAGE_ADDR (Development mode only).
INVALID_PARAM	A NULL passed for LENGTH (Development mode only).
INVALID_PARAM	A NULL passed for TIME_STAMP (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-2 standard.

### Queuing Port Services (P1:3.6.2.2)

A queuing port is a communication object allowing a partition to access a channel of communication configured to operate in queuing mode. Messages are stored in FIFO order. Messages have variable length. In queuing mode, each new instance of a message cannot overwrite the previous one stored in the send or receive FIFO. If the receiving FIFO is full, however, new messages may be discarded; an appropriate value of `return_code` is set in accordance.

A send/respond protocol can be implemented by the applications to guard against communication failures and to ensure flow control between source and destination. The destination partition determines when it will read its messages. A port must be created during the initialization mode before it can be used.

#### CREATE\_QUEUING\_PORT (P1:3.6.2.2.1)

##### Synopsis

```
void CREATE_QUEUING_PORT(
/*in */ QUEUING_PORT_NAME_TYPE    QUEUING_PORT_NAME,
/*in */ MESSAGE_SIZE_TYPE         MAX_MESSAGE_SIZE,
/*in */ MESSAGE_RANGE_TYPE        MAX_NB_MESSAGE,
/*in */ PORT_DIRECTION_TYPE        PORT_DIRECTION,
/*in */ QUEUING_DISCIPLINE_TYPE    QUEUING_DISCIPLINE,
/*out*/ QUEUING_PORT_ID_TYPE       *QUEUING_PORT_ID,
/*out*/ RETURN_CODE_TYPE           *RETURN_CODE)
```

##### Description

The `CREATE_QUEUING_PORT` service is used to create a port of communication operating in queuing mode. The port is named `QUEUING_PORT_NAME`. It can handle up to `MAX_NB_MESSAGE` number of messages, where each message can be up to `MAX_MESSAGE_SIZE` bytes in size. `PORT_DIRECTION` can be either `SOURCE` or `DESTINATION`. An identifier `QUEUING_PORT_ID` is assigned by the OS and returned to the calling process. At creation, the port is empty. The `QUEUING_DISCIPLINE` attribute indicates whether blocked processes are queued

in FIFO, or in priority order. The partition initialization process can create as many queuing ports as the preallocated memory space will support, but not more than `MAX_NUMBER_OF_QUEUING_PORTS`.

Note: in development mode this API will return without any action if value of specified `RETURN_CODE` parameter is `NULL`.

#### Diagnostics

Return Code Value	Error Condition
<code>NO_ERROR</code>	Successful completion.
<code>NO_ACTION</code>	Port named <code>QUEUING_PORT_NAME</code> is already created.
<code>INVALID_CONFIG</code>	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"> <li>• <code>QUEUING_PORT_NAME</code> does not identify a queuing port known by the configuration.</li> <li>• <code>MAX_MESSAGE_SIZE</code> is out of range or not compatible with the configuration.</li> <li>• <code>MAX_NB_MESSAGE</code> is out of range or not compatible with the configuration.</li> <li>• <code>PORT_DIRECTION</code> is invalid or not compatible with the configuration.</li> </ul> <code>QUEUING_DISCIPLINE</code> is invalid or not compatible with the configuration.
<code>INVALID_MODE</code>	Operating mode is <code>NORMAL</code> .
<code>INVALID_PARAM</code>	A <code>NULL</code> pointer is passed as <code>QUEUING_PORT_ID</code> parameter (Development mode only).
<code>INVALID_MODE</code>	No ARINC 653 partition associated with current thread (Development mode only).

#### Conformance

This service is implemented according to the ARINC 653-1 standard with the following exceptions:

- Due to design restrictions, the only available queuing discipline is priority order. The request for the FIFO queuing discipline is ignored, and the priority-based one is used instead.

Rationale for deviations:



It is impossible to implement FIFO queuing of processes using existing LynxOS-178 primitives. Modifying the LynxOS-178 standard method of blocking a process would break the existing LynxOS-178 certification.

## SEND\_QUEUING\_MESSAGE (P1:3.6.2.2.2)

### Synopsis

```
void SEND_QUEUING_MESSAGE(
/*in */ QUEUING_PORT_ID_TYPE    QUEUING_PORT_ID,
/*in */ MESSAGE_ADDR_TYPE       MESSAGE_ADDR,
/*in */ MESSAGE_SIZE_TYPE       LENGTH,
/*in */ SYSTEM_TIME_TYPE        TIME_OUT,
/*out*/ RETURN_CODE_TYPE        *RETURN_CODE)
```

### Description

The `SEND_QUEUING_MESSAGE` service request is used to send a message to the queuing port `QUEUING_PORT_ID`. The message is of size `LENGTH` and is stored in `MESSAGE_ADDR`. If there is sufficient space in the queuing port to accept the message, the message is added to the end of the port's message queue. If there is insufficient space, the process is blocked and added to the sending process queue, according to the queuing discipline of the port. The process stays on the queue until the specified `TIME_OUT`, if finite, expires or there is enough space in the port to accept the message.

Note: in development mode this API will return without any action if value of specified `RETURN_CODE` parameter is `NULL`.

### Diagnostics

Return Code Value	Error Condition
<code>NO_ERROR</code>	Successful completion.
<code>INVALID_PARAM</code>	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"> <li>• <code>QUEUING_PORT_ID</code> does not identify an existing queuing port.</li> <li>• <code>TIME_OUT</code> is out of range.</li> <li>• <code>LENGTH</code> is zero or negative.</li> </ul>
<code>INVALID_CONFIG</code>	<code>LENGTH</code> is greater than <code>MAX_MESSAGE_SIZE</code> for the specified port.

INVALID_MODE	The return code corresponds to one of the following conditions: QUEUEING_PORT_ID is not configured to operate as a SOURCE. (Preemption is disabled or process is error handler process) and TIME_OUT is not zero.
NOT_AVAILABLE	Insufficient space in the QUEUEING_PORT_ID to accept a new message.
TIMED_OUT	Specified TIME_OUT has expired.
INVALID_MODE	No ARINC 653 partition/process associated with current thread (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

---

**NOTE:** SEND\_QUEUEING\_MESSAGE takes LENGTH as an input parameter. In case the queuing port is used for text messages, the developer has to keep in mind that LENGTH has to be large enough to account for the terminating zero symbol \0. This is due to the way C language handles strings. For example, the following call to SEND\_QUEUEING\_MESSAGE is valid:

```
SEND_QUEUEING_MESSAGE(InPortId, (MESSAGE_ADDR_TYPE)"OK", 3, InTimeOut,
&ReturnCode);
```

This note does not apply to binary messages.

---

## RECEIVE\_QUEUEING\_MESSAGE (P1:3.6.2.2.3)

### Synopsis

```
void RECEIVE_QUEUEING_MESSAGE (
/*in */QUEUEING_PORT_ID_TYPE    QUEUEING_PORT_ID,
/*in */SYSTEM_TIME_TYPE         TIME_OUT,
/*in */MESSAGE_ADDR_TYPE        MESSAGE_ADDR,
/*out*/MESSAGE_SIZE_TYPE        *LENGTH,
/*out*/RETURN_CODE_TYPE         *RETURN_CODE)
```

### Description

The RECEIVE\_QUEUEING\_MESSAGE service request is used to receive a message from the queuing port QUEUEING\_PORT\_ID. If the queuing port is not empty, the message at the head of the port's message queue is removed and returned to the calling process. If the queuing port is empty, the process is blocked and added to

the receiving process queue, according to the queuing discipline of the port. The process stays on the queue until the specified `TIME_OUT`, if finite, expires or a message arrives in the port. The received message of size `LENGTH` is stored in `MESSAGE_ADDR`.

Note: in development mode this API will return without any action if value of specified `RETURN_CODE` parameter is `NULL`.

**NOTE:** This service request does not specify the amount of memory available at the `MESSAGE_ADDR` address. It is assumed that the biggest message as it is configured can be safely accommodated. The maximum message size can be queried with the `GET_QUEUING_PORT_STATUS` request.

### Diagnostics

Return Code Value	Error Condition
<code>NO_ERROR</code>	Successful completion.
<code>TIMED_OUT</code>	Specified <code>TIME_OUT</code> has expired.
<code>INVALID_PARAM</code>	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"> <li>• <code>QUEUING_PORT_ID</code> does not identify an existing queuing port.</li> <li>• <code>TIME_OUT</code> is out of range.</li> </ul>
<code>INVALID_MODE</code>	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"> <li>• <code>QUEUING_PORT_ID</code> is not configured to operate as a <code>DESTINATION</code>.</li> <li>• (Preemption is disabled or process is error handler process) and <code>TIME_OUT</code> is not zero.</li> </ul> Message queue of the specified port is empty and <code>TIME_OUT</code> is equal to 0
<code>NOT_AVAILABLE</code>	There is no message in the <code>QUEUING_PORT_ID</code> and <code>TIME_OUT</code> is equal to 0.
<code>INVALID_PARAM</code>	A <code>NULL</code> pointer passed for <code>LENGTH</code> (Development mode only).
<code>INVALID_MODE</code>	No ARINC 653 partition/process associated with current thread (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard with the following exceptions:

- The error when `RETURN_CODE` is set to `INVALID_CONFIG` due to message queue overflow will never occur.

Rationale for deviations:

Due to design restrictions message queue overflow detection is not implemented.

## GET\_QUEUING\_PORT\_ID (P1:3.6.2.2.4)

### Synopsis

```
void GET_QUEUING_PORT_ID(  
    /*in */ QUEUING_PORT_NAME_TYPE    QUEUING_PORT_NAME,  
    /*out*/ QUEUING_PORT_ID_TYPE      *QUEUING_PORT_ID,  
    /*out*/ RETURN_CODE_TYPE          *RETURN_CODE)
```

### Description

The `GET_QUEUING_PORT_ID` service allows a process to obtain a queuing port identifier `QUEUING_PORT_ID` by specifying the queuing port name `QUEUING_PORT_NAME`.

Note: in development mode this API will return without any action if value of specified `RETURN_CODE` parameter is `NULL`.

### Diagnostics

Return Code Value	Error Condition
<code>NO_ERROR</code>	Successful completion.
<code>INVALID_CONFIG</code>	There is no current queuing port named <code>QUEUING_PORT_NAME</code> .
<code>INVALID_PARAM</code>	A <code>NULL</code> pointer passed for <code>QUEUING_PORT_ID</code> (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## GET\_QUEUING\_PORT\_STATUS (P1:3.6.2.2.5)

### Synopsis

```

void GET_QUEUING_PORT_STATUS (
/*in */ QUEUING_PORT_ID_TYPE      QUEUING_PORT_ID,
/*out*/ QUEUING_PORT_STATUS_TYPE  *QUEUING_PORT_STATUS,
/*out*/ RETURN_CODE_TYPE          *RETURN_CODE)

```

### Description

The `GET_QUEUING_PORT_STATUS` service returns the current status `QUEUING_PORT_STATUS` of the queuing port `QUEUING_PORT_ID`.

### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_PARAM	QUEUING_PORT_ID does not identify an existing queuing port.
INVALID_PARAM	A NULL pointer passed for QUEUING_PORT_STATUS (Development mode only).

## CLEAR\_QUEUING\_PORT (P1: 3.6.2.2.6)

### Synopsis

```

void CLEAR_QUEUING_PORT (
/*in */ QUEUING_PORT_ID_TYPE      QUEUING_PORT_ID,
/*out*/ RETURN_CODE_TYPE          *RETURN_CODE )

```

### Description

The `CLEAR_QUEUING_PORT` service request discards any messages in the specified port's receive queue. The service request has no effect on processes waiting on the queuing port (i.e., when there are no messages in the port's receive queuing).

### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_PARAM	QUEUING_PORT_ID does not identify an existing queuing port.
INVALID_MODE	The specified port is not configured as a destination port

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## Intra-partition Communication Services

This section contains services responsible for communication between processes residing in the same partition.

### Buffer Services (P1:3.7.2.1)

A buffer is a communication object used by processes of a same partition to send or receive messages. In buffers, the messages are queued in FIFO order. The buffer message size is variable, but a maximum size value is given at buffer creation. A buffer must be created during the initialization mode before it can be used. A name is given at buffer creation. This name is local to the partition and is not an attribute of the partition configuration table.

#### CREATE\_BUFFER (P1:3.7.2.1.1)

##### Synopsis

```
void CREATE_BUFFER(  
/*in */ BUFFER_NAME_TYPE          BUFFER_NAME,  
/*in */ MESSAGE_SIZE_TYPE         MAX_MESSAGE_SIZE,  
/*in */ MESSAGE_RANGE_TYPE        MAX_NB_MESSAGE,  
/*in */ QUEUING_DISCIPLINE_TYPE   QUEUING_DISCIPLINE,  
/*out*/ BUFFER_ID_TYPE            *BUFFER_ID,  
/*out*/ RETURN_CODE_TYPE          *RETURN_CODE)
```

##### Description

The `CREATE_BUFFER` service request is used to create a message buffer operating in queuing mode. The buffer is named `BUFFER_NAME`. It can handle up to `MAX_NB_MESSAGE` number of messages, where each message can be up to `MAX_MESSAGE_SIZE` bytes in size.

`BUFFER_ID` is assigned by the OS and returned to the calling process. Processes can create as many buffers as the preallocated memory space will support. The `QUEUING_DISCIPLINE` input parameter indicates the process queuing policy (FIFO or priority order) associated with that buffer.

##### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_CONFIG	Not enough available storage space for the creation of the specified buffer or maximum number of buffers have been created.

NO_ACTION	The buffer named BUFFER_NAME has already been created.
INVALID_PARAM	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"> <li>• MAX_MESSAGE_SIZE is zero or negative.</li> <li>• MAX_NB_MESSAGE is out of range.</li> <li>• QUEUING_DISCIPLINE is not valid.</li> </ul>
INVALID_MODE	Operating mode is NORMAL.
INVALID_MODE	No ARINC 653 partition associated with current thread (Development mode only).
INVALID_PARAM	A NULL pointer is specified as BUFFER_ID (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard with the following exception:

- Due to design restrictions, the only available queuing discipline is priority order. The request for the FIFO queuing discipline is ignored, and the priority-based one is used instead.

### Rationale for deviation:

It is impossible to implement FIFO queuing of processes using existing LynxOS-178 primitives. Modifying the LynxOS-178 standard method of blocking a process would break the existing LynxOS-178 certification.

## SEND\_BUFFER (P1:3.7.2.1.2)

### Synopsis

```
void SEND_BUFFER(
/*in */ BUFFER_ID_TYPE      BUFFER_ID,
/*in */ MESSAGE_ADDR_TYPE   MESSAGE_ADDR,
/*in */ MESSAGE_SIZE_TYPE   LENGTH,
/*in */ SYSTEM_TIME_TYPE    TIME_OUT,
/*out*/ RETURN_CODE_TYPE    *RETURN_CODE)
```

### Description

The SEND\_BUFFER service request is used to send a message of size LENGTH stored in MESSAGE\_ADDR to the buffer BUFFER\_ID. The calling process will be queued while the buffer is full for a maximum duration of specified TIME\_OUT.

## Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_PARAM	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"><li>• BUFFER_ID does not identify an existing buffer.</li><li>• LENGTH is greater than the MAX_MESSAGE_SIZE specified for the buffer</li><li>• LENGTH is zero or negative</li><li>• TIME_OUT is out of range.</li></ul>
INVALID_MODE	(Preemption is disabled or process is error handler process) and TIME_OUT is not zero.
NOT_AVAILABLE	No place in the buffer to put a message and TIME_OUT is zero.
TIMED_OUT	The specified TIME_OUT has expired.
INVALID_MODE	<ul style="list-style-type: none"><li>• No ARINC 653 partition associated with current thread (Development mode only).</li><li>• Partition mode is not NORMAL (Development mode only).</li><li>• No ARINC 653 process associated with current thread (Development mode only).</li></ul>
INVALID_PARAM	A NULL pointer is passed for MESSAGE_ADDR (Development mode only).

## Conformance

This service is implemented according to the ARINC 653-1 standard.

---

**NOTE:** SEND\_BUFFER takes LENGTH as an input parameter. If the buffer is used for text messages, the developer has to keep in mind that LENGTH has to be large enough to account for the terminating zero symbol \0. This is due to the way C language handles strings. For example, the following call to SEND\_BUFFER is valid:

```
SEND_BUFFER(BufferId, (MESSAGE_ADDR_TYPE)"OK", 3, InTimeOut,  
&ReturnCode);
```

This note does not apply to binary messages.

---



## RECEIVE\_BUFFER (P1:3.7.2.1.3)

### Synopsis

```
void RECEIVE_BUFFER(  
    /*in */ BUFFER_ID_TYPE      BUFFER_ID,  
    /*in */ SYSTEM_TIME_TYPE    TIME_OUT,  
    /*in */ MESSAGE_ADDR_TYPE   MESSAGE_ADDR,  
    /*out*/ MESSAGE_SIZE_TYPE   *LENGTH,  
    /*out*/ RETURN_CODE_TYPE    *RETURN_CODE)
```

### Description

The `RECEIVE_BUFFER` service request is used to receive a message from the buffer `BUFFER_ID`. The calling process will be queued while the buffer is empty for a `TIME_OUT` maximum duration. The received message is of size `LENGTH` and is stored in `MESSAGE_ADDR`.

---

**NOTE:** This service request does not specify the amount of memory available at the `MESSAGE_ADDR` address. It is assumed that the biggest message as it is configured can be safely accommodated. The maximum message size can be queried with the `GET_BUFFER_STATUS` request.

---

### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_PARAM	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"><li>• <code>BUFFER_ID</code> does not identify an existing buffer.</li></ul> <code>TIME_OUT</code> is out of range.
INVALID_MODE	(Preemption is disabled or process is error handler process) and <code>TIME_OUT</code> is not zero.
NOT_AVAILABLE	The buffer does not contain any message and <code>TIME_OUT</code> is 0.
TIMED_OUT	The specified <code>TIME_OUT</code> has expired.

INVALID_MODE	<ul style="list-style-type: none"> <li>• No ARINC 653 partition associated with current thread (Development mode only).</li> <li>• Partition mode is not NORMAL (Development mode only).</li> <li>• No ARINC 653 process associated with current thread (Development mode only).</li> </ul>
INVALID_PARAM	<ul style="list-style-type: none"> <li>• A NULL pointer is passed for MESSAGE_ADDR (Development mode only).</li> <li>• A NULL pointer is passed for LENGTH (Development mode only).</li> </ul>

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## GET\_BUFFER\_ID (P1:3.7.2.1.4)

### Synopsis

```
void GET_BUFFER_ID(
    /*in */ BUFFER_NAME_TYPE    BUFFER_NAME,
    /*out*/ BUFFER_ID_TYPE      *BUFFER_ID,
    /*out*/ RETURN_CODE_TYPE    *RETURN_CODE)
```

### Description

The GET\_BUFFER\_ID service request allows the current process to get the identifier BUFFER\_ID for the buffer named BUFFER\_NAME.

### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_CONFIG	There is no current partition buffer named BUFFER_NAME.
INVALID_PARAM	A NULL pointer passed for BUFFER_ID (Development mode only).
INVALID_MODE	No ARINC 653 partition associated with current thread (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## GET\_BUFFER\_STATUS (P1:3.7.2.1.5)

### Synopsis

```
void GET_BUFFER_STATUS(
/*in */ BUFFER_ID_TYPE      BUFFER_ID,
/*out*/ BUFFER_STATUS_TYPE  *BUFFER_STATUS,
/*out*/ RETURN_CODE_TYPE    *RETURN_CODE)
```

### Description

The `GET_BUFFER_STATUS` service request returns the status `BUFFER_STATUS` of the buffer `BUFFER_ID`.

### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_PARAM	BUFFER_ID does not identify an existing buffer.
INVALID_PARAM	A NULL pointer passed for BUFFER_STATUS (Development mode only).
INVALID_MODE	No ARINC 653 partition associated with current thread (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## Blackboard Services (P1:3.7.2.2)

A blackboard is a communication object used by processes of the same partition to send or receive messages. A blackboard does not use message queues; each new occurrence of a message overwrites the current one. The blackboard message size is variable, but a maximum size value is given at blackboard creation. A blackboard must be created during the initialization mode before it can be used. The memory size given in the configuration table is the size necessary to manage all the blackboards of a partition. A name is given at blackboard creation. This name is local to the partition and is not an attribute of the partition configuration table.

## CREATE\_BLACKBOARD (P1:3.7.2.2.1)

### Synopsis

```
void CREATE_BLACKBOARD(
/*in */ BLACKBOARD_NAME_TYPE  BLACKBOARD_NAME,
```

```
/*in */ MESSAGE_SIZE_TYPE          MAX_MESSAGE_SIZE,  
/*out*/ BLACKBOARD_ID_TYPE        *BLACKBOARD_ID,  
/*out*/ RETURN_CODE_TYPE          *RETURN_CODE)
```

### Description

The `CREATE_BLACKBOARD` service request is used to create a blackboard named `BLACKBOARD_NAME`. The parameter `MAX_MESSAGE_SIZE` defines the maximum size of the message that can be communicated through the blackboard.

`BLACKBOARD_ID` is assigned by the OS and returned to the calling process. Processes can create as many buffers as the preallocated memory space will support.

### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_CONFIG	Not enough available storage space for the creation of the specified blackboard or maximum number of blackboards has been created
NO_ACTION	The blackboard named <code>BLACKBOARD_NAME</code> has already been created.
INVALID_PARAM	<code>MAX_MESSAGE_SIZE</code> zero or negative.
INVALID_MODE	Operating mode is NORMAL.
INVALID_MODE	No ARINC 653 partition associated with current thread (Development mode only).
INVALID_PARAM	A NULL pointer is specified as <code>BLACKBOARD_ID</code> (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## DISPLAY\_BLACKBOARD (P1:3.7.2.2.2)

### Synopsis

```
void DISPLAY_BLACKBOARD(  
/*in */ BLACKBOARD_ID_TYPE          BLACKBOARD_ID,  
/*in */ MESSAGE_ADDR_TYPE           MESSAGE_ADDR,  
/*in */ MESSAGE_SIZE_TYPE           LENGTH,  
/*out*/ RETURN_CODE_TYPE            *RETURN_CODE)
```

## Description

The `DISPLAY_BLACKBOARD` service request is used to display a message of size `LENGTH` in the blackboard `BLACKBOARD_ID`. The message to display is stored in `MESSAGE_ADDR`. On success, the specified blackboard becomes not empty. The message overwrites the previous one.

## Diagnostics

Return Code Value	Error Condition
<code>NO_ERROR</code>	Successful completion.
<code>INVALID_PARAM</code>	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"><li>• <code>BLACKBOARD_ID</code> does not identify an existing blackboard.</li><li>• <code>LENGTH</code> is greater than <code>MAX_MESSAGE_SIZE</code> specified for the blackboard.</li><li>• <code>LENGTH</code> is zero or negative</li></ul>
<code>INVALID_MODE</code>	No ARINC 653 partition associated with current thread (Development mode only).
<code>INVALID_PARAM</code>	A <code>NULL</code> pointer is passed for <code>MESSAGE_ADDR</code> (Development mode only).

## Conformance

This service is implemented according to the ARINC 653-1 standard.

---

**NOTE:** `DISPLAY_BLACKBOARD` takes `LENGTH` as an input parameter. If the blackboard is used for text messages, the developer has to keep in mind that `LENGTH` has to be large enough to account for the terminating zero symbol `\0`. This is due to the way C language handles strings. For example, the following call to `DISPLAY_BLACKBOARD` is valid:

```
DISPLAY_BLACKBOARD(BlackboardId, (MESSAGE_ADDR_TYPE)"TRUE", 5, &ReturnCode);
```

This note does not apply to binary messages.

---

## READ\_BLACKBOARD (P1:3.7.2.2.3)

### Synopsis

```
void READ_BLACKBOARD(  
/*in */ BLACKBOARD_ID_TYPE      BLACKBOARD_ID,  
/*in */ SYSTEM_TIME_TYPE        TIME_OUT,  
/*in */ MESSAGE_ADDR_TYPE       MESSAGE_ADDR,  
/*out */ MESSAGE_SIZE_TYPE       *LENGTH,
```

```
/*out */ RETURN_CODE_TYPE          *RETURN_CODE)
```

### Description

The `READ_BLACKBOARD` service request is used to read a message in the blackboard `BLACKBOARD_ID`. The calling process will be in the `WAITING` state while the blackboard is empty for a `TIME_OUT` maximum duration. The received message is of size `LENGTH` and is stored in `MESSAGE_ADDR`.

---

**NOTE:** This service request does not specify the amount of memory available at the `MESSAGE_ADDR` address. It is assumed that the biggest message as it is configured can be safely accommodated. The maximum message size can be queried with the `GET_BLACKBOARD_STATUS` request.

---

### Diagnostics

Return Code Value	Error Condition
<code>NO_ERROR</code>	Successful completion.
<code>INVALID_PARAM</code>	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"> <li>• <code>BLACKBOARD_ID</code> does not identify an existing blackboard.</li> <li>• <code>TIME_OUT</code> is out of range.</li> </ul>
<code>INVALID_MODE</code>	(Preemption is disabled or process is error handler process) and <code>TIME_OUT</code> is not zero.
<code>NOT_AVAILABLE</code>	No message in the blackboard.
<code>TIMED_OUT</code>	The specified <code>TIME_OUT</code> has expired.
<code>INVALID_MODE</code>	<ul style="list-style-type: none"> <li>• No ARINC 653 partition/process associated with current thread (Development mode only)</li> <li>• Partition mode not <code>NORMAL</code> (Development mode only).</li> </ul>
<code>INVALID_PARAM</code>	<ul style="list-style-type: none"> <li>• A <code>NULL</code> pointer is passed for <code>MESSAGE_ADDR</code> (Development mode only).</li> <li>• A <code>NULL</code> is passed for <code>LENGTH</code> (Development mode only).</li> </ul>

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## CLEAR\_BLACKBOARD (P1:3.7.2.2.4)

### Synopsis

```
void CLEAR_BLACKBOARD(  
    /*in */ BLACKBOARD_ID_TYPE    BLACKBOARD_ID,  
    /*out*/ RETURN_CODE_TYPE      *RETURN_CODE)
```

### Description

The CLEAR\_BLACKBOARD service request is used to clear the message in the blackboard BLACKBOARD\_ID. The specified blackboard becomes empty.

### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_PARAM	BLACKBOARD_ID does not identify an existing blackboard.
INVALID_MODE	No ARINC 653 partition associated with current thread (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## GET\_BLACKBOARD\_ID (P1:3.7.2.2.5)

### Synopsis

```
void GET_BLACKBOARD_ID(  
    /*in */ BLACKBOARD_NAME_TYPE    BLACKBOARD_NAME,  
    /*out*/ BLACKBOARD_ID_TYPE      *BLACKBOARD_ID,  
    /*out*/ RETURN_CODE_TYPE        *RETURN_CODE)
```

### Description

The GET\_BLACKBOARD\_ID service request allows the current process to get the identifier BLACKBOARD\_ID of the blackboard named BLACKBOARD\_NAME.

### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_CONFIG	There is no current partition blackboard named BLACKBOARD_NAME.

INVALID_PARAM	A NULL pointer passed for BLACKBOARD_ID (Development mode only).
INVALID_MODE	No ARINC 653 partition associated with current thread (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## GET\_BLACKBOARD\_STATUS (P1:3.7.2.2.6)

### Synopsis

```
void GET_BLACKBOARD_STATUS(  
    /*in */ BLACKBOARD_ID_TYPE      BLACKBOARD_ID,  
    /*out*/ BLACKBOARD_STATUS_TYPE  *BLACKBOARD_STATUS,  
    /*out*/ RETURN_CODE_TYPE        *RETURN_CODE)
```

### Description

The GET\_BLACKBOARD\_STATUS service request returns the status BLACKBOARD\_STATUS of the blackboard BLACKBOARD\_ID.

### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_PARAM	BLACKBOARD_ID does not identify an existing blackboard.
INVALID_MODE	No ARINC 653 partition associated with current thread (Development mode only).
INVALID_PARAM	A NULL is passed as BLACKBOARD_STATUS (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## Semaphore Services (P1:3.7.2.3)

A counting semaphore is a synchronization object commonly used to provide access to partition resources. A semaphore must be created during the initialization



mode before it can be used. A name is given at semaphore creation. This name is local to the partition and is not an attribute of the partition configuration table.

### CREATE\_SEMAPHORE (P1:3.7.2.3.1)

#### Synopsis

```
void CREATE_SEMAPHORE(
/*in */ SEMAPHORE_NAME_TYPE      SEMAPHORE_NAME,
/*in */ SEMAPHORE_VALUE_TYPE     CURRENT_VALUE,
/*in */ SEMAPHORE_VALUE_TYPE     MAXIMUM_VALUE,
/*in */ QUEUING_DISCIPLINE_TYPE  QUEUING_DISCIPLINE,
/*out*/ SEMAPHORE_ID_TYPE        *SEMAPHORE_ID,
/*out*/ RETURN_CODE_TYPE         *RETURN_CODE)
```

#### Description

The `CREATE_SEMAPHORE` service request is used to create a semaphore named `SEMAPHORE_NAME`. An identifier `SEMAPHORE_ID` is assigned by the OS and returned to the calling process. The `MAXIMUM_VALUE` parameter is the maximum value that the semaphore can be signaled to. The `CURRENT_VALUE` is the semaphore's starting value after creation. For example, if the semaphore was used to manage access to five resources, and at the time of creation three resources were available, the semaphore would be created with a maximum value of five and a current value of three. The `QUEUING_DISCIPLINE` parameter indicates the process queuing policy (FIFO or priority order) associated with that semaphore.

#### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
NO_ACTION	The semaphore named <code>SEMAPHORE_NAME</code> has already been created.
INVALID_PARAM	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"> <li>• <code>CURRENT_VALUE</code> is less than zero or greater than <code>MAX_SEMAPHORE_VALUE</code>.</li> <li>• <code>MAXIMUM_VALUE</code> is less than zero or greater than <code>MAX_SEMAPHORE_VALUE</code>.</li> <li>• <code>CURRENT_VALUE</code> is greater than <code>MAXIMUM_VALUE</code>.</li> <li>• <code>QUEUING_DISCIPLINE</code> is not valid.</li> </ul>
INVALID_CONFIG	Implementation limits on the number of semaphores exceeded.
INVALID_MODE	Operating mode is NORMAL.

INVALID_MODE	No ARINC 653 partition associated with current thread (Development mode only).
INVALID_PARAM	A NULL pointer is specified as SEMAPHORE_ID (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard with the following exceptions:

- Due to design restrictions, the only available queuing discipline is priority order. The request for the FIFO queuing discipline is ignored, and the priority-based one is used instead.

### Rationale for deviations:

It is impossible to implement FIFO queuing of processes using existing LynxOS-178 primitives. Modifying the LynxOS-178 standard method of blocking a process would break the existing LynxOS-178 certification.

## WAIT\_SEMAPHORE (P1:3.7.2.3.2)

### Synopsis

```
void WAIT_SEMAPHORE(  
/*in */ SEMAPHORE_ID_TYPE SEMAPHORE_ID,  
/*in */ SYSTEM_TIME_TYPE TIME_OUT,  
/*out*/ RETURN_CODE_TYPE *RETURN_CODE)
```

### Description

The `WAIT_SEMAPHORE` service request moves the current process from the `RUNNING` state to the `WAITING` state if the current value of the semaphore `SEMAPHORE_ID` is zero and if the specified `TIME_OUT` is not null. The process goes on executing if the current value of the specified semaphore is positive, and the semaphore current value is decremented.

### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_PARAM	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"><li>• <code>SEMAPHORE_ID</code> does not identify an existing semaphore.</li><li>• <code>TIME_OUT</code> is out of range.</li></ul>

INVALID_MODE	(Preemption is disabled or process is error handler process) and TIME_OUT is not zero.
NOT_AVAILABLE	Current value of SEMAPHORE_ID is not zero and TIME_OUT is zero.
TIMED_OUT	The specified TIME_OUT has expired.
INVALID_MODE	<ul style="list-style-type: none"> <li>• No ARINC 653 partition/process associated with current thread (Development mode only).</li> <li>• Partition mode not NORMAL (Development mode only).</li> </ul>

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## SIGNAL\_SEMAPHORE (P1:3.7.2.3.3)

### Synopsis

```
void SIGNAL_SEMAPHORE(
/*in */ SEMAPHORE_ID_TYPE      SEMAPHORE_ID,
/*out*/ RETURN_CODE_TYPE      *RETURN_CODE)
```

### Description

The SIGNAL\_SEMAPHORE service request increments the current value of the semaphore SEMAPHORE\_ID. If processes are waiting on that semaphore, the first process of the queue is moved from the WAITING state to the READY state. A scheduling takes place.

### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_PARAM	SEMAPHORE_ID does not identify an existing semaphore.
NO_ACTION	Current value of the semaphore is equal to maximum value.
INVALID_MODE	No ARINC 653 partition associated with current thread (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

### GET\_SEMAPHORE\_ID (P1:3.7.2.3.4)

#### Synopsis

```
void GET_SEMAPHORE_ID(  
    /*in */ SEMAPHORE_NAME_TYPE    SEMAPHORE_NAME,  
    /*out*/ SEMAPHORE_ID_TYPE       *SEMAPHORE_ID,  
    /*out*/ RETURN_CODE_TYPE        *RETURN_CODE)
```

#### Description

The GET\_SEMAPHORE\_ID service request allows the current process to get the identifier SEMAPHORE\_ID of the semaphore named SEMAPHORE\_NAME.

#### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_CONFIG	There is no current partition semaphore named SEMAPHORE_NAME.
INVALID_PARAM	A NULL pointer passed for SEMAPHORE_ID (Development mode only).
INVALID_MODE	No ARINC 653 partition associated with current thread (Development mode only).

#### Conformance

This service is implemented according to the ARINC 653-1 standard.

### GET\_SEMAPHORE\_STATUS (P1:3.7.2.3.5)

#### Synopsis

```
void GET_SEMAPHORE_STATUS(  
    /*in */ SEMAPHORE_ID_TYPE    SEMAPHORE_ID,  
    /*out*/ SEMAPHORE_STATUS_TYPE *SEMAPHORE_STATUS,  
    /*out*/ RETURN_CODE_TYPE        *RETURN_CODE)
```

#### Description

The GET\_SEMAPHORE\_STATUS service request returns the status SEMAPHORE\_STATUS of the semaphore SEMAPHORE\_ID. The status structure contains the CURRENT\_VALUE, MAXIMUM\_VALUE, and the number of processes WAITING\_PROCESSES waiting on the semaphore.

## Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_PARAM	SEMAPHORE_ID does not identify an existing semaphore.
INVALID_PARAM	A NULL pointer passed for SEMAPHORE_STATUS (Development mode only).
INVALID_MODE	No ARINC 653 partition associated with current thread (Development mode only).

## Conformance

This service is implemented according to the ARINC 653-1 standard.

## Event Services (P1:3.7.2.4)

An event is a synchronization object used to notify the occurrence of a condition to processes that may wait for it. An event must be created during the initialization mode before it can be used. A name is given at event creation. This name is local to the partition and is not an attribute of the partition configuration table.

### CREATE\_EVENT (P1:3.7.2.4.1)

#### Synopsis

```
void CREATE_EVENT(
/*in */ EVENT_NAME_TYPE    EVENT_NAME,
/*out*/ EVENT_ID_TYPE      *EVENT_ID,
/*out*/ RETURN_CODE_TYPE   *RETURN_CODE)
```

#### Description

The CREATE\_EVENT service request creates an event object named EVENT\_NAME for use by any of the processes in the partition. Upon creation, the event is set to the state DOWN. An identifier EVENT\_ID is assigned by the OS and returned to the calling process.

## Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.

INVALID_CONFIG	The maximum number of events has been created.
NO_ACTION	The event named EVENT_NAME has already been created.
INVALID_MODE	Operating mode is NORMAL.
INVALID_MODE	No ARINC 653 partition associated with current thread (Development mode only).
INVALID_PARAM	A NULL pointer is passed as EVENT_ID (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## SET\_EVENT (P1:3.7.2.4.2)

### Synopsis

```
void SET_EVENT(  
    /*in */ EVENT_ID_TYPE    EVENT_ID,  
    /*out*/ RETURN_CODE_TYPE *RETURN_CODE)
```

### Description

The SET\_EVENT service request sets the event EVENT\_ID to the state UP. All the processes waiting on that event are moved from the WAITING state to the READY state (unless suspended while waiting on the event), and a scheduling takes place. If processes were waiting on the event, the processes will be released on a priority followed by FIFO (when priorities are equal) basis.

### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_PARAM	EVENT_ID does not identify an existing event.
INVALID_MODE	No ARINC 653 partition associated with current thread (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

### RESET\_EVENT (P1:3.7.2.4.3)

#### Synopsis

```
void RESET_EVENT(  
    /*in */ EVENT_ID_TYPE      EVENT_ID,  
    /*out*/ RETURN_CODE_TYPE    *RETURN_CODE)
```

#### Description

The `RESET_EVENT` service request sets the event `EVENT_ID` to the state `DOWN`.

#### Diagnostics

Return Code Value	Error Condition
<code>NO_ERROR</code>	Successful completion.
<code>INVALID_PARAM</code>	<code>EVENT_ID</code> does not identify an existing event.
<code>INVALID_MODE</code>	No ARINC 653 partition associated with current thread

#### Conformance

This service is implemented according to the ARINC 653-1 standard.

### WAIT\_EVENT (P1:3.7.2.4.4)

#### Synopsis

```
void WAIT_EVENT(  
    /*in */ EVENT_ID_TYPE      EVENT_ID,  
    /*in */ SYSTEM_TIME_TYPE    TIME_OUT,  
    /*out*/ RETURN_CODE_TYPE    *RETURN_CODE)
```

#### Description

The `WAIT_EVENT` service request moves the current process from the `RUNNING` state to the `WAITING` state if the event `EVENT_ID` is in the `DOWN` state and if the specified `TIME_OUT` is not zero. The process goes on executing if the specified event is in the state `UP` or if it is a conditional wait (that is, event is down and `TIME_OUT` is zero).

#### Diagnostics

Return Code Value	Error Condition
<code>NO_ERROR</code>	Successful completion.

INVALID_PARAM	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"> <li>• EVENT_ID does not identify an existing event.</li> <li>• TIME_OUT is out of range.</li> </ul>
INVALID_MODE	(Preemption is disabled or process is error handler process) and TIME_OUT is not zero.
NOT_AVAILABLE	The event is in the DOWN state and TIME_OUT is 0.
TIMED_OUT	The specified TIME_OUT has expired.
INVALID_MODE	<ul style="list-style-type: none"> <li>• No ARINC 653 partition/process associated with current thread (Development mode only).</li> <li>• Partition mode is not NORMAL (Development mode only).</li> </ul>

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## GET\_EVENT\_ID (P1:3.7.2.4.5)

### Synopsis

```

void GET_EVENT_ID(
/*in */ EVENT_NAME_TYPE      EVENT_NAME,
/*out*/ EVENT_ID_TYPE        *EVENT_ID,
/*out*/ RETURN_CODE_TYPE     *RETURN_CODE)

```

### Description

The GET\_EVENT\_ID service allows the current process to get the identifier EVENT\_ID of an event named EVENT\_NAME.

### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_CONFIG	There is no current partition event named EVENT_NAME.
INVALID_MODE	No ARINC 653 partition associated with current thread (Development mode only).
INVALID_PARAM	A NULL pointer is passed as EVENT_ID (Development mode only).



## Conformance

This service is implemented according to the ARINC 653-1 standard.

### GET\_EVENT\_STATUS (P1:3.7.2.4.6)

#### Synopsis

```
void GET_EVENT_STATUS(
/*in */ EVENT_ID_TYPE      EVENT_ID,
/*out*/ EVENT_STATUS_TYPE  *EVENT_STATUS,
/*out*/ RETURN_CODE_TYPE   *RETURN_CODE)
```

#### Description

The GET\_EVENT\_STATUS service request returns the status EVENT\_STATUS of the event EVENT\_ID. The status structure contains the event state and the number of waiting processes.

#### Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_PARAM	EVENT_ID does not identify an existing event.
INVALID_MODE	No ARINC 653 partition associated with current thread (Development mode only).
INVALID_PARAM	A NULL pointer is passed as EVENT_STATUS (Development status).

## Conformance

This service is implemented according to the ARINC 653-1 standard.

## Memory Blocks (P2:3.9)

This section contains types and specifications related to the Memory Blocks service. The scope of the Memory Blocks service is restricted to its partition.

### GET\_MEMORY\_BLOCK\_STATUS (P2:3.9.2.1)

#### Synopsis

```
void GET_MEMORY_BLOCK_STATUS (
/* in */ MEMORY_BLOCK_NAME_TYPE  MEMORY_BLOCK_NAME,
/* out */ MEMORY_BLOCK_STATUS_TYPE *MEMORY_BLOCK_STATUS,
/* out */ RETURN_CODE_TYPE        *RETURN_CODE)
```

## Description

The `GET_MEMORY_BLOCK_STATUS` service request returns the attributes of a Memory Block. Partitions can only receive the status for Memory Block for which they have been granted access via the configuration tables.

## Diagnostics

Return Code Value	Error Condition
<code>NO_ERROR</code>	Successful completion.
<code>INVALID_CONFIG</code>	No memory block assigned to the partition is named <code>MEMORY_BLOCK_NAME</code> in the configuration
<code>INVALID_PARAM</code>	A NULL pointer passed for <code>MEMORY_BLOCK_STATUS</code> (Development mode only).
<code>INVALID_MODE</code>	No ARINC 653 partition associated with current thread (development mode only).

## Conformance

This service is implemented according to the ARINC 653-2 standard.

## Health Monitoring (P1:3.8)

The Health Monitor (HM) is invoked by an application calling the `RAISE_APPLICATION_ERROR` service or by the OS or hardware detecting a fault.

Health monitoring is implemented using the standard LynxOS-178 `hm` driver. There are some differences, however, in the usage of the ARINC 653-1 and LynxOS-178 health monitoring facilities:

- If an error handler process is created in a partition, the `hm` driver relegates error handling to that process. The error handler is required to take some action for a failing process, such as stopping the process or shutting down a partition. Note that LynxOS-178 does not terminate a failing process with a signal if the `hm` driver is installed. If no action is taken, a failing process may loop forever.
- ARINC 653 device driver can store a limited number of error messages in its internal queue. The size of the queue is configurable in the ARINC 653 device information file. If error events occur faster than the error handler process reads them, a VM-fatal exception is raised to the health monitoring driver.

## REPORT\_APPLICATION\_MESSAGE (P1:3.8.2.1)

### Synopsis

```
void REPORT_APPLICATION_MESSAGE(  
/*in */ MESSAGE_ADDR_TYPE      MESSAGE_ADDR,  
/*in */ MESSAGE_SIZE_TYPE      LENGTH,  
/*out*/ RETURN_CODE_TYPE       *RETURN_CODE)
```

### Description

The `REPORT_APPLICATION_MESSAGE` service request allows the current partition to transmit a message to the HM function if it detects an erroneous behavior. The message to be transmitted is of size `LENGTH` and is stored in `MESSAGE_ADDR`.

### Diagnostics

Return Code Value	Error Condition
<code>NO_ERROR</code>	Successful completion.
<code>INVALID_PARAM</code>	Length is <code>LENGTH</code> is less than zero or greater than or equal to <code>MAX_ERROR_STRING_SIZE</code> .
<code>INVALID_PARAM</code>	A <code>NULL</code> pointer is passed as a message address (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-1 standard.

## CREATE\_ERROR\_HANDLER (P1:3.8.2.2)

### Synopsis

```
void CREATE_ERROR_HANDLER(  
/*in */ SYSTEM_ADDRESS_TYPE      ENTRY_POINT,  
/*in */ STACK_SIZE_TYPE          STACK_SIZE,  
/*out*/ RETURN_CODE_TYPE         *RETURN_CODE)
```

### Description

The `CREATE_ERROR_HANDLER` service request creates an error handler process for the current partition. This process has no identifier (`ID`) and cannot be accessed by the other processes of the partition.

## Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
NO_ACTION	Error handler is already created.
INVALID_CONFIG	Stack size is out of range.
INVALID_MODE	Operating mode is NORMAL.

## Conformance

This service is implemented according to the ARINC 653-1 standard with the following exceptions:

- The error when `RETURN_CODE` is set to `INVALID_CONFIG` and if there is not enough memory will never occur.

Rationale for deviations:

This error is not implemented in production mode. Error handler is implemented as an extra thread and it is required for every partition. It must be taken into account during system configuration.

## GET\_ERROR\_STATUS (P1:3.8.2.3)

### Synopsis

```
void GET_ERROR_STATUS(  
    /*out*/ ERROR_STATUS_TYPE *ERROR_STATUS,  
    /*out*/ ERROR_CODE_TYPE *RETURN_CODE)
```

### Description

The `GET_ERROR_STATUS` service must be used by the error handler process to determine the error code, the identifier of the faulty process, the address at which the error occurs, and the message associated with the fault.

The errors are stored temporarily in a FIFO queue by the HM when they occur and are removed by the `GET_ERROR_STATUS` service requests in the FIFO order. If the error was raised by the application with the `RAISE_APPLICATION_ERROR`, the message is the one provided by the application; otherwise, the message is the one provided by LynxOS-178.

## Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_CONFIG	The current process is not the error handler.
NO_ACTION	There is no process in error.

## Conformance

This service is implemented according to the ARINC 653-1 standard.

## RAISE\_APPLICATION\_ERROR (P1:3.8.2.4)

### Synopsis

```
void RAISE_APPLICATION_ERROR(  
    /*in */ ERROR_CODE_TYPE      ERROR_CODE,  
    /*in */ MESSAGE_ADDR_TYPE    MESSAGE_ADDR,  
    /*in */ ERROR_MESSAGE_SIZE_TYPE LENGTH,  
    /*out*/ RETURN_CODE_TYPE     *RETURN_CODE)
```

### Description

The `RAISE_APPLICATION_ERROR` service request allows the current partition to invoke the error handler process for the specific error code `APPLICATION_ERROR`. The message to transmit is of size `LENGTH` and is stored in `MESSAGE_ADDR`. It will be read with the `GET_ERROR_STATUS` service call. The error handler of the partition is then started (if created) to take the recovery action for the process that raises the error code; otherwise (the error handler is not created), the error is considered at partition level error.

## Diagnostics

Return Code Value	Error Condition
NO_ERROR	Successful completion.
INVALID_PARAM	The return code corresponds to one of the following conditions: <ul style="list-style-type: none"><li>• <code>LENGTH</code> is negative or is greater than <code>MAX_ERROR_MESSAGE_SIZE</code>.</li><li>• <code>ERROR_CODE</code> is not <code>APPLICATION_ERROR</code>.</li></ul>

## Conformance

This service is implemented according to the ARINC 653-1 standard.

---

**NOTE:** `RAISE_APPLICATION_ERROR` takes `LENGTH` as an input parameter. If text messages are used, the developer has to keep in mind that `LENGTH` has to be large enough to account for the terminating zero symbol `\0`. This is due to the way C language handles strings. For example, the following call to

`RAISE_APPLICATION_ERROR` is valid:

```
RAISE_APPLICATION_ERROR (APPLICATION_ERROR, (MESSAGE_ADDR_TYPE) "111", 4,
&rc);
```

This note does not apply to binary messages.

---

## File System Services (P2:3.2.5)

This section specifies the service requests related to the ARINC 653 File System.

### OPEN\_NEW\_FILE (P2:3.2.5.1)

#### Synopsis

```
void OPEN_NEW_FILE (
/*in*/ FILE_NAME_TYPE          FILE_NAME,
/*out */ FILE_ID_TYPE          *FILE_ID,
/*out */ RETURN_CODE_TYPE      *RETURN_CODE,
/*inout*/ FILE_ERRNO_TYPE      *ERRNO)
```

#### Description

The `OPEN_NEW_FILE` service request creates and opens the file specified by its absolute file name and provides the identifier for the new file to permit subsequent read and write to the file. A partition must have write permissions for the requested file's volume in order to create the file.

Files created with this service do not already exist.

The file's creation time is set to the current composite time value.

#### Diagnostics

Return Code Value	ERRNO	Error Condition
NO_ERROR		Successful completion.
INVALID_CONFIG	EMFILE	Maximum number of open files/directories is reached by the current partition

INVALID_PARAM	ENAMETOOLONG	FILE_NAME or one of its components exceed maximum character length
INVALID_PARAM	EINVAL	FILE_NAME is syntactically invalid
INVALID_PARAM	ENODIR	A component of the path prefix of FILE_NAME is not a volume or directory
INVALID_CONFIG	EACCESS	The current partition does not have read-write access rights for the volume that would contain FILE_NAME
INVALID_PARAM	EROFS	The storage device that would FILE_NAME is currently write protected
INVALID_PARAM	EEXIST	FILE_NAME is an existing file
INVALID_PARAM	EISDIR	FILE_NAME is an existing directory
INVALID_CONFIG	ENOSPC	There is not enough space available on volume Storage device containing FILE_NAME reports a failure
NOT_AVAILABLE	EIO	Storage device containing FILE_NAME reports a failure
INVALID_MODE	EACCESS	Preemption is disabled or the current process is the error handler
INVALID_PARAM		A NULL pointer is passed as is passed ERRNO (Development mode only).
INVALID_PARAM	EINVAL	A NULL pointer is passed for FILE_ID (Development mode only).
INVALID_MODE	EACCES	No ARINC 653 partition associated with current thread (Development mode only).
INVALID_CONFIG	EMFILE	No open file handles available (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-2 standard with the following exception:

- File change time (st\_ctime) is used to implement creation time that may lead inaccurate result.

### OPEN\_FILE (P2:3.2.5.2)

#### Synopsis

```

void OPEN_FILE (
/*in      */ FILE_NAME_TYPE      FILE_NAME,
/*in      */ FILE_MODE_TYPE      FILE_MODE,
/*out     */ FILE_ID_TYPE        *FILE_ID,
/*out     */ RETURN_CODE_TYPE    *RETURN_CODE,
/*inout   */ FILE_ERRNO_TYPE     *ERRNO)

```

### Description

The `OPEN_FILE` service request opens the file specified by its full pathname and provides the identifier of the file to permit subsequent read and write to the file.

Files opened with this service must already exist. The position indicator of the file initially points to the `Beginning_Of_File`. There can be only one open in stance of a file for write access at a time.

### Diagnostics

Return Code Value	ERRNO	Error Condition
NO_ERROR		Successful completion.
INVALID_CONFIG	EMFILE	Maximum number of open files/directories is reached by the current partition
INVALID_PARAM	ENAMETOOLONG	FILE_NAME or one of its components exceed maximum character length
INVALID_PARAM	EINVAL	FILE_NAME is syntactically invalid
INVALID_PARAM	ENODIR	A component of the path prefix of FILE_NAME is not a volume or directory
INVALID_PARAM	EINVAL	FILE_NAME does not represent an existing access mode
INVALID_PARAM	EISDIR	FILE_NAME is an existing directory
INVALID_PARAM	ENOENT	FILE_NAME is not an existing file
INVALID_MODE	EACCESS	FILE_NAME is an existing file currently open as read-write and FILE_MODE is read-write
INVALID_MODE	EACCESS	The current partition does not have the access rights represented by FILE_MODE for the volume containing FILE_NAME
INVALID_PARAM	EROFS	The storage device that would FILE_NAME is currently write protected
NOT_AVAILABLE	EIO	Storage device containing FILE_NAME reports a failure



INVALID_MODE	EACCESS	Preemption is disabled or the current process is the error handler
INVALID_PARAM		A NULL pointer is passed for ERRNO (Development mode only).
INVALID_MODE	EACCES	No ARINC 653 partition associated with current thread (Development mode only).
INVALID_CONFIG	EMFILE	No open file handles available (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-2 standard.

### CLOSE\_FILE (P2:3.2.5.3)

#### Synopsis

```
void CLOSE_FILE (
/*in      */ FILE_ID_TYPE      FILE_ID,
/*out     */ RETURN_CODE_TYPE  *RETURN_CODE,
/*inout   */ FILE_ERRNO_TYPE  *ERRNO)
```

#### Description

The CLOSE\_FILE service request signals the end of read or write activities. The associated file identifier is de-allocated.

#### Diagnostics

Return Code Value	ERRNO	Error Condition
NO_ERROR		Successful completion.
INVALID_CONFIG	EBADF	FILE_ID is not an open file identifier for the current partition
NOT_AVAILABLE	EIO	The storage device containing FILE_ID
NOT_AVAILABLE	EBUSY	FILE_ID has an operation in progress
INVALID_MODE	EACCESS	Preemption is disabled or the current process is the error handler
INVALID_PARAM		A NULL pointer is passed for ERRNO (Development mode only).

INVALID_MODE	EACCES	No ARINC 653 partition associated with current thread (Development mode only).
INVALID_PARAM	EBADF	Invalid file descriptor for ARINC file (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-2 standard.

## READ\_FILE (P2:3.2.5.4)

### Synopsis

```
void READ_FILE (
/*in      */ FILE_ID_TYPE      FILE_ID,
/*in      */ MESSAGE_ADDR_TYPE MESSAGE_ADDR,
/*in      */ MESSAGE_SIZE_TYPE IN_LENGTH,
/*out     */ MESSAGE_SIZE_TYPE *OUT_LENGTH,
/*out     */ RETURN_CODE_TYPE *RETURN_CODE,
/*inout   */ FILE_ERRNO_TYPE *ERRNO)
```

### Description

The `READ_FILE` service request attempts to read a message of the designated number of bytes pointed by the position indicator from the specified file. The number of bytes actually read is returned. It can be different from the number of bytes to read if `End_Of_File` is encountered before having read all the specified bytes. No data transfer occurs past the `End_Of_File`. If the starting position is at or after the `End_Of_File`, the number of bytes read is zero. The position indicator is advanced by the number of bytes read.

### Diagnostics

Return Code Value	ERRNO	Error Condition
NO_ERROR		Successful completion.
INVALID_CONFIG	EBADF	<code>FILE_ID</code> is not an open file identifier for the current partition
NOT_AVAILABLE	EBUSY	<code>FILE_ID</code> has an operation in progress
NOT_AVAILABLE	ESTALE	<code>FILE_ID</code> is no longer valid due to the file owner's action
INVALID_PARAM	EINVAL	<code>IN_LENGTH</code> is zero or negative
INVALID_PARAM	EFBIG	<code>IN_LENGTH</code> is greater than maximum atomicity

NOT_AVAILABLE	EIO	The storage device containing FILE_ID reports a failure
NOT_AVAILABLE	EIO	I/O is in error
NOT_AVAILABLE	EOVERFLOW	file POSITION is greater than file SIZE
INVALID_MODE	EACCESS	Preemption is disabled or the current process is the error handler
INVALID_PARAM		A NULL pointer is passed for ERRNO (Development mode only).
INVALID_PARAM	EINVAL	A NULL pointer is passed for MESSAGE_ADDR or OUT_LENGTH (Development mode only).
INVALID_MODE	EACCES	No ARINC 653 partition associated with current thread (Development mode only).
INVALID_PARAM	EBADF	Invalid file descriptor for ARINC file (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-2 standard with the following exceptions:

- The service never returns the NOT\_AVAILABLE / ESTALE error.

Rationale for this deviation:

In LynxOS-178 ARINC FS operations implemented upon POSIX FS interface of Lynx file system. If file descriptor is no longer valid it will be detected by underlying POSIX calls and appropriate error (e.g. EBADF) will be set.

- The service never returns the NOT\_AVAILABLE / EOVERFLOW error.

Rationale for this deviation:

In LynxOS-178 ARINC FS operations implemented upon POSIX FS interface of Lynx file system. It is not possible to set file position greater than file size for ARINC process, this is handled at the kernel implementation (system calls) level.

## WRITE\_FILE (P2:3.2.5.5)

### Synopsis

```

void WRITE_FILE (
/*in      */ FILE_ID_TYPE FILE_ID,
/*in      */ MESSAGE_ADDR_TYPE MESSAGE_ADDR,
/*in      */ MESSAGE_SIZE_TYPE LENGTH,
/*out     */ RETURN_CODE_TYPE *RETURN_CODE,
/*inout*   */ FILE_ERRNO_TYPE *ERRNO)

```

### Description

The WRITE\_FILE service request writes a message of the designated number of bytes to the specified file at its position indicator. After the write operation is complete, the file position indicator is advanced by the number of bytes written. If this incremented position indicator is greater than the file size, the size of the file is set to this position indicator. The file's update time is set to the current composite time value.

### Diagnostics

Return Code Value	ERRNO	Error Condition
NO_ERROR		Successful completion.
INVALID_CONFIG	EBADF	FILE_ID is not an open file identifier for the current partition
NOT_AVAILABLE	EBUSY	FILE_ID has an operation in progress
INVALID_PARAM	EACCESS	FILE_ID is not opened with read-write access mode
INVALID_PARAM	EINVAL	IN_LENGTH is zero or negative
INVALID_PARAM	ENOSPC	Space required to allocate LENGTH bytes are not available for the volume
INVALID_PARAM	EFBIG	IN_LENGTH is greater than maximum atomicity
NOT_AVAILABLE	EIO	The storage device containing FILE_ID reports a failure
NOT_AVAILABLE	EIO	I/O is in error
INVALID_MODE	EACCESS	Preemption is disabled or the current process is the error handler
INVALID_PARAM		A NULL pointer is passed for ERRNO (Development mode only).
INVALID_PARAM	EINVAL	A NULL pointer is passed for MESSAGE_ADDR (Development mode only).

INVALID_MODE	EACCES	No ARINC 653 partition associated with current thread (Development mode only).
INVALID_PARAM	EBADF	Invalid file descriptor for ARINC file (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-2 standard with the following exceptions:

- NB\_OF\_CHANGES and NB\_OF\_WRITE\_ERRORS fields are not updated.

### Rationale for this deviation:

NB\_OF\_CHANGES and NB\_OF\_WRITE\_ERRORS fields are not updated as this data is not stored in Lynx FS which is used to implement ARINC file system services.

## SEEK\_FILE (P2:3.2.5.6)

### Synopsis

```
void SEEK_FILE (
/*in      */ FILE_ID_TYPE      FILE_ID,
/*in      */ FILE_SIZE_TYPE    OFFSET,
/*in      */ FILE_SEEK_TYPE    WHENCE,
/*out     */ FILE_SIZE_TYPE    *POSITION,
/*out     */ RETURN_CODE_TYPE  *RETURN_CODE,
/*inout   */ FILE_ERRNO_TYPE   *ERRNO)
```

### Description

The SEEK\_FILE service request sets the file position indicator according to offset value and seek type. This file position indicator is the location where subsequent reads or writes will take place.

### Diagnostics

Return Code Value	ERRNO	Error Condition
NO_ERROR		Successful completion.
INVALID_CONFIG	EBADF	FILE_ID is not an open file identifier for the current partition
NOT_AVAILABLE	EBUSY	FILE_ID has an operation in progress
NOT_AVAILABLE	ESTALE	FILE_ID is no longer valid due to the file owner's action

INVALID_PARAM	EINVAL	WHENCE does not represent a defined type
INVALID_PARAM	EINVAL	WHENCE is SEEK_SET and OFFSET is not in the range 0..file SIZE
INVALID_PARAM	EINVAL	WHENCE is SEEK_CUR and file POSITION + OFFSET is not in the range 0..file SIZE
INVALID_PARAM	EINVAL	WHENCE is SEEK_END and file SIZE + OFFSET is not in the range 0..file SIZE
NOT_AVAILABLE	EIO	The storage device containing FILE_ID reports a failure
INVALID_MODE	EACCESS	Preemption is disabled or the current process is the error handler
INVALID_PARAM		A NULL pointer is passed for ERRNO (Development mode only).
INVALID_PARAM	EINVAL	A NULL pointer is passed for POSITION (Development mode only).
INVALID_MODE	EACCES	No ARINC 653 partition associated with current thread (Development mode only).
INVALID_PARAM	EBADF	Invalid file descriptor for ARINC file (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-2 standard with the following exceptions:

- The service never returns the NOT\_AVAILABLE / ESTALE error.

Rationale for this deviation:

In LynxOS-178 ARINC FS operations implemented upon POSIX FS interface of Lynx file system. If file descriptor is no longer valid it will be detected by underlying POSIX calls and appropriate error (e.g. EBADF) will be set.

- The service never returns the NOT\_AVAILABLE / EIO error.

Rationale for this deviation:

In LynxOS-178 ARINC FS operations implemented upon POSIX FS interface of Lynx file system. In Lynx file system seek operation on a file does not imply any I/O operations on a disk.

## REMOVE\_FILE (P2:3.2.5.7)

### Synopsis

```
void REMOVE_FILE (
/*in      */ FILE_NAME_TYPE      FILE_NAME,
/*out     */ RETURN_CODE_TYPE    *RETURN_CODE,
/*inout   */ FILE_ERRNO_TYPE    *ERRNO)
```

### Description

The REMOVE\_FILE service request removes a file. Once removed, the file is no longer available to be opened.

### Diagnostics

Return Code Value	ERRNO	Error Condition
NO_ERROR		Successful completion.
INVALID_PARAM	ENAMETOOLONG	FILE_NAME or one of its components exceed maximum character length
INVALID_PARAM	EINVAL	FILE_NAME is syntactically invalid
INVALID_PARAM	ENODIR	A component of the path prefix of FILE_NAME is not a volume or directory
INVALID_PARAM	EPERM	FILE_NAME is an existing directory
INVALID_PARAM	ENOENT	FILE_NAME is not an existing file
INVALID_CONFIG	EACCES	The current partition does not have read-write access rights to the volume containing FILE_NAME
NOT_AVAILABLE	EBUSY	The file referenced by FILE_NAME has an operation in progress
INVALID_PARAM	EROFS	The storage device containing FILE_NAME is currently write protected
NOT_AVAILABLE	EBUSY	FILE_NAME is open for any access mode by the owning partition
NOT_AVAILABLE	EIO	The storage device containing FILE_ID reports a failure
INVALID_MODE	EACCES	Preemption is disabled or the current process is the error handler

INVALID_PARAM		A NULL pointer is passed for ERRNO (Development mode only).
INVALID_MODE	EACCES	No ARINC 653 partition associated with current thread (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-2 standard.

## RENAME\_FILE (P2:3.2.5.8)

### Synopsis

```

void RENAME_FILE (
/*in      */ FILE_NAME_TYPE      OLD_FILE_NAME,
/*in      */ FILE_NAME_TYPE      NEW_FILE_NAME,
/*out     */ RETURN_CODE_TYPE     *RETURN_CODE,
/*inout   */ FILE_ERRNO_TYPE     *ERRNO)

```

### Description

The `RENAME_FILE` service request renames the existing file with the specified new file name. A file can be given a new name in the same directory or another directory on the same volume (cannot rename to a different volume). The file's data contents are unaffected by the rename operation. Once renamed, the `OLD_FILE_NAME` name is no longer valid to open the file.

### Description

Return Code Value	ERRNO	Error Condition
NO_ERROR		Successful completion.
INVALID_PARAM	ENAMETOOLONG	OLD_FILE_NAME or one of its components exceed maximum character length
INVALID_PARAM	EINVAL	OLD_FILE_NAME is syntactically invalid
INVALID_PARAM	ENODIR	A component of the path prefix of OLD_FILE_NAME is not a volume or directory
INVALID_PARAM	ENAMETOOLONG	NEW_FILE_NAME or one of its components exceed maximum character length
INVALID_PARAM	EINVAL	NEW_FILE_NAME is syntactically invalid



INVALID_PARAM	ENODIR	A component of the path prefix of NEW_FILE_NAME is not a volume or directory
INVALID_CONFIG	EACCES	The current partition does not have read-write access rights to the volume containing OLD_FILE_NAME
NOT_AVAILABLE	EBUSY	OLD_FILE_NAME has an operation in progress
INVALID_PARAM	EROFS	The storage device containing OLD_FILE_NAME is currently write protected
INVALID_PARAM	EINVAL	The volume for OLD_FILE_NAME is not identical to the volume for NEW_FILE_NAME
INVALID_CONFIG	ENOSPC	Not able to create the file in the new directory (not enough resources or max number of files exceeded)
INVALID_PARAM	EPERM	OLD_FILE_NAME is an existing directory
INVALID_PARAM	EISDIR	NEW_FILE_NAME is an existing directory
INVALID_PARAM	ENOENT	OLD_FILE_NAME is not an existing file
INVALID_PARAM	EEXIST	OLD_FILE_NAME is an existing file
NOT_AVAILABLE	EBUSY	OLD_FILE_NAME is open for any access mode by the owning partition
NOT_AVAILABLE	EIO	The storage device containing OLD_FILE_NAME or NEW_FILE_NAME reports a failure
INVALID_MODE	EACCES	Preemption is disabled or the current process is the error handler

### Conformance

This service is implemented according to the ARINC 653-2 standard.

### GET\_FILE\_STATUS (P2:3.2.5.9)

#### Synopsis

```
void GET_FILE_STATUS (
/*in      */ FILE_ID_TYPE  FILE_ID,
```

```

/*out      */ FILE STATUS TYPE      *FILE STATUS,
/*out      */ RETURN CODE TYPE      *RETURN CODE,
/*inout    */ FILE_ERRNO_TYPE      *ERRNO)

```

### Description

The `GET_FILE_STATUS` service request obtains the statistics (e.g., position, size, etc.) of the specified file or file identifier.

### Diagnostics

Return Code Value	ERRNO	Error Condition
NO_ERROR		Successful completion.
INVALID_PARAM	EBADF	FILE_ID is not an open file identifier for the current partition
NOT_AVAILABLE	EBUSY	FILE_ID has an operation in progress
INVALID_MODE	EACCES	Preemption is disabled or the current process is the error handler
INVALID_PARAM		A NULL pointer is passed for ERRNO (Development mode only).
INVALID_PARAM	EINVAL	A NULL pointer is passed for FILE_STATUS (Development mode only).
INVALID_MODE	EACCES	No ARINC 653 partition associated with current thread (Development mode only).
INVALID_PARAM	EBADF	Invalid file descriptor for ARINC file (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-2 standard with the following exceptions:

- File creation time is not set.

Rationale for this deviation:

In LynxOS-178 ARINC FS is implemented upon Lynx file system which does not support file creation time data.

- The service never returns the NOT\_AVAILABLE/ESTALE error.

Rationale for this deviation:

In LynxOS-178 ARINC FS operations implemented upon POSIX FS interface of Lynx file system. If file descriptor is no longer valid it will be detected by underlying POSIX calls and appropriate error (e.g. EBADF) will be set.

- The service never returns the `NOT_AVAILABLE` / `EIO` error.

Rationale for this deviation:

Getting file parameters does not imply any disk I/O operations on underlying file system.

## GET\_VOLUME\_STATUS (P2:3.2.5.10)

### Synopsis

```
void GET_VOLUME_STATUS (
/*in      */ FILE_NAME_TYPE      FILE_NAME,
/*out     */ VOLUME_STATUS_TYPE  *VOLUME_STATUS,
/*out     */ RETURN_CODE_TYPE    *RETURN_CODE,
/*inout   */ FILE_ERRNO_TYPE     *ERRNO)
```

### Description

The `GET_VOLUME_STATUS` service request provides information about the file system of the volume containing the specified file.

The information includes:

- the size in bytes of a block (minimum allocation unit).
- the total number of bytes allocated to volume.
- the total number of bytes used, including overhead.
- the total number of bytes free.
- the configuration defined access rights for the partition for this volume.
- the configuration defined media type for this volume.

### Diagnostics

Return Code Value	ERRNO	Error Condition
<code>NO_ERROR</code>		Successful completion.
<code>INVALID_PARAM</code>	<code>ENAMETOOLONG</code>	<code>FILE_NAME</code> or one of its components exceed maximum character length
<code>INVALID_PARAM</code>	<code>EINVAL</code>	<code>FILE_NAME</code> is syntactically invalid

INVALID_PARAM	ENODIR	A component of the path prefix of FILE_NAME is not a volume or directory
INVALID_PARAM	ENOENT	FILE_NAME is not an existing file, directory, or volume
INVALID_CONFIG	EACCES	The current partition does not have read-write access rights to the volume containing FILE_NAME
NOT_AVAILABLE	EIO	The storage device containing FILE_ID reports a failure
INVALID_MODE	EACCES	Preemption is disabled or the current process is the error handler
INVALID_PARAM		A NULL pointer is passed for ERRNO (Development mode only).
INVALID_PARAM	EINVAL	A NULL pointer is passed for VOLUME_STATUS (Development mode only).
INVALID_MODE	EACCES	No ARINC 653 partition associated with current thread (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-2 standard.

## RESIZE\_FILE (P2:3.2.5.11)

### Synopsis

```
void RESIZE_FILE (  
/*in      */ FILE_ID_TYPE      FILE_ID,  
/*in      */ FILE_SIZE_TYP     NEW_SIZE,  
/*out     */ RETURN_CODE_TYPE  *RETURN_CODE,  
/*inout   */ FILE_ERRNO_TYPE   *ERRNO)
```

### Description

The RESIZE\_FILE service request changes the file size to a specified number of bytes. This service causes the regular file referenced by the file identifier to allow the extension (or contraction) of its size to a new size.

## Diagnostics

Return Code Value	ERRNO	Error Condition
NO_ERROR		Successful completion.
INVALID_PARAM	EBADF	FILE_ID is not an open file identifier for the current partition
NOT_AVAILABLE	EBUSY	FILE_ID has an operation in progress
INVALID_PARAM	EACCES	FILE_ID is a file opened as read-only
INVALID_PARAM	EINVAL	NEW_SIZE is not a proper value or the resulting file offset would be invalid
INVALID_PARAM	ENOSPC	Allocating NEW_SIZE exceeds the blocks allocated to the volume
INVALID_MODE	EACCES	Preemption is disabled or the current process is the error handler
INVALID_PARAM		A NULL pointer is passed for ERRNO (Development mode only).
INVALID_MODE	EACCES	No ARINC 653 partition associated with current thread (Development mode only).

## Conformance

This service is implemented according to the ARINC 653-2 standard with the following exception:

- The service never returns the NOT\_AVAILABLE / EIO error.

Rationale for this deviation:

Changing file parameters does not imply any disk I/O operations on underlying file system.

## SYNC\_FILE (P2:3.2.5.12)

### Synopsis

```
void SYNC_FILE (
/*in      */ FILE ID TYPE      FILE ID,
/*out     */ RETURN_CODE_TYPE  *RETURN_CODE,
/*inout   */ FILE ERRNO TYPE   *ERRNO)
```

### Description

The SYNC\_FILE service request causes data associated with the specified file be transferred to the associated storage device. The service does not return until the

transfer completes or an error is reported. This provides a means to potentially prevent data from being lost due to a power-loss. Just before closing a file, the file system will also perform a sync.

### Diagnostics

Return Code Value	ERRNO	Error Condition
NO_ERROR		Successful completion.
INVALID_PARAM	EBADF	FILE_ID is not an open file identifier for the current partition
NOT_AVAILABLE	EBUSY	FILE_ID has an operation in progress
INVALID_PARAM	EACCES	FILE_ID is a file opened as read-only
NOT_AVAILABLE	EIO	The storage device containing FILE_ID reports a failure
INVALID_MODE	EACCES	Preemption is disabled or the current process is the error handler
INVALID_PARAM		A NULL pointer is passed for ERRNO (Development mode only).
INVALID_MODE	EACCES	No ARINC 653 partition associated with current thread (Development mode only).
INVALID_PARAM	EBADF	Invalid file descriptor for ARINC file (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-2 standard.

## OPEN\_DIRECTORY (P2:3.2.5.13)

### Synopsis

```
void OPEN_DIRECTORY (
/*in      */ FILE_NAME_TYPE      DIRECTORY_NAME,
/*out     */ DIRECTORY_ID_TYPE   *DIRECTORY_ID,
/*out     */ RETURN_CODE_TYPE    *RETURN_CODE,
/*inout   */ FILE_ERRNO_TYPE     *ERRNO)
```

### Description

The OPEN\_DIRECTORY service request opens the directory specified by its full path name and provides the identifier of the directory to permit subsequent read from the

directory. Directories opened with this service must already exist. The position indicator of the directory points to the first directory entry.

#### Diagnostics

Return Code Value	ERRNO	Error Condition
NO_ERROR		Successful completion.
INVALID_CONFIG	EMFILE	The maximum number of open files/directories is reached by the current partition
INVALID_PARAM	ENAMETOOLONG	DIRECTORY_NAME or one of its components exceed maximum character length
INVALID_PARAM	EINVAL	DIRECTORY_NAME is syntactically invalid
INVALID_PARAM	ENOTDIR	A component of the path prefix of DIRECTORY_NAME is not a volume or directory
INVALID_PARAM	ENOENT	DIRECTORY_NAME is not an existing directory
INVALID_CONFIG	EACCES	The current partition does not have read access rights for the volume containing DIRECTORY_NAME
NOT_AVAILABLE	EIO	The storage device containing DIRECTORY_NAME reports a failure
INVALID_MODE	EACCES	Preemption is disabled or the current process is the error handler
INVALID_PARAM		A NULL pointer is passed for ERRNO (Development mode only).
INVALID_PARAM	EINVAL	A NULL pointer is passed for DIRECTORY_ID (Development mode only).
INVALID_MODE	EACCES	No ARINC653 partition associated with current thread (Development mode only).
INVALID_CONFIG	EMFILE	No open directory handles available (Development mode only).

## Conformance

This service is implemented according to the ARINC 653-2 standard.

## CLOSE\_DIRECTORY (P2:3.2.5.14)

### Synopsis

```
void CLOSE_DIRECTORY (
/*in      */ DIRECTORY_ID_TYPE    DIRECTORY_ID,
/*out     */ RETURN_CODE_TYPE    *RETURN_CODE,
/*inout   */ FILE_ERRNO_TYPE     *ERRNO)
```

### Description

The `CLOSE_DIRECTORY` service request signals the end of read activities and the associated directory identifier is de-allocated.

### Diagnostics

Return Code Value	ERRNO	Error Condition
NO_ERROR		Successful completion.
INVALID_PARAM	EBADF	<code>DIRECTORY_ID</code> is not an open directory identifier for the current partition
NOT_AVAILABLE	EBUSY	<code>DIRECTORY_ID</code> has an operation in progress
INVALID_MODE	EACCES	Preemption is disabled or the current process is the error handler
INVALID_PARAM		A NULL pointer is passed for <code>ERRNO</code> (Development mode only).
INVALID_MODE	EACCES	No ARINC 653 partition associated with current thread (Development mode only).
INVALID_PARAM	EBADF	Invalid directory file descriptor (Development mode only).

## Conformance

This service is implemented according to the ARINC 653-2 standard with the following exception:

- The service never returns the `NOT_AVAILABLE` / `EIO` error.

Rationale for this deviation:



Closing directory file does not imply any disk I/O operations on underlying file system (file is not changed).

## READ\_DIRECTORY (P2:3.2.5.15)

### Synopsis

```
void READ_DIRECTORY (
/*in      */ DIRECTORY_ID_TYPE    DIRECTORY_ID,
/*out     */ DIRECTORY_ENTRY_TYPE *ENTRY_NAME,
/*out     */ ENTRY_KIND_TYPE     *ENTRY_KIND,
/*out     */ RETURN_CODE_TYPE    *RETURN_CODE,
/*inout   */ FILE_ERRNO_TYPE     *ERRNO)
```

### Description

The READ\_DIRECTORY service request attempts to read the directory entry pointed by the position indicator from the specified directory and returns its associated filename.

The kind of entry (file, sub-directory, other or no more) is also returned. When all of the directory entries have been read, a null-terminated empty string is returned as file name. The position indicator is advanced to the next directory entry.

### Diagnostics

Return Code Value	ERRNO	Error Condition
NO_ERROR		Successful completion.
INVALID_PARAM	EBADF	DIRECTORY_ID is not an open directory identifier for the current partition
NOT_AVAILABLE	EBUSY	DIRECTORY_ID has an operation in progress
NOT_AVAILABLE	EIO	The storage device containing DIRECTORY_ID reports a failure
INVALID_MODE	EACCES	Preemption is disabled or the current process is the error handler
NO_ACTION	ENAMETOOLONG	The directory entry's length exceeds maximum directory entry length
INVALID_PARAM		A NULL pointer is passed for ERRNO (Development mode only).
INVALID_PARAM	EINVAL	A NULL pointer is passed for ENTRY_NAME or ENTRY_KIND (Development mode only).

INVALID_MODE	EACCES	No ARINC 653 partition associated with current thread (Development mode only).
INVALID_PARAM	EBADF	Invalid directory file descriptor (Development mode only).
NOT_AVAILABLE		Failed to change working directory (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-2 standard with the following exception:

- The service never returns the NOT\_AVAILABLE / ESTALE error.

Rationale for this deviation:

In LynxOS-178 ARINC FS operations implemented upon POSIX FS interface of Lynx file system. If file descriptor is no longer valid it will be detected by underlying POSIX calls and appropriate error will be set.

## REWIND\_DIRECTORY (P2:3.2.5.16)

### Synopsis

```
void REWIND_DIRECTORY (
/*in      */ DIRECTORY_ID_TYPE    DIRECTORY_ID,
/*out     */ RETURN_CODE_TYPE     *RETURN_CODE,
/*inout   */ FILE_ERRNO_TYPE     *ERRNO)
```

### Description

The REWIND\_DIRECTORY service positions the directory pointer to its first entry for subsequent reads. REWIND\_DIRECTORY will cause the directory identifier to refer to the current state of the corresponding directory, as if a call to OPEN\_DIRECTORY () was performed.

### Diagnostics

Return Code Value	ERRNO	Error Condition
NO_ERROR		Successful completion.
INVALID_PARAM	EBADF	DIRECTORY_ID is not an open directory identifier for the current partition
NOT_AVAILABLE	EBUSY	DIRECTORY_ID has an operation in progress

NOT_AVAILABLE	ESTALE	DIRECTORY_ID is no longer valid due to the directory owner's action
NOT_AVAILABLE	EIO	The storage device containing DIRECTORY_ID reports a failure
INVALID_MODE	EACCES	Preemption is disabled or the current process is the error handler
INVALID_PARAM		A NULL pointer is passed for ERRNO (Development mode only).
INVALID_MODE	EACCES	No ARINC 653 partition associated with current thread (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-2 standard with the following exceptions:

- The service never returns the NOT\_AVAILABLE / ESTALE error.

Rationale for this deviation:

In LynxOS-178 ARINC FS operations implemented upon POSIX FS interface of Lynx file system. If file descriptor is no longer valid it will be detected by underlying POSIX calls and appropriate error will be set.

- The service never returns the INVALID\_PARAM / EIO error.

Rationale for this deviation:

Rewinding directory file does not imply any disk I/O operations on the underlying file system.

## MAKE\_DIRECTORY (P2:3.2.5.17)

### Synopsis

```
void MAKE_DIRECTORY (
/*in   */ FILE_NAME_TYPE    DIRECTORY_NAME,
/*out  */ RETURN_CODE_TYPE  *RETURN_CODE,
/*inout*/ FILE_ERRNO_TYPE   *ERRNO)
```

### Description

The MAKE\_DIRECTORY service request creates the directory specified by its full path name on the volume. A partition must have write permissions for the volume in which the directory is to be created. Directories created with this service cannot already exist.

## Diagnostics

Return Code Value	ERRNO	Error Condition
NO_ERROR		Successful completion.
INVALID_PARAM	ENAMETOOLONG	DIRECTORY_NAME or one of its components exceed maximum character length
INVALID_PARAM	EINVAL	DIRECTORY_NAME is syntactically invalid
INVALID_PARAM	ENOTDIR	A component of the path prefix of DIRECTORY_NAME is not a volume or directory
INVALID_CONFIG	EACCES	The current partition does not have read-write access rights for the volume that would contain DIRECTORY_NAME
INVALID_PARAM	EROFS	The storage device containing DIRECTORY_NAME is currently write protected
INVALID_PARAM	EISDIR	DIRECTORY_NAME is an existing directory
INVALID_PARAM	EEXIST	DIRECTORY_NAME is an existing file
INVALID_CONFIG	ENOSPC	There is not enough space available on volume to make a directory
NOT_AVAILABLE	EIO	The storage device containing DIRECTORY_NAME reports a failure

## Conformance

This service is implemented according to the ARINC 653-2 standard.

## REMOVE\_DIRECTORY (P2:3.2.5.18)

## Synopsis

```
void REMOVE_DIRECTORY (
/*in      */ FILE_NAME_TYPE      DIRECTORY_NAME,
/*out     */ RETURN_CODE_TYPE    *RETURN_CODE,
/*inout   */ FILE_ERRNO_TYPE    *ERRNO)
```

## Description

The **REMOVE\_DIRECTORY** service request removes a directory. A partition must have write permissions for the requested parent directory in order to remove the directory. The directory must be empty of files and sub-directories.

#### Diagnostics

<b>Return Code Value</b>	<b>ERRNO</b>	<b>Error Condition</b>
NO_ERROR		Successful completion.
INVALID_PARAM	ENAMETOOLONG	DIRECTORY_NAME or one of its components exceed maximum character length
INVALID_PARAM	EINVAL	DIRECTORY_NAME is syntactically invalid
INVALID_PARAM	ENOTDIR	A component of the path prefix of DIRECTORY_NAME is not a volume or directory
INVALID_PARAM	EPERM	DIRECTORY_NAME is an existing file
INVALID_PARAM	ENOENT	DIRECTORY_NAME is not an existing directory
INVALID_CONFIG	EACCES	The current partition does not have read-write access rights for the volume that would contain DIRECTORY_NAME
INVALID_PARAM	EROFS	The storage device containing DIRECTORY_NAME is currently write protected
INVALID_PARAM	ENOTEMPTY	DIRECTORY_NAME is not empty of files and directories
NOT_AVAILABLE	EIO	The storage device containing DIRECTORY_NAME reports a failure
INVALID_MODE	EACCES	Preemption is disabled or the current process is the error handler
NOT_AVAILABLE	EBUSY	DIRECTORY_NAME is open by the owning partition
INVALID_PARAM		A NULL pointer is passed for ERRNO (Development mode only).
INVALID_MODE	EACCES	No ARINC 653 partition associated with current thread (Development mode only).

## Conformance

This service is implemented according to the ARINC 653-2 standard with the following exception:

- The service never returns the `INVALID_PARAM` / `EACCES` error when removing root directory.

Rationale for this deviation:

LynxOS behaves like BSD here: `ERRNO` will be set to `EBUSY`.

## SYNC\_DIRECTORY (P2:3.2.5.19)

### Synopsis

```
void SYNC_DIRECTORY(  
    /* in */ DIRECTORY_ID_TYPE  DIRECTORY_ID,  
    /* out */ RETURN_CODE_TYPE  *RETURN_CODE,  
    /* inout */ FILE_ERRNO_TYPE *ERRNO)
```

### Description

The `SYNC_DIRECTORY` service request causes data associated with the specified directory to be transferred to the associated storage device. The service does not return until the transfer completes or an error is reported. This provides a means to potentially prevent data from being lost due to a power-loss.

### Diagnostics

Return Code Value	ERRNO	Error Condition
<code>NO_ERROR</code>		Successful completion.
<code>INVALID_PARAM</code>	<code>EBADF</code>	<code>DIRECTORY_ID</code> is not an open directory identifier for the current partition
<code>NOT_AVAILABLE</code>	<code>EBUSY</code>	<code>DIRECTORY_ID</code> has an operation in progress
<code>NOT_AVAILABLE</code>	<code>EIO</code>	The storage device containing <code>DIRECTORY_ID</code> reports a failure
<code>INVALID_MODE</code>	<code>EACCES</code>	Preemption is disabled or the current process is the error handler.
<code>INVALID_PARAM</code>		A NULL pointer is passed for <code>ERRNO</code> (Development mode only).

INVALID_MODE	EACCES	No ARINC 653 partition associated with current thread (Development mode only).
INVALID_PARAM	EBADF	Invalid directory file descriptor (Development mode only).

### Conformance

This service is implemented according to the ARINC 653-2 standard with the following exceptions:

- The service never returns the `INVALID_PARAM` / `EACCES` error when trying to sync the directory which is located on a volume partition does not have the access rights.

### Rationale for this deviation:

Underlying LynxOS file system does not support volume access rights.

## RENAME\_DIRECTORY (P2:3.2.5.20)

### Synopsis

```
void RENAME_DIRECTORY (
    /*in */ FILE_NAME_TYPE      OLD_DIRECTORY_NAME,
    /*in */ FILE_NAME_TYPE      NEW_DIRECTORY_NAME,
    /*out */ RETURN_CODE_TYPE    *RETURN_CODE,
    /*inout*/ FILE_ERRNO_TYPE    *ERRNO)
```

### Description

The `RENAME_DIRECTORY` service request renames the existing directory with the specified new directory name. A directory can be given a new directory name that resides on the same volume (cannot rename to a different volume). The directory's contents are unaffected by the rename operation. Once renamed, the `OLD_DIRECTORY_NAME` name is no longer valid to open the directory.

### Diagnostics

Return Code Value	ERRNO	Error Condition
NO_ERROR		Successful completion.
INVALID_PARAM	ENAMETOOLONG	OLD_DIRECTORY_NAME or one of its components exceed maximum character length. NEW_DIRECTORY_NAME or one of its components exceed maximum character length.

INVALID_PARAM	EINVAL	OLD_DIRECTORY_NAME is syntactically invalid
INVALID_PARAM	ENOTDIR	A component of the path prefix of OLD_DIRECTORY_NAME is not a volume or directory
INVALID_PARAM	EINVAL	NEW_DIRECTORY_NAME is syntactically invalid
INVALID_PARAM	ENOTDIR	A component of the path prefix of NEW_DIRECTORY_NAME is not a volume or directory
INVALID_CONFIG	EACCES	The current partition does not have read-write access rights to the volume containing
NOT_AVAILABLE	EBUSY	OLD_DIRECTORY_NAME has an operation in progress
INVALID_PARAM	EROFS	The storage device containing OLD_DIRECTORY_NAME is currently write protected
INVALID_PARAM	EINVAL	The volume for OLD_DIRECTORY_NAME is not identical to the volume for NEW_DIRECTORY_NAME
INVALID_CONFIG	ENOSPC	Not able to rename the directory (not enough resources or max number of directories exceeded)
INVALID_PARAM	EPERM	OLD_DIRECTORY_NAME is an existing file
INVALID_PARAM	EISDIR	NEW_DIRECTORY_NAME is an existing file
INVALID_PARAM	ENOENT	OLD_DIRECTORY_NAME is not an existing directory
INVALID_PARAM	EEXIST	NEW_DIRECTORY_NAME is an existing directory and it is not empty
NOT_AVAILABLE	EBUSY	OLD_DIRECTORY_NAME is open for any access mode by the owning partition
INVALID_PARAM	EINVAL	OLD_DIRECTORY_NAME contains only the name of the volume



INVALID_PARAM	EINVAL	NEW_DIRECTORY_NAME is a subdirectory of OLD_DIRECTORY_NAME
NOT_AVAILABLE	EIO	The storage device containing OLD_DIRECTORY_NAME or NEW_DIRECTORY_NAME reports a failure
INVALID_MODE	EACCES	Preemption is disabled or the current process is the error handler

### Conformance

This service is implemented according to the ARINC 653-2 standard.

