Lw and ULw POSIX AEPs for Resource Constrained Processors

Document WINNF-14-S-0009

Version V1.0.1
25 July 2014

TERMS, CONDITIONS & NOTICES

This document has been prepared by the work group of WInnF project SCA-2013-003 “AEPs Improvements for SCA 4.1” to assist The Software Defined Radio Forum Inc. (or its successors or assigns, hereafter “the Forum”). It may be amended or withdrawn at a later time and it is not binding on any member of the Forum or of the AEPs Improvements for SCA 4.1 work group.

Contributors to this document that have submitted copyrighted materials (the Submission) to the Forum for use in this document retain copyright ownership of their original work, while at the same time granting the Forum a non-exclusive, irrevocable, worldwide, perpetual, royalty-free license under the Submitter’s copyrights in the Submission to reproduce, distribute, publish, display, perform, and create derivative works of the Submission based on that original work for the purpose of developing this document under the Forum's own copyright.

Permission is granted to the Forum’s participants to copy any portion of this document for legitimate purposes of the Forum. Copying for monetary gain or for other non-Forum related purposes is prohibited.

THIS DOCUMENT IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NON-INFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY USE OF THIS SPECIFICATION SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE FORUM, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER, DIRECTLY OR INDIRECTLY, ARISING FROM THE USE OF THIS DOCUMENT.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the specification set forth in this document, and to provide supporting documentation.

This document was developed following the Forum's policy on restricted or controlled information (Policy 009) to ensure that that the document can be shared openly with other member organizations around the world. Additional Information on this policy can be found here: http://www.wirelessinnovation.org/page/Policies_and_Procedures

Although this document contains no restricted or controlled information, the specific implementation of concepts contain herein may be controlled under the laws of the country of origin for that implementation. Readers are encouraged, therefore, to consult with a cognizant authority prior to any further development.

Wireless Innovation Forum ™ and SDR Forum ™ are trademarks of the Software Defined Radio Forum Inc.

Table of Contents

List of Figures

List of Tables

Executive Summary

This specification addresses specification of standard software interfaces applicable between SDR Applications and RTOS available on resource constrained processors such as DSPs.

It has been elaborated by the Wireless Innovation Forum as a contribution to SCA 4.1 elaboration, and aims at providing standard solution for the broadest range of actors of the SDR eco-system.

It defines base AEPs groups of POSIX functions for "Lightweight" / "Lw" and "Ultra-Lightweight" / "ULw" AEPs, in complement to the POSIX AEP defined by SCA 2.2.2.

Both have the same features for threads, semaphores, mutexes, timers and message queues. The Lw profile additionally features condition variables. The ULw brings minor restrictions concerning message queues (limited size and no support of priorities).

By contrast to the SCA AEPs, the AEPs defined in this specification do not address support of C language standard libraries.

In addition to the base AEPs groups, the specification defines two additional groups of POSIX functions, which enable to complement the base AEPs groups with: (i) run-time detection of errors, (ii) release of RTOS resources for SDR Application termination.

As support content, an exhaustive overview of the specification content is provided, by positioning content in front of the entire set of POSIX functions, as done in the SCA 2.2.2 Appendix B tables.

Besides, exhaustive rationale regarding the design choices is provided, on a function-by-function basis.

This specification is a solution that took into account and harmonized two publicly available solutions: the SCA 4.0.1 Lw AEP and the ESSOR Architecture DSP AEP.

It is expected that this specification will be broadly used within the international SDR eco-system structured through usage of SDR standards supporting SDR Applications portability.

Contributors

The direct contributors to this specification were:

- Aeroflex, with Dave Hagood as main contributor.
- Harris Corporation, with Dave Carlson as main contributor,
- Hitachi Kokusai Electric, with Dave Murotake as main contributor,
- Raytheon, with Jimmie Marks as main contributor,
- THALES Communications & Security, with Eric Nicollet as main contributor.

Direct contributions to the specification have been provided by contributors working with JTNC Standards, who provided methodological guidance in order to facilitate later exploitation of the specification in the elaboration process of SCA 4.1.

Lw and ULw POSIX AEPs for Resource Constrained Processors

1 Introduction

1.1 Aim of the specification

1.1.1 Positioning

This specification addresses definition of standard software interfaces between SDR (Software Defined Radio) Applications and RTOS (Real-Time Operating System) capabilities of SDR OEs (Operating Environments), addressing resource constrained processors.

The said software interfaces are defined as POSIX™-compliant (Portable Operating System Interfaces) Application Environment Profiles (AEPs).

This specification defines groups of POSIX functions that can be used by specification users for definition of "Lightweight" / "Lw" and "Ultra-Lightweight" / "ULw" AEPs applicable for their work products, in complement to the POSIX AEP defined by SCA 2.2.2.

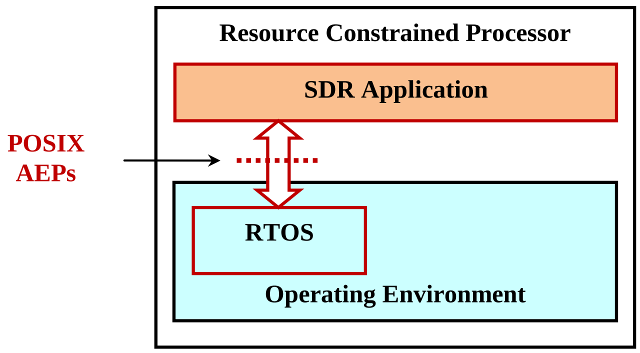

Figure 1 Positioning of the specification

By resource constrained processors are meant processors where a single address space is available (implying absence of process concept) and where no file system is implemented.

The SDR Applications running on such environments are considered to be hard-real time applications, implying their execution is constrained by strict execution deadlines.

While DSPs (Digital Signal Processors) are typical examples of what is meant by resource constrained processors, the set of potentially interested processors is broader than only DSPs, while this specification is not relevant for DSPs where not RTOS is available.

The specification supports emergence of portable SDR Applications and open, standard compliant, Operating Environments in resource constrained processors.

The addressed AEPs address a range of processors that were not covered by the SCA 2.2.2 AEP, which was primarily established for GPPs (General Purpose Processors).

By contrast to the AEP defined in SCA 2.2.2, only the relationship between SDR Application and RTOS capabilities of OEs are addressed. Other aspects addressed by POSIX are SCA 2.2.2 AEP, in particular the support of standard C language libraries, is not part of this specification.

1.1.2 Elaboration context and initial objectives

The AEPs addressed in this specification are primarily aimed at supporting elaboration of SCA 4.1. Besides, they are subject to be used by any interested user within the SDR eco-system.

The reference POSIX specification taken as basis for elaboration of the specification is POSIX 2013, as publicly available through the Open Group online distribution (see **[Ref1]**).

The initial elaboration approach was to improve and harmonize into a reference specification the achievements of two pre-existing publicly available AEPs: the Lw AEP defined in Appendix B of SCA 4.0.1 (see **[Ref2]**) and the DSP AEP of the ESSOR Architecture (see **[Ref3]**).

1.1.3 OEs implementation possibilities

The terminology “Operating Environment RTOS” is NOT necessarily referring to a COTS RTOS available for RTOS vendors, and is to be understood in a broader sense, only corresponding to the set of specified POSIX capabilities.

The implementation strategies of the said “OE RTOS” are then entirely under platform providers’ responsibility. Some examples of possibilities offered are:

- A COTS RTOS natively supporting the AEPs, integrated “as is” within the complete OE,
- A software implementation of the expected AEPs capability realized on top of a COTS micro-kernel or a proprietary RTOS, and being integrated within the complete OE.

The specification content is therefore written in abstraction of the OE implementation choices, without presupposing that the underlying solution shall be a POSIX-compliant RTOS.

1.2 Specification content

1.2.1 Lw and ULw Base AEPs functions groups

Base AEPs functions groups provide the essential capabilities for RTOS usage.

The two *Base AEPs functions groups* are defined as:

- The “Base Lightweight / Lw AEP functions group”,
- The “Base Ultra-lightweight / ULw AEP functions group”.

The two Base AEPs functions groups commonly feature *Threads*, *Semaphores*, *Message queues*, *Mutexes* and *Timers* capabilities.

The differences between the Base AEPs functions groups are:

- The Lw features *Condition variables* capabilities, while the ULw does not,
- The ULw does not support *Message queues* priorities and restricts the messages size to sizeof(void *).

The Base ULw AEP functions group is therefore as a strict subset of the Base Lw AEP functions group.

The existence of the Base Lw and ULw AEPs functions groups is justified by grounded distinct use cases for conforming to one AEP or the other.

The Base Lw AEP functions group is designed as the natural choice for resource constrained processors featuring lightweight POSIX-certified RTOS.

The Base ULw AEP functions group is designed as a suitable solution for processors with ultra-constrained resources, where the Base Lw AEP functions group features may consume too many resource. Such environments are prone to usage of COTS DSP RTOSes or micro-kernels, completed by the OE integrator so as to expose the Base ULw AEP functions group.

Normative content for Base AEP is in § 2, with associated rationale in § 6.2.

1.2.2 Additional groups

Additional groups of POSIX functions enable users of the specification to extend the selected Base AEP functions group, if and as required.

Group A

Group A essentially contains the POSIX functions enabling SDR Applications to get values of RTOS-related attributes (“get” functions). Their usage can typically ease the porting phase in enabling errors detection in the SDR Application code.

Normative content for Group A is in § 3, with associated rationale in § 6.3.

Group B

Group B contains the POSIX functions enabling SDR Application to release RTOS resources. Their usage is required for termination of SDR Applications ensuring that no unused RTOS resource remains allocated once the application is inactive.

Normative content for Group B is provided in § 4, with associated rationale in § 6.4.

1.3 Conformance

1.3.1 Applicable AEPs

Users of the specification are using the specification in determining the *applicable AEP* for their conformant work products.

First, one Base AEPs functions group has to be selected among the **Base Lw AEP functions group** and the **Base ULw AEP functions group.** This choice determines if the *applicable AEP* is a Lw or a ULw AEP.

Then, as needed, the selected Base AEPs functions group can be augmented with **Group A** and/or **Group B** functions, thus completing definition of the *applicable AEP*. This choice is not impacting the fundamental design choices of the work products, and is rather influenced by porting assumptions.

1.3.2 Conformance of SDR Applications

The *applicable AEP* for a conformant SDR Application has to be defined in accordance with § 1.3.1.

Conformity criteria

A SDR Application is *conformant* with one *applicable AEP* if the said SDR Application **exclusively uses**, as far as RTOS capabilities are concerned, the capabilities of the said *applicable AEP*.

Verification procedure

Establishment of the *verification procedure* applicable for verification of the conformity criteria is beyond the scope of this specification.

It implies to determine the exact meaning of the "*exclusively uses*" part of the conformity clause, that can go from simple syntactical verification of a source code to exhaustive verification of dependencies, including flagging of forbidden POSIX functions using a black list of functions.

Significant margins of interpretation exist as well concerning the decision to test or not that the SDR Application is making some "forced" calls, such as the attributes setting calls listed by the specification for threads or mutexes creation.

1.3.3 Conformance of Operating Environments

Conformity criteria

The *applicable AEP* for a conformant OE has to be defined in accordance with § 1.3.1.

An Operating Environment is *conformant* with one *applicable AEP* if the said OE **at least implements the full set** of capabilities specified in the said *applicable AEP*. An Operating Environment is *compliant* with one *applicable AEP* if the said OE **implements a non-trivial subset** of capabilities of the said *applicable AEP*, the implemented functions being individually complying with the specification.

Verification procedure

Establishment of the *verification procedure* applicable for verification of those conformity criteria is beyond the scope of this specification.

It implies to determine if a specific testing procedure is required to declare OE conformance, or if a usage-based validation approach is sufficient.

Besides, a verification procedure could imply to require a formal POSIX conformity certificate, duly established in application of POSIX conformity testing rules.

POSIX conformance

As far as the range of capabilities addressed by this specification is concerned, no difference has been identified by specification developers between the POSIX 2013 **[Ref1]** version of POSIX, which serves as main reference for this document, and POSIX 2008.1, which is a version of POSIX largely implemented by COTS RTOS vendors. This implies that RTOS complying with POSIX 2008 are assumed conformant with the content of this specification.

1.3.4 Benefits of conformance

For an Operating Environment, conformance to one *applicable AEP* will ensure capability for the said OE to host SDR Applications conformant with the same *applicable AEP*, or with narrower AEPs.

For SDR Applications, conformance to one *applicable AEP* will ensure possibility to port the said SDR Application to OEs conformant with the same *applicable AEP*, or with broader AEPs.

2 Base Lw and ULw AEPs

2.1 Introduction

This chapter provides the normative content for definition of the Base AEPs functions groups (ULw ad Lw).

It is structured across 6 POSIX capabilities:

- Threads,
- Semaphores,
- Mutexes,
- Message queues (with ULw-specific restrictions),
- Timers,
- Condition variables (Lw only).

2.2 Threads

2.2.1 Overview of Base AEPs Threads

The following table gives the set of POSIX functions of the Base Lw and ULw AEPs for *Threads* capability.

Table 1 POSIX functions of the Base AEPs for Threads

POSIX functions of the Base AEPs	**Summary role**	**Normative content in**
pthread_create()	Thread creation	Threads lifecycle
pthread_self()	Thread ID retrieving	Threads lifecycle
pthread_attr_init()	Thread attr creation	Threads attr lifecycle
pthread_attr_destroy()	Thread attr destruction	Threads attr lifecycle
pthread_attr_setdetachstate()	Setting *detachstate*	Attribute *detachstate*
pthread_attr_setinheritsched()	Setting *inheritsched*	Attribute *inheritsched*
pthread_attr_setschedparam()	Setting *schedparam*	Attribute *schedparam*
pthread_attr_setschedpolicy()	Setting *schedpolicy*	Attribute *schedpolicy*
pthread_attr_setstack()	Setting stack size and address	Stack handling
pthread_attr_setstacksize()	Setting stack size	Stack handling

2.2.2 Normative content for Base AEPs Threads

2.2.2.1 Threads lifecycle

POSIX functions: *pthread_create*() and *pthread_self*().

POSIX standard is fully applicable.

The created threads only support the specific values indicated in following chapters for attributes *detachstate*, *inheritsched* and *schedpolicy*.

The related destructor *pthread_exit*() is member of Group B of functions, see § 4.2.

2.2.2.2 Threads attributes lifecycle

POSIX functions: *pthread_attr_init*() and *pthread_attr_destroy*().

POSIX standard is fully applicable.

2.2.2.3 Attribute *detachstate*

POSIX functions: *pthread_attr_setdetachstate*().

The *detachstate* attribute of threads is always equal to PHTHREAD_CREATE_DETACHED. Other values, including POSIX default PTHREAD_CREATE_JOINABLE, are not part of the required capabilities.

POSIX standard is otherwise fully applicable.

The POSIX function *pthread_attr_setdetachstate()* is in the Base AEPs, to explicitly set attribute *detachstate* to PTHREAD_CREATE_DETACHED.

The related accessor *pthread_attr_getdetachstate()* is member of Group A of functions, see § 3.2.

Implications on conformant products:

- A conformant SDR Application has to explicitly call *pthread_attr_setdetachstate(* PTHREAD_CREATED_DETACHED*)*,
- A conformant OE has to implement *pthread_attr_setdetachstate()*, even if only supporting PTHREAD_CREATE_DETACHED value.

2.2.2.4 Attribute *inheritsched*

POSIX functions: *pthread_attr_setinheritsched*().

The *inheritsched* attribute of threads is always equal to PTHREAD_EXPLICIT_SCHED. Other values are not part of the required capabilities.

POSIX standard is otherwise fully applicable.

The POSIX function *pthread_attr_setinheritsched*() is in the Base AEPs, to explicitly set attribute *inheritsched* to PTHREAD_EXPLICIT_SCHED.

The related accessor *pthread_attr_getinheritsched*() is member of Group A of functions, see § 3.2.

Implications on conformant products:

- A conformant SDR Application has to explicitly call *pthread_attr_setinheritsched(* PTHREAD_EXPLICIT_SCHED*)*,
- A conformant OE has to implement *pthread_attr_setinheritsched()*, even if only supporting PTHREAD_EXPLICIT_SCHED value.

2.2.2.5 Attribute *schedparam*

POSIX functions: *pthread_attr_setschedparam*().

POSIX standard is fully applicable.

The related accessor *pthread_attr_gettschedparam*() is member of Group A of functions, see § 3.2.

2.2.2.6 Attribute *schedpolicy*

POSIX functions: *pthread_attr_setschedpolicy*().

The *schedpolicy* attribute of threads is always equal to SCHED_FIFO. Other values are not part of the required capabilities.

POSIX standard is otherwise fully applicable.

The POSIX function *pthread_attr_setschedpolicy*() is in the Base AEPs, to explicitly set attribute *schedpolicy* to SCHED_FIFO.

The related accessor *pthread_attr_gettschedpolicy*() is member of Group A of functions, see § 3.2.

Implications on conformant products:

- A conformant SDR Application has to explicitly call *pthread_attr_setschedpolicy(* SCHED_FIFO*)*,
- A conformant OE has to implement *pthread_attr_setschedpolicy()*, even if only supporting SCHED_FIFO.

2.2.2.7 Stack handling

POSIX functions: *pthread_attr_setsstack*() and *pthread_attr_setsstacksize*().

POSIX standard is fully applicable.

The related accessors *pthread_attr_getstacksize*() and *pthread_attr_getstackaddr*() are members of Group A of functions, see § 3.2.

2.3 Semaphores

2.3.1 Overview of Base AEPs Semaphores

The following table gives the set of POSIX functions of the Base Lw and ULw AEPs for *Semaphores* capability.

Table 2 POSIX functions of the Base AEPs for Semaphores

POSIX function	Summary role	Normative content in
sem_init()	Semaphore creation	Semaphores lifecycle
sem_post()	Posting	Semaphores usage
sem_wait()	Waiting	Semaphores usage

2.3.2 Normative content for Base AEPs Semaphores

2.3.2.1 Semaphores lifecycle

POSIX functions: *sem_init*().

The related destructor *sem_destroy*() is member of Group B of functions, § 4.2.

POSIX standard is fully applicable.

2.3.2.2 Semaphores usage

POSIX functions: *sem_post*() and *sem_wait*().

POSIX standard is fully applicable.

The accessor to instant value of semaphore *sem_getvalue*() is the member function of the Group A of functions, see § 3.2.

2.4 Mutexes

The following table gives the set of POSIX functions of the Base Lw and ULw AEPs for *Mutexes* capability.

Table 3 POSIX functions of the Base AEPs for Mutexes

POSIX function	Summary role	Normative content in
pthread_mutex_init()	Mutex creation	Mutexes lifecycle
pthread_mutexattr_init()	Mutex attr creation	Mutexes attr lifecycle
pthread_mutexattr_destroy()	Mutex attr destruction	Mutexes attr lifecycle
pthread_mutexattr_setprioceiling()	Setting *prioceiling*	Attribute *prioceiling*
pthread_mutexattr_setprotocol()	Setting *protocol*	Attribute *protocol*
pthread_mutexattr_settype()	Setting *type*	Attribute *type*
pthread_mutex_lock()	Mutex locking	Mutexes usage
pthread_mutex_unlock()	Mutex unlocking	Mutexes usage

2.4.1 Mutexes lifecycle

POSIX functions: *pthread_mutex_init*().

POSIX standard is fully applicable.

The related destructor *pthread_mutex_destroy()* is member of Group B of functions, § 4.2.

2.4.2 Mutexes attributes lifecycle

POSIX functions: *pthread_mutexattr_init*() and *pthread_mutexattr_destroy*().

POSIX standard is fully applicable.

2.4.3 Attribute protocol

POSIX functions: *pthread_mutexattr_setprotocol*().

POSIX standard is fully applicable.

Only the following POSIX-defined values are supported for attribute *protocol*: PTHREAD_PRIO_INHERIT and PTHREAD_PRIO_PROTECT.

The related accessor *pthread_mutexattr_getprotocol*() is member of the Group A of functions, see § 3.2.

Implications on conformant products:

- A conformant OE has to implement *pthread_mutexattr_setprotocol*(), with at least one *protocol* value supported among PTHREAD_PRIO_INHERIT and PTHREAD_PRIO_PROTECT.
- A conformant SDR Application has to use one of the available values for attribute *protocol*.
- A modification in the SDR Application can be required to adapt the SDR Application if migrating from one platform supporting one protocol to another only supporting protocol.

2.4.4 Attribute prioceiling

POSIX functions: *pthread_mutexattr_setprioceiling*().

Those functions are only necessary in case PTHREAD_PRIO_PROTECT is the applicable protocol.

POSIX standard is fully applicable.

The related accessor *pthread_mutexattr_getprioceiling*() is member of the Group A of functions, see § 3.2.

Implications on conformant products:

- A conformant OE has to implement *pthread_attr_setprioceiling*(), if supporting PTHREAD_PRIO_PROTECT,
- A conformant SDR Application has to call *pthread_attr_setprioceiling*(), if using PTHREAD_PRIO_PROTECT.

2.4.5 Attribute type

POSIX functions: *pthread_mutexattr_settype*().

The *type* attribute of mutexes is always equal to PTHREAD_MUTEX_NORMAL, implying usage of non-robust mutexes is sufficient.

POSIX standard is otherwise fully applicable.

The POSIX function *pthread_mutexattr_settype*() is in the Base AEPs, to explicitly set attribute *type* to PTHREAD_MUTEX_NORMAL.

The related accessor *pthread_mutexattr_gettype*() is member of the Group A of functions, see § 3.2.

Implications on conformant products:

- A conformant SDR Application has to explicitly call *pthread_mutexattr_settype(* PTHREAD_MUTEX_NORMAL*),*
- A conformant OE has to implement *pthread_mutexattr_settype()*, even if only supporting PTHREAD_MUTEX_NORMAL.

2.4.6 Mutexes usage

POSIX functions: *pthread_mutex_lock*() and *pthread_mutex_unlock*().

POSIX standard is fully applicable.

2.5 Message queues

2.5.1 Overview of Basre AEPs Message queues

The following table gives the set of POSIX functions of the Base Lw and ULw AEPs for *Message queues* capability.

Table 4 POSIX Functions of the Base AEPs for Message queues

POSIX function	Summary role	Normative content in
mq_open()	Message queue creation	Message queues lifecycle
mq_receive()	Message queue locking	Message queues usage
mq_send()	Message queue unlocking	Message queues usage

2.5.2 Normative content for Base AEPs Message queues

2.5.2.1 Message queues lifecycle

POSIX functions: *mq_open*().

POSIX standard is fully applicable.

The related destructor *mq_close*() is member of Group B of functions, § 4.2.

2.5.2.2 Message queues usage

POSIX functions: *mq_receive*() and *mq_send*().

POSIX standard is fully applicable.

The flag O_NON_BLOCKING is never set, implying nominal blocking behavior is always used by message queues.

IMPORTANT: the following restrictions specifically apply to Base ULw AEP group of functions:

- No support of priorities,
- mq_msgsize = sizeof(void *).

The Base Lw AEP group of functions is NOT affected by those restrictions.

2.6 Timers

The following table gives the set of POSIX functions of the Base Lw and ULw AEPs for *Timers* capability.

Table 5 POSIX Functions of the Base AEPs for Timers

POSIX function	Summary role	Normative content in
timer_create()	Timer creation	Timers lifecycle
clock_getres()	Timer resolution	Clock resolution
timer_settime()	Timer activation	Timers usage
clock_gettime()	Time retrieving	Clock usage

2.6.1 Timers lifecyle

POSIX functions: *timer_create*().

For timers, particular dispositions apply to arguments *clockid* and *sigevent* of function *timer_create*().

For *clockid*, no other POSIX-defined values than CLOCK_REALTIME can be used.

In case the resolution available with CLOCK_REALTIME is not satisfying the need of the SDR Application, usage of implementation-defined *clockid* values for higher resolution clocks is permitted.

For *sigevent* argument, the *sigev_notify* member of the *sigevent* structure has to be equal to SIGEV_THREAD or to one implementation-defined value.

In case SIGEV_THREAD is implemented, the thread of execution of the callback has to be a system thread created at timer creation, otherwise an implementation-defined mechanism is to be used.

The *sigevent* structure *sigev_notify* member values SIGEV_NONE and SIGEV_SIGNAL are not part of the required capabilities.Otherwise, POSIX standard is fully applicable.

The related destructor *timer_delete*() is member of Group B of functions, § 4.2.

Implications on conformant products:

- A conformant OE has to support value CLOCK_REALTIME for argument *clockid*, plus implementation-defined value for higher resolution if required,
- A conformant OE has to support, for member *sigev_notify* of argument *sigevent*, at least one value among SIGEV_THREAD or implementation-defined value(s),
- A conformant SDR Application has to use one of the values supported by the OE (if several),
- A modification in the SDR Application will be required when adapting it to support an implementation-defined value for *sigev_notify*.

2.6.2 Clock resolution

POSIX functions: *clock_getres*().

POSIX standard is fully applicable.

Only CLOCK_REALTIME and implementation-defined value for higher resolution, if required, are supported for argument *clockid*.

2.6.3 Timers usage

POSIX functions: *timer_settime*().

POSIX standard is fully applicable.

The dispositions described in § 2.6.1 are applicable to activation of callback at timer expiration.

The related accessor *timer_gettime*() is the member of Group A of functions, see § 3.2.

2.6.4 Clock usage

POSIX functions: *clock_gettime*().

POSIX standard is fully applicable.

Only CLOCK_REALTIME and implementation-defined value for higher resolution, if required, are supported for argument *clockid*.

2.7 Condition variables

The following table gives the set of POSIX functions of the Base Lw AEP group of functions for *Condition variables* capability.

IMPORTANT: *Condition variables* are not supported in the Base ULw AEP group of functions.

Table 6 POSIX Functions of the Base Lw AEP for Condition Variables

POSIX function	Summary role	Normative content in
pthread_cond_init()	Start of lifecycle	Condition variables lifecycle
pthread_cond_broadcast()	Wake-up all waiters	Condition variables usage
pthread_cond_signal()	Wake-up one waiter	Condition variables usage
pthread_cond_wait()	Wait for signal/broadcast	Condition variables usage

2.7.1 Condition variables lifecycle

POSIX functions: *pthread_cond_init*().

POSIX standard is fully applicable.

The SDR Applications have to pass a NULL *pthread_cond_attr_t* pointer when calling *pthread_cond_init().*

The related destructor *pthread_cond_destroy()* is member of Group B of functions, § 4.2.

2.7.2 Condition variables attributes lifecycle

POSIX functions: *pthread_condattr_init*() and *pthread_condattr_destroy*().

POSIX standard is fully applicable.

2.7.3 Condition variables usage

POSIX functions: *pthread_cond_broadcast*(), *pthread_cond_signal*() and *pthread_cond_wait*().

POSIX standard is fully applicable.

3 Group A of functions

3.1 Introduction

Group A essentially contains the POSIX functions enabling SDR Applications to get values of RTOS-related attributes ("get" functions). Their usage can typically ease the porting phase in enabling errors detection in the SDR Application code.

3.2 Group A Functions

The functions of Group A are listed in the following table.

Table 7 POSIX Group A Functions

POSIX function	Summary role	Normative content in
pthread_attr_getguardsize()	Retrieving *guardsize*	Threads creation attributes
pthread_attr_getdetachstate()	Retrieving *detachstate*	Threads creation attributes
pthread_attr_getinheritsched()	Retrieving *inheritsched*	Threads creation attributes
pthread_attr_getschedparam()	Retrieving *schedparam*	Threads creation attributes
pthread_attr_getschedpolicy()	Retrieving *schedpolicy*	Threads creation attributes
pthread_attr_getstackaddr()	Retrieving *stackaddr*	Threads creation attributes
pthread_attr_getstacksize()	Retrieving *stacksize*	Threads creation attributes
pthread_mutexattr_getprioceiling()	Retrieving *prioceiling*	Mutexes creation attributes
pthread_mutexattr_getprotocol()	Retrieving *protocol*	Mutexes creation attributes
pthread_mutexattr_gettype()	Retrieving *type*	Mutexes creation attributes
pthread_attr_setguardsize()	Setting *guardsize*	Guard size handling
sem_getvalue()	Retrieving instant semaphore value	Semaphores monitoring
timer_gettime()	Retrieving instant timers time	Timers monitoring

To facilitate porting on an OE where Group A would not be available, a SDR Application conformant with Group A functions has to limit usage of Group A functions to run-time errordetection.

3.2.1 Threads creation attributes

POSIX functions: *pthread_attr_getguardsize*(), *pthread_attr_getdetachstate*(), *pthread_attr_getinheritsched*(), *thread_attr_getschedparam*(), *pthread_attr_getschedpolicy*(), *pthread_attr_getschedstackaddr*() and *pthread_attr_getstaksize*().

POSIX standard is fully applicable.

3.2.2 Mutexes creation attributes

POSIX functions: *pthread_mutexattr_getprioceiling*(), *pthread_mutexattr_getprotocol*() and *pthread_mutexattr_gettype().*

POSIX standard is fully applicable.

3.2.3 Guard size setting

POSIX function: *pthread_attr_setguardsize*() and *pthread_attr_getguardsize*().

POSIX standard is fully applicable.

For OE where *guardsize* setting is not possible, the function *pthread_attr_setguardsize*() has to be implemented with a void behavior.

3.2.4 Semaphores monitoring

POSIX function: *sem_getvalue*().

POSIX standard is fully applicable.

3.2.5 Timers monitoring

POSIX function: *timer_gettime*().

POSIX standard is fully applicable.

4 Group B of functions

4.1 Introduction

Group B contains the POSIX functions enabling SDR Application to release RTOS resources. Their usage is required for termination of SDR Applications ensuring that no unused RTOS resource remains allocated once the application is inactive.

4.2 Group B Functions

The functions of Group B are listed in the following table.

Table 8 POSIX Group B Functions

POSIX function	Summary role	Normative content in
pthread_exit()	Terminate a thread	Threads resources releasing
sem_destroy()	Destroy a semaphore	Semaphore releasing
mq_close()	Close a message queue	Message queue releasing
pthread_mutex_destroy()	Destroy a mutex	Mutexes resources releasing
timer_delete()	Delete a timer	Timer releasing
pthread_cond_destroy()	Destroy a condition variable (Base Lw only)	Condition variable releasing

To facilitate porting on OE where Group B would not be available, a SDR Application conformant with Group B functions has to limit usage of Group B functions to the termination phase.

4.2.1 Threads resources releasing

POSIX function: *pthread_exit*().

POSIX standard is fully applicable.

4.2.2 Semaphore releasing

POSIX function: *sem_destroy*().

POSIX standard is fully applicable.

4.2.3 Message queue releasing

POSIX function: *mq_close*().

POSIX standard is fully applicable.

4.2.4 Mutexes resources releasing

POSIX function: *pthread_mutex_destroy*().

POSIX standard is fully applicable.

4.2.5 Timer releasing

POSIX function: *timer_delete*().

POSIX standard is fully applicable.

4.2.6 Condition variable releasing

POSIX function: *pthread_cond_destroy*().

POSIX standard is fully applicable.

This function is member of Group B only when Base Lw AEP group is used, since only Lw features condition variables.

5 Perspectives on specification content

5.1 Possible applicable AEPs

The rules for elaboration of applicable AEPs result in the following 8 different possibilities for applicable AEPs:

- 4 variants of Lw AEPs:
    - **Base Lw AEP** functions group **only**,
    - **Base Lw AEP** functions group with **Group A**,
    - **Base Lw AEP** functions group with **Group B**,
    - **Base Lw AEP** functions group with **Group A** and **Group B**.
- 4 variants of ULw AEPs:
    - **Base ULw AEP** functions group **only**,
    - **Base ULw AEP** functions group with **Group A**,
    - **Base ULw AEP** functions group with **Group B**,
    - **Base ULw AEP** functions group with **Group A** and **Group B**.

The essential choice represents selection of the Base functions group, that will determine if the applicable AEP is Lw or ULw.

The variants allowed by presence of additional groups enable to broaden usage of the specification to a variety of porting situations.

Definition of more prescriptive requirements concerning usage of the specification is possible, for instance in limiting, for certain markets or users perspective, the number of applicable AEPs possibly used.

5.2 AEPs content

This section is based on tables that provide for each of the Base AEPs functions groups (Lw and ULw), a status regarding presence of each individual function of the functions group.

As such, the tables are close to those available in SCA Appendix B (see **[Ref2]**).

The status is provided according to the following convention:

- IN (MAN): the function is IN the profile, with no identified restriction compared to the POSIX standard
    - This is equivalent to the "mandatory = MAN" status used in SCA tables,
- OUT (NRQ): the function is OUT of the profile,
    - This is equivalent to the "not requested = NRQ" status used in SCA tables,
- PRT (PRT): the function is in the profile, with a partial set of capabilities with regards to full POSIX functionality,
    - This is somehow equivalent to the "partial = PRT" status used in SCA tables.

By contrast with SCA Appendix E, the only tables provided are corresponding to the function groups where at least one function is present IN the Base Lw and Base ULw functions groups:

- POSIX_MQUEUES: covers the *message queues* capability,

- POSIX_SEMAPHORES: covers the *semaphores* capability,
- POSIX_TIMERS: covers the *timers* capability,
- POSIX_THREADS_BASE: covers the *threads*, *mutexes* and *condition variables* capabilities.

5.2.1 POSIX_MQUEUE functions

Table 9 Specified functions groups wrt POSIX_MQUEUE

Function	SCA AEP	Lw AEPs	ULw AEPs	Diff Lw/ULw
mq_close()	MAN	**Group B**	**Group B**	
mq_getattr()	MAN	OUT (NRQ)	OUT (NRQ)	
mq_notify()	MAN	OUT (NRQ	OUT (NRQ	
mq_open()	MAN	**IN (MAN)**	**PRT (PRT)***	**YES**
mq_receive()	MAN	**IN (MAN)**	**IN (MAN)**	
mq_send()	MAN	**IN (MAN)**	**IN (MAN)**	
mq_setattr()	MAN	OUT (NRQ)	OUT (NRQ)	
mq_timedreceive()	NRQ	OUT (NRQ)	OUT (NRQ)	
mq_timedsend()	NRQ	OUT (NRQ)	OUT (NRQ)	
mq_unlink()	MAN	OUT (NRQ)	OUT (NRQ)	

*: the partial (PRT) indication for function *mq_open*() of the ULw AEP denotes absence of priorities support and the size limited to sizeof(void *).

This is one point of difference between the Lw and the ULw AEPs.

5.2.2 POSIX_SEMAPHORES functions

Table 10 Specification content wrt POSIX_SEMAPHORES

Function	SCA AEP	Lw AEPs	ULw AEPs	Diff Lw/Ulw
sem_close()	MAN	OUT (NRQ)	OUT (NRQ)	
sem_destroy()	MAN	**Group B**	**Group B**	
sem_getvalue()	MAN	**Group A**	**Group A**	
sem_init()	MAN	**IN (MAN)**	**IN (MAN)**	
sem_open()	MAN	OUT (NRQ)	OUT (NRQ)	
sem_post()	MAN	**IN (MAN)**	**IN (MAN)**	
sem_timedwait()	MAN	OUT (NRQ)	OUT (NRQ)	
sem_trywait()	MAN	OUT (NRQ)	OUT (NRQ)	
sem_unlink()	MAN	OUT (NRQ)	OUT (NRQ)	
sem_wait()	MAN	**IN (MAN)**	**IN (MAN)**	

5.2.3 POSIX_TIMERS functions

Table 11 Specification content wrt POSIX_TIMERS

Function	SCA AEP	Lw AEPs	ULw AEPs	Diff Lw/Ulw
clock_getres()	MAN	**IN (MAN)**	**IN (MAN)**	
clock_gettime()	MAN	**IN (MAN)**	**IN (MAN)**	
clock_settime()	MAN	OUT (NRQ)	OUT (NRQ)	
nanosleep()	MAN	OUT (NRQ)	OUT (NRQ)	
timer_create()	MAN	**PRT (PRT)***	**PRT (PRT)***	
timer_delete()	MAN	**Group B**	**Group B**	
timer_getoverrun()	MAN	OUT (NRQ)	OUT (NRQ)	
timer_gettime()	MAN	**Group A**	**Group A**	
timer_settime()	MAN	**IN (MAN)**	**IN (MAN)**	

*: the partial (PRT) indication for function *timer_create*() denote the limitations set in timers expiration conditions, with either possibility to use SIGEV_THREAD or an implementation specific mechanism.

5.2.4 POSIX_THREADS functions

Table 12 Specification content wrt POSIX_THREADS_BASE (except mutexes and cond vars)

Function	SCA AEP	Lw AEPs	ULw AEPs	Diff Lw/Ulw
pthread_atfork()	NRQ	OUT (NRQ)	OUT (NRQ)	
pthread_attr_init()	MAN	**IN (MAN)**	**IN (MAN)**	
pthread_attr_destroy()	MAN	**IN (MAN)**	**IN (MAN)**	
pthread_attr_getdetachstate()	MAN	**Group A**	**Group A**	
pthread_attr_getguardsize()	MAN	**Group A**	**Group A**	
pthread_attr_getinheritsched()	MAN	**Group A**	**Group A**	
pthread_attr_getschedparam()	MAN	**Group A**	**Group A**	
pthread_attr_getschedpolicy()	MAN	**Group A**	**Group A**	
pthread_attr_getscope()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_attr_getstack()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_attr_getstackaddr()	MAN	**Group A**	**Group A**	
pthread_attr_getstacksize()	MAN	**Group A**	**Group A**	
pthread_attr_setdetachstate()	MAN	**PRT (PRT)***	**PRT (PRT)***	
pthread_attr_setguardsize()	MAN	**Group A**	**Group A**	
pthread_attr_setinheritsched()	MAN	**PRT (PRT)***	**PRT (PRT)***	
pthread_attr_setschedparam()	MAN	**IN (MAN)**	**IN (MAN)**	
pthread_attr_setschedpolicy()	MAN	**PRT (PRT)***	**PRT (PRT)***	
pthread_attr_setscope()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_attr_setstack()	MAN	**IN (MAN)**	**IN (MAN)**	
pthread_attr_setstackaddr()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_attr_setstacksize()	MAN	**IN (MAN)**	**IN (MAN)**	
pthread_cancel()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_cleanup_xxx()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_create()	MAN	**IN (MAN)**	**IN (MAN)**	
pthread_detach()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_equal()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_exit()	MAN	**Group B**	**Group B**	
pthread_getschedparam()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_getspecific()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_join()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_key_xxx()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_kill()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_once()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_self()	MAN	**IN (MAN)**	**IN (MAN)**	
pthread_setcancelstate()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_setcanceltype()	MAN	OUT (NRQ)	OUT (NRQ)	

pthread_setschedparam()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_setspecific()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_sigmask()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_testcancel()	MAN	OUT (NRQ)	OUT (NRQ)	

*: the partial (PRT) indications for functions *pthread_attr_setdetachstate*(), *pthread_attr_setinheritsched*() and *pthread_attr_setschedpolicy*() correspond to the limitations set on values of attributes *detachstate* (always equal to PTHREAD_CREATE_DETACH), *inheritsched* (always equal to PTHREAD_EXPLICIT_SCHED) and *schedpolicy* (always equal to SCHED_FIFO).

Table 13 Specification content wrt POSIX_THREADS_BASE (mutexes)

Function	SCA AEP	Lw AEPs	ULw AEPs	Diff Lw/Ulw
pthread_mutex_destroy()	MAN	**Group B**	**Group B**	
pthread_mutex_getprioceiling()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_mutex_init()	MAN	**IN (MAN)**	**IN (MAN)**	
pthread_mutex_lock()	MAN	**IN (MAN)**	**IN (MAN)**	
pthread_mutex_setprioceiling()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_mutex_timedlock()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_mutex_trylock()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_mutex_unlock()	MAN	**IN (MAN)**	**IN (MAN)**	
pthread_mutexattr_destroy()	MAN	**IN (MAN)**	**IN (MAN)**	
pthread_mutexattr_getprioceiling()	MAN	**Group A***	**Group A***	
pthread_mutexattr_getprotocol()	MAN	**Group A****	**Group A****	
pthread_mutexattr_getpshared()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_mutexattr_gettype()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_mutexattr_init()	MAN	**IN (MAN)**	**IN (MAN)**	
pthread_mutexattr_setprioceiling()	MAN	**IN (MAN)***	**IN (MAN)***	
pthread_mutexattr_setprotocol()	MAN	**IN (MAN)***	**IN (MAN)***	
pthread_mutexattr_setpshared()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_mutexattr_settype()	MAN	**PRT (PRT)*****	**PRT (PRT)*****	

*: only required if THREAD_PRIO_PROTECT is supported.

**: only support of one among PTHREAD_PRIO_INHERIT and THREAD_PRIO_PROTECT values for attribute *protocol* is required for the OE. On a given platform, the SDR Application needs to use one of the available protocols.

***: the partial (PRT) indication for function *pthread_mutexattr_settype*() corresponds to the limitation set on values of attribute *type* (always equal to PTHREAD_MUTEX_NORMAL).

Table 14 Specification content wrt POSIX_THREADS_BASE (cond vars)

Function	SCA AEP	Lw AEPs	ULw AEPs	Diff Lw/Ulw
pthread_cond_broadcast()	MAN	**IN (MAN)**	OUT (NRQ)	**YES**
pthread_cond_destroy()	MAN	**Group B**	OUT (NRQ)	**YES**
pthread_cond_init()	MAN	**IN (MAN)***	OUT (NRQ)	**YES**
pthread_cond_signal()	MAN	**IN (MAN)**	OUT (NRQ)	**YES**
pthread_cond_timedwait()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_cond_wait()	MAN	**IN (MAN)**	OUT (NRQ)	**YES**
pthread_condattr_destroy()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_condattr_init()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_condattr_getclock()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_condattr_setclock()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_condattr_getpshared()	MAN	OUT (NRQ)	OUT (NRQ)	
pthread_condattr_setpshared()	MAN	OUT (NRQ)	OUT (NRQ)	

*: creation is to be done by SDR Applications passing a NULL *pthread_condattr_t* pointer.

Presence of POSIX Condition variables functions in the Base Lw AEP group and not in the Base ULw AEP group is the main difference between Base Lw and ULw groups.

6 Rationales

6.1 Design paradigm

6.1.1 Selection of main POSIX capabilities

This resulted in consensus regarding support of *Threads*, *Semaphores*, *Mutexes*, *Timers* and *Message queues*.

Support of *Condition variables* appeared to be a point of divergence that stemmed identification of distinct Lw and ULw Base AEP functions groups.

It was as well clarified that the scope of the specification would only address RTOS interfaces, considering as out of scope aspects related to support of standard C language libraries.

6.1.2 Function-by-function evaluation

The second step of convergence has been, for each of the selected functionalities, to determine which where the POSIX functions to be kept in.

Core functions, for *creation* and *usage* of the POSIX capability were included.

A number of limitations in the supported POSIX attributes were identified for some capabilities. This resulted in integration of some "set" functions with associated limitations in attributes value.

Consideration of *Message queues* brought additional differences between the Lw and ULw Base AEPs functions groups.

The case of "get" functions appeared to be hard to decide about, with grounded contradictory argumentations relative to their inclusion. The case of "destroy" functions appeared similarly hard to decide about.

6.1.3 Creation of the Additional groups

By contrast with the Lw vs ULw differences, usage or not of the "get" (=Group A) and "destroy" (=Group B) functions are of minor impact for SDR Application portability, and are mostly influenced by porting and integrations assumptions strongly depending on the usage context.

This justified to include those functions as additional groups possibly added to the Base AEPs functions groups, based on users decision related to porting objectives and constraints.

6.2 Design rationale for Base AEPs functions groups

6.2.1 Design rationale for Base AEPs Threads

Lifecycle management

POSIX functions for *threads* creation is IN the Base AEPs functions groups: *pthread_create*().

The related POSIX function for *threads* release, *pthread_exit*(), is in Group B.

POSIX functions for *threads attributes* creation and destruction are in the Base AEPs functions groups: *pthread_attr_init*() and *pthread_attr_destroy*().

pthread_attr_destroy() is in the Base AEPs functions group and not in Group B because the attributes destruction is typically done at the end of the SDR Application initialization, releasing the threads creation attributes from memory once threads have been created.

Threads cancelation is NOT allowed, so *pthread_cancel*() is NOT in the Base AEPs functions groups. Attributes *cancelstate* and *canceltype* are therefore undefined, with related POSIX functions *pthread_[s|g]etcancelstate*() and *pthread_[s|g]etcanceltype*() OUT of the Base AEPs functions groups.

Usage

No related POSIX functions for threads, since handled by the scheduler.

Attributes

The threads are created in a unique address space, with no processes. Attribute *contentionscope* is therefore undefined, with related POSIX functions *pthread_attr_[s|g]etscope*() OUT of the Base AEPs functions groups.

The function *pthread_attr_setschedparam*() is IN the Base AEPs functions groups, with no specific limitation. The related accessor *pthread_attr_getschedparam*() is OUT of the Base AEPs functions groups, but is a member of Group A.

The threads have imposed values for the following attributes:

- Attribute *detachstate*: PTHREAD_CREATED_DETACHED,
- Attribute *inheritsched*: PTHREAD_EXPLICIT_SCHED,
- Attribute *schedpolicy*: SCHED_FIFO.

To enable portability of SDR Application on OEs with RTOS supporting more creation attributes values that the limited set defined by the Base AEPs functions groups, the related attributes handling POSIX functions are IN the Base AEPs functions groups: *pthread_attr_setdetachstate*(), *pthread_attr_setinheritsched*(), *pthread_attr_setschedpolicy*().

The related accessors *pthread_attr_getdetachstate*(), *pthread_attr_getinheritsched*(), *pthread_attr_getschedpolicy*() are OUT of the Base AEPs functions groups, but are members of Group A.

Stack size handling functions are IN the Base AEPs functions groups: *pthread_attr_setstacksize*().

Stack base address handling functions are IN the profile, with *pthread_attr_setstack*(), enabling simultaneous setting of stack address and size. The function *pthread_attr_setstackaddr*() is OUT of the Base AEPs functions groups, since deprecated since POSIX 2008.1.

The accessors related to stack allocation *pthread_attr_getstacksize*() and *pthread_attr_getstackaddr*() are OUT of the Base AEPs functions groups, but are members of Group A.

Stack overflow detection functions *pthread_attr_[s|g]etguardsize*() are OUT of the Base AEPs functions groups, but are members of Group A.

The functions for dynamic handling of *schedparam* and *schedprio* attributes, *pthread_[s|g]etschedparam*() and *pthread_[s|g]etschedprio*() are OUT of the Base AEPs functions groups, since no dynamic handling of those parameters is not supported.

6.2.2 Design rationale for Base AEPs Semaphores

Lifecycle management

POSIX functions for *semaphores* creation is IN the Base AEPs functions groups: *sem_init*().

The related POSIX function for *semaphores* release, *sem_destroy*(), is in Group B.

Since resource constrained processors have a single address space no support for named semaphores is required, therefore functions *sem_open*(), *sem_close*() and *sem_unlink*()are OUT of the Base AEPs functions groups.

Usage

The base POSIX functions for *semaphores* usage are IN the Base AEPs functions groups: *sem_post* () and *sem_wait*().

Related accessor *sem_getvalue*() is OUT of the Base AEPs functions groups, but is member of Group A, see § 3.2.

Since the SDR applications considered for execution in resource constrained processors are designed according to hard real-time constraints, the functions *sem_timedwait*() and *sem_trywait*() are OUT of the Base AEPs functions groups.

Attributes

No attributes structure is defined for semaphores.

6.2.3 Rationale for design choices of Base AEPs Message queues

Lifecycle management

POSIX function for *message queues* creation is IN the Base AEPs functions groups: *mq_open*().

The related POSIX function for *message queues* release, *mq_close*(), is in Group B.

Usage

The base POSIX functions for *message queues* usage are IN the Base AEPs functions groups: *mq_send*() and *mq_receive*().

The absence of support for messages priority of the Base ULw AEP functions group is motivated by lack of support of this feature in COTS micro-kernels, coupled to secondary added value in front of SDR Applications for resource constrained processors.

The Base Lw AEP functions groups supports messages priority since largely availability in COTS POSIX RTOS.

The limitation of messages size to sizeof(void *) in the Base ULw AEP functions group is justified by assignment of dynamic memory allocations so SDR Application, enabling optimization of memory allocation according to available memory types.

Since the SDR applications considered for execution in resource constrained processors are designed according to hard real-time constraints, the functions *mq_timedreceive*() and *mq_timedsend*() are OUT of the Base AEPs functions groups.

Since resource constrained processors have a single address space, functions *mq_notify*() and *mq_unlink*() are OUT of the Base AEPs functions groups.

Attributes

No dynamic modification of the message queue flag is supported. Therefore, the functions *mq_getattr*() and *mq_setattr*() are OUT of the Base AEPs functions groups.

6.2.4 Rationale for design choices of Base AEPs Mutexes

Lifecycle management

POSIX function for *mutexes* creation is IN the Base AEPs functions groups: *pthread_mutex_init()*.

The related POSIX function for *mutexes* release, *pthread_mutex_destroy*(), is in Group B.

POSIX functions for *mutexes attributes* creation and destruction are IN the Base AEPs functions groups: *pthread_mutexattr_init*() and *pthread_mutexattr_destroy*().

pthread_mutexattr_destroy() is in the Base AEPs functions group and not in Group B because the mutexes attributes destruction is typically done at the end of the SDR Application initialization, releasing the mutexes creation attributes from memory once mutexes have been created.

Usage

The base POSIX functions for *mutexes* usage are IN the Base AEPs functions groups: *pthread_mutex_lock*() and *pthread_mutex_unlock*().

Since the SDR applications considered for execution in resource constrained processors are designed according to hard real-time constraints, the function *pthread_mutex_trylock*() and *pthread_mutex_timedlock*() are OUT of the Base AEPs functions groups.

Attributes

Since resource constrained processors have a single address space, functions *pthread_mutexattr_[s|g]etpshared*() are OUT of the Base AEPs functions group.

For attribute *protocol*, some RTOS are only supporting value PTHREAD_PRIO_PROTECT while others only support PTHREAD_PRIO_INHERIT. Therefore the AEPs are including the two possibilities, with obligation for the OE to implement at least one. The function *pthread_mutexattr_setprotocol*() is therefore IN the Base AEPs functions groups.

Related accessor *pthread_mutexattr_getprotocol*() is OUT of the Base AEPs functions groups, but is member of Group A, see § 3.2.

In case *protocol* PTHREAD_PRIO_PROTECT is used, functions *pthread_mutexattr_setprioceiling*() is required. It is therefore IN the Base AEPs functions groups, in case PTHREAD_PRIO_PROTECT is used.

Related accessor *pthread_mutexattr_getprioceiling*() is OUT of the Base AEPs functions groups, but is member of Group A, see § 3.2.

The attribute *type* is always having value NORMAL. The functions *pthread_mutexattr_settype*() is therefore IN the Base AEPs functions groups to force this value. This implies that only non-robust mutexes are required.

Related accessor *pthread_mutexattr_gettype*() is OUT of the Base AEPs functions groups, but is member of Group A, see § 3.2.

6.2.5 Rationale for design choices of AEPs Timers

Lifecycle management

POSIX function for *timers* creation is IN the Base AEPs functions groups: *timer_create*().

The related POSIX function for *timers* release, *timer_delete*(), is in Group B.

The possibility to use the SIGEV_THREAD value for *sigevent_notify* is allowed under the assumption that a system thread is used for callback execution. This precision is provided since POSIX implementations diverge in interpretation of the clause “as if executing in a newly created threads”. The recommendation followed in the Base AEPs functions groups is motivated by evident real-time efficiency considerations, since systematic creation of a dedicated thread for handling of each expiring timer is not real-time efficient.

SIGEV_SIGNAL is not permitted since signals are not supported in Base AEPs functions groups conformant RTOS. SIGEV_NONE is not permitted since a notification mechanism is evidently required in hard real-time SDR Applications.

Therefore, the possibility to use other mechanism is offered for implementations where the SIGEV_THREAD interpretation would not allow for efficient timers expiration handling.

Usage

The base POSIX function for *timers* usage are IN the Base AEPs functions groups: *timer_settime*().

The POSIX function for SDR Application to retrieve time is IN the Base AEPs functions groups: *clock_gettime*(). This is allowed to enable usage of absolute time for timers expiration setting.

The associated accessor *timer_gettime*() is OUT of the Base AEPs functions groups, but is a member of Group A.

The base POSIX function for setting the clock value is OUT of the Base AEPs functions groups, since the SDR Application has no privilege to set clock time.

Attributes

The POSIX function for SDR Application to retrieve clock resolution is IN the Base AEPs functions group: *clock_getres*().

6.2.6 Rationale for design choices of AEPs Condition variables

Most RTOS have at least one native mechanism that allows one or more threads of control to wait for an "event" or "condition" and to be released (dispatched) from the waiting state when the event is "signaled" by some other thread of control.

The exact semantics of the native event wait-dispatch mechanisms vary from vendor to vendor as is the case for other native mechanism (eg., native mutex versus the POSIX pthread_mutex_t.). The POSIX mapping of the event wait-dispatch idiom is the POSIX Condition variable (pthread_cond_t). The POSIX Condition variable semantics were introduced in the original POSIX.1c (pthreads).

There is existing practice in SDR Applications (namely Waveforms) for the POSIX CondVar, justifying insertion of condition variables in Base Lw AEP functions group. The semantics of the POSIX CondVar are conducive to writing scalable, race-free multi-waiter/multi-dispatcher codes that are common in many multi-threaded real-time problems.

Conversely, there is existing practice of SDR Applications (namely Waveforms) the no condition variables or equivalent event dispatch mechanisms are used from the RTOS, justifying exclusion of condition variables in the Base ULw AEP functions group.

Lifecycle management

POSIX function for *condition variables* creation is IN the Base Lw AEP functions group: *pthread_cond_init*().

The related POSIX function for *condition variables* release, *pthread_cond_destroy*(), is in Group B.

POSIX functions for *condition variables attributes* lifecycle management are OUT the Base Lw AEP functions group: *pthread_condattr_init*() and *pthread_condattr_destroy*() since, as explained below in section Attributes, no specific attribute setting related to shared memory or time is supported by the Base Lw AEP functions group condition variables.

Usage

The base POSIX functions for *condition variables* usage are IN the Base Lw AEPfunctions group: *pthread_cond_signal*(), *pthread_cond_broadcast*() and *pthread_cond_wait*().

Since the SDR applications considered for execution in resource constrained processors are designed according to hard real-time constraints, the function *pthread_cond_timedwait*() is OUT of the Base Lw AEP functions group.

Attributes

The Base Lw AEP functions group provides no support for *pthread_cond_timedwait*(). Consistently, the *pthread_condattr_[g|s]etclock*() are not supported.

The Base Lw AEP function group provides no support for POSIX shared memory, therefore the associated functions *pthread_condattr_[g|s]setpshared*() are not supported.

As a consequence, *condattr* is not settable for condition variables.

6.3 Rationale for Group A

6.3.1 Rationale for Group A existence

The existence of Group A is justified by distinct cases for conforming or NOT conforming with Group A, as exposed hereinafter.

Case for conforming to Group A

Usage of Group A functions is nominal situation unless the processing resources constraints become predominant.

A SDR Application conformant with Group A will take advantage of the POSIX functions of Group A, typically for enabling errors detection based on the value of the “get” retrieved thanks to Group A functions. As stated in the normative content, the usage has to be limited to run-time errors detection.

An OE conformant with Group A will offer the set of POSIX functions of Group A. This is nominally provided by COTS POSIX-certified RTOS.

Case for NOT conforming to Group A

The case is grounded by situations where minimization of resources consumption for conformant products is considered more important than usage of values retrieved from the "get" functions.

A SDR Application NOT conformant with Group A only executes with the base AEPs functions. It is not benefitting of usage of the Group A functions, implying likely increase in porting efforts. This enables to avoid any overhead created by run-time errors detection.

An OE NOT conformant with Group A only implements the base AEPs functions. It is not enabling SDR Application to benefit from "get" values, but this saves memory footprint in the concerned OE.

This approach is particularly applicable in resource constrained OE where AEP-compliant RTOS capability is based on specific developments on top of scalable micro-kernels, eventually open source and recompiled by the OE provider.

6.3.2 Rationale for design choices of Group A

Only the POSIX "get" functions corresponding to "set" functions of the AEPs are IN Group A: *pthread_attr_getdetachstate*(), *pthread_attr_getinheritsched*(), *thread_attr_getschedparam*(), *pthread_attr_getschedpolicy*(), *pthread_attr_getschedstackaddr*() and *pthread_attr_getstaksize*().

Only the POSIX "get" functions corresponding to "set" functions of the AEPs are IN Group A: *pthread_mutexattr_getprioceiling*(), *pthread_mutexattr_getprotocol*() and *pthread_mutexattr_gettype*().

The case for having the previous "get" functions is primarily grounded on the possibility it offers for integrator to retrieve the default creation values of the RTOS when attributes are initialized.

The POSIX function *sem_getvalue*() is IN Group A, since getting the instant run-time value taken by a semaphore is enabling writing of error detection assertions related to semaphores status.

The POSIX function *timer_gettime*() is IN Group A, since getting the instant value of a timer count can be used is enabling writing of error detection assertions related to timers execution.

The POSIX function *pthread_attr_setguardsize*() is IN Group A, since enabling detection of stack memory leakage.

The POSIX "get" function *pthread_attr_getguardsize*() is IN Group A since its "set" function is IN Group A as well, in order to enable checking of correct support of the requested guard size by the RTOS.

6.4 Rationale for Group B

6.4.1 Rationale for Group B existence

The existence of Group B is justified by distinct cases for conforming or NOT conforming with Group B, as exposed hereinafter.

Case for conforming to Group B

The case is grounded by desire to have ability to terminate SDR Applications without resetting the processor. This implies to cleanly release the RTOS resources used by the SDR Application.

For SDR Application, the application will be capable to be dynamically removed without maintaining usage of RTOS resources.

For OE, the OE will support dynamic removal of SDR Applications.

Case for NOT conforming to Group B

The case is grounded by designs where the reconfiguration across SDR Application simply takes place using reset of the resource constrained processor.

For SDR Application, the application does not need to encode proper releasing of the RTOS resources it uses.

For OE, the OE does not have to implement the resource-releasing functions of Group B.

6.4.2 Rationale for design choices for Group B functions

All destructors associated to Base AEPs functions groups have been included in Group B, except destructors of threads and mutexes creation attributes, which are kept in Base AEPs since not related to termination of SDR Applications.

7 Acronyms List

AEP	Application Environment Profile
COTS	Commercially Off-The-Shelf
DSP	Digital Signal Processor
ESSOR	European Secure Software Radio
GPP	General Purpose Processor
JTNC	Joint Tactical Networking Center
Lw	Lightweight
OE	Operating Environment
POSIX	Portable Operating System Interface
RTOS	Real-Time Operating System
SCA	Software Communications Architecture
SDR	Software Defined Radio
ULw	Ultra-lightweight
WInnF/WINNF	Wireless Innovation Forum

8 References

8.1 POSIX 2013

[Ref1] The Open Group Base Specifications Issue 7 – IEEE Std 1003.1™, 2013 Edition - © 2001-2013 The IEEE and the Open Group.
URL: http://pubs.opengroup.org/onlinepubs/9699919799/ (free online distribution, no file).

8.2 SCA 4.0.1 Appendix B

[Ref2] *Software Communications Architecture Specification, Appendix B: SCA Application Environment Profiles*, Joint Tactical Networking Center, Version 4.0.1, 01 October 2012.
URL: http://jtnc.mil/sca/Pages/sca1.aspx (.pdf file).

8.3 ESSOR Architecture DSP AEP

[Ref3] *Reference content from ESSOR on DSP AEPs*, input document to WInnF submitted by Indra Sistemas, WINNF-11-I-0007.
URL: http://groups.winnforum.org/d/do/4690 (.pdf file).

END OF THE DOCUMENT