Autotalks API Manual / 4.11.0

October 23, 2016

Contents

1 Disclaimer

Autotalks reserves the right to make changes to information published in this document, at any time and without notice.

Information is this document is believed to be accurate and reliable. However, Autotalks does not give any representations or warranties as to the accuracy of information and shall have no liability for the consequences of use of such information.

2 Introduction

Autotalks offers a set of APIs that enable software development on its V2X platform. The APIs can be used by programs written in the C and C++ programming languages.

2.1 V2X platform

Autotalks V2X platform is composed of CRATON V2X communication processor and PLUTON V2X RF transceiver.

Evaluation of this chipset can be done on PANGAEA4 V2X development platform which includes a GNSS receiver and a V2X Hardware Security Module (HSM) flashed with Autotalks dedicated firmware.

Paired with a Tablet running Autotalks Tablet application, PANGAEA4 can be used to perform field tests, sending and receiving Autotalks proprietary V2X Communications Analyzer (VCA) frames.

Post processing and statistical analysis of test results can be done with the desktop (Windows OS) application Autotalks V2X Tools.

2.2 SDK flavours

Autotalks SDK supports two SDK flavours – single-core (SC) and multi-core (MC). The multi-core SDK tarball contains the acronym `mc` in its name.

Both flavours contain the same API headers, although certain functions are implemented in one and not the other. Please refer to the module_support section for further details.

CRATON V2X communication processor contains three CPU cores:

1. ARM Cortex-R4F, referred to as `arm`.
2. ARC 625D, referred to as `arc1`.
3. ARC 625D, referred to as `arc2`.

IMQ API can be used for communications between SW components running on different CRATON cores. For more information on how to develop SW which runs in a multi-core environment please contact Autotalks support.

2.3 Compilation environments

When there's a need to maintain source code targeting both CRATON and other runtime environments, the following macros can help enable or disable features at compilation time:

1. `__CRATON__` and `__THREADX__` are defined in all CRATON compilation environments.
2. `__CRATON_ARM` is defined only in CRATON `arm` compilation environment.
3. `__CRATON_ARC1` is defined only in CRATON `arc1` compilation environment.
4. `__CRATON_ARC2` is defined only in CRATON `arc2` compilation environment.

Example:

```
#ifdef __THREADX__
// On ThreadX use POSIX Compliancy Wrapper API
#include <tx_posix.h>
#endif
#ifdef __linux__
// On Linux use pthread API
#include <pthread.h>
#endif
```

2.4 Module services

Most API modules use an abstract interface called a `service`. This interface can be implemented in more than one way depending among other things on the flavour of the SDK used and on whether the module is used locally or remotely.

The term `locally` refers to SW which is running directly on CRATON V2X communication processor; `remotely` refers to SW which is running on an external processor.

Each module contains functions to create and delete service instances, whose names end with `_create` and `_delete` respectively.

Each module also contains a getter for the `default` service. In typical usage scenarios users are **expected to get the `default` service**.

Users should always delete service instances when they are no longer needed. Please note that deleting a service is also required when the service was gotten (via the `default` service getter).

2.5 API headers

The SDK includes two types of API headers:

1. Generic APIs – headers located in the folder `include/atlk/` are generic in nature and are not tied down to a specific Autotalks chipset.
2. CRATON-specific APIs – headers located in the folder `include/craton/` are specific to Autotalks CRATON chipset.

The latter type can only be used for SW running locally on CRATON V2X communication processor. In some cases, CRATON API headers are also tied down to ThreadX RTOS or to the NetX-Duo TCP/IP network stack.

Some of CRATON's device driver APIs are included in `include/craton` but are not documented in this manual. SDK users are not expected to use these APIs at this stage. Please contact Autotalks support if you wish to do so.

Please note that regardless of a header's type, this document discusses it in the context of current Autotalks V2X chipset offering.

2.5.1 Optional APIs

The SDK includes two optional sub-systems. These are included as separate libraries - `libsntp` and `libvca`:

1. SNTP API - Based on NetX SNTP Client Component v5.9-beta.
2. VCA API - Enables initialization of Autotalks proprietary V2X Communications Analyzer daemon and its management sub-system.

2.5.2 Third party headers

Third party headers can be found under `include`:

1. ThreadX and POSIX Compliancy Wrapper for ThreadX.
2. NetX-Duo and BSD 4.3 Socket API Compatible Interface to NetX-Duo.
3. libcli.

Note that these headers might contain minor alterations done at Autotalks for the purpose of integration.

Usage of these APIs is outside the scope of this document. Please refer to the following:

1. ThreadX: `ThreadX_User_Guide.pdf`
2. NetX-Duo: `NetX_Duo_User_Guide.pdf`

For an example on how to use the POSIX Compliancy Wrapper on target, please refer to `examples/craton-threadx/posix/.`

For an example on how to use BSD 4.3 Socket API Compatible Interface to NetX-Duo on target, please refer to `examples/craton-threadx/net/nx-bsd-udp-receive-example.c`.

2.5.3 Unsupported headers

Additional headers can be found under `src/include/` accompanied by implementation source code.

These headers are used as a reference of how to implement libraries which interface with "remote" services supported by Autotalks SDK on an external CPU.

Please note that these headers are

not supported by Autotalks and are not considered a part of the SDK's API. Autotalks reserves the right to change these headers at any time without notice.

3 Glossary

- **AES:** Advanced Encryption Standard
- **AOA:** Angle Of Arrival
- **ASN.1**: Abstract Syntax Notation One
- **BSM:** Basic Safety Message (see SAE J2735 NOV2011 std)
- **CAN:** Controller Area Network
- **CBC-MAC**: Cipher Block Chaining Message Authentication Code
- **CCM:** CTR mode with CBC-MAC
- **CSD:** Cyclic Shift Delay
- **CTR:** Counter mode encyrption
- **DCOC:** DC Offset Cancellation
- **DR:** Dead Reckoning
- **ECC:** Elliptic Curve Cryptography
- **ECDH:** Elliptic Curve Diffie–Hellman
- **ECDSA:** Elliptic Curve Digital Signature Algorithm
- **ECIES:** Elliptic Curve Integrated Encryption Scheme
- **ECQV:** Elliptic Curve Qu-Vanstone algorithm
- **EPK:** Execution Profile Kit
- **EUI:** Extended Unique Identifier
- **EVK:** Evaluation Kit
- **GA:** Guaranteed Accuracy
- **GNSS:** Global Navigation Satellite System
- **GPS:** Global Positioning System
- **HSM:** Hardware Security Module
- **IF:** Interface
- **IMQ:** Inter-processor Message Queue
- **LLC:** Logical Link Control
- **MAC:** Media Access Control
- **MC:** Multi Core
- **MIB:** Management Information Base

- **MSDU:** MAC Service Data Unit
- **OFDM:** Orthogonal Frequency Division Multiplexing
- **PA:** Power Amplifier
- **PDU:** Protocol Data Unit
- **PHY:** Physical Layer
- **PKI:** Public-Key Infrastructure
- **PPS:** Pulse Per Second
- **RFIC:** Radio Frequency Integrated Circuit
- **RSSI:** Receiver Signal Strength Indication
- **RTOS:** Real-Time Operating System
- **RX:** Reception
- **SHA:** Secure Hash Algorithm
- **SNAP:** Subnetwork Access Protocol
- **SNMP:** Simple Network Management Protocol
- **TAI:** Temps Atomique International
- **TSF:** Timing Synchronization Function
- **TSSI:** Transmitter Signal Strength Indication
- **TX:** Transmission
- **UTC:** Coordinated Universal Time
- **V2X:** Vehicle-to-X (where X may be vehicle, infrastructure or portable device)
- **VCA:** V2X Communications Analyzer (proprietary Autotalks test frames)
- **WAVE:** Wireless Access in Vehicular Environment
- **WGS:** World Geodetic System
- **WLAN:** Wireless Local Area Network
- **WSMP:** WAVE Short Message Protocol (see IEEE 1609.x std)

4 API overview

4.1 API modules

This chapter gives an overview of API modules available to SDK users and a mapping of API headers to these modules.

It is important to note that some modules use both generic API headers and CRATON-specific API headers in certain usage scenarios.

The following headers are shared SDK headers and are not a part of any specific module:

- `atlk/sdk.h` - Shared declarations and macros.
- `atlk/os.h` - Autotalks OS abstraction definitions.
- `atlk/compiler.h` - Compiler specific attributes, declarations and macros.
- `atlk/verinfo.h` - SDK version information.
- `atlk/eui48.h` - 48-bit Extended Unique Identifier (i.e. a "MAC address").
- `craton/user.h` - CRATON user's init function and abort handler.

4.1.1 V2X

The V2X API enables transmission and reception of V2X frames above the Link Layer, locally or remotely to CRATON as well as emulating this functionality.

- `atlk/v2x.h` - Shared V2X API declarations.
- `atlk/v2x_service.h` - V2X service handling, TX and RX functions.
- `atlk/v2x_emulator.h` - Emulated V2X service handling, TX and RX functions.
- `craton/v2x_emulator_init.h` - Initialize emulated V2X service.

4.1.2 Management

The management API, referred to as `MIB API`, provides and interface to Autotalks device management, locally and remotely.

- `atlk/mib_service.h` - MIB service handling.

Generally, the API mirrors proprietary Autotalks MIBs and selected MIB attributes from standard MIBs and is divided into headers named accordingly.

Autotalks MIBs:

- `atlk/mibs/wlan-mib.h` - Management of the Wireless LAN sub-system, mirrors `AUTOTALKS-WLAN-MIB`.
- `atlk/mibs/nav-mib.h` - Management of the Navigation sub-system, mirrors `AUTOTALKS-NAV-MIB`.
- `atlk/mibs/vca-mib.h` - Management of the VCA sub-system, mirrors `AUTOTALKS-VCA-MIB`.
- `atlk/mibs/rsvc-mib.h` - Management of Remote Services, mirrors `AUTOTALKS-RSVC-MIB`.
- `atlk/mibs/tc.h` - Mirrors `AUTOTALKS-TC` (Autotalks textual conventions).

Standard MIBs:

- `atlk/mibs/if-mib.h` - Management of device interfaces, mirrors a subset of `IF-MIB` (see RFC-2863).
- `atlk/mibs/snmpv2-mib.h` - Management of SNMP, mirrors a subset of `SNMPv2-MIB` (see RFC-3418).
- `atlk/mibs/snmpv2-tc.h` - Mirrors a subset of `SNMPv2-TC` (see RFC-2579).

Please note that the VCA subsystem is provided for **demonstrational** purposes during development stages and comes with **limited support**.

4.1.3 Crypto

The cryptographic API is a set of APIs exposing CRATON's HW cryptographic capabilities.

ECC API:

- `atlk/ecc.h` - Shared ECC API declarations.
- `atlk/ecc_service.h` - ECC service handling, sign and verify functions.

HSM API:

- `atlk/hsm.h` - Shared HSM API declarations.
- `atlk/hsm_service.h` - HSM service handling, HSM functions.
- `atlk/hsm_emulator.h` - Enables usage of an emulated HSM device.
- `craton/slx97_host.h` - Infineon SLx97 HSM device.

SHA and RNG APIs:

- `atlk/sha.h` - Shared SHA API declarations.
- `craton/sha_hw.h` - Compute SHA using CRATON's HW SHA engine.
- `atlk/rng.h` - Get random bytes.

Other APIs: Additional cryptographic capabilities are exposed via the following headers:

- `atlk/aes.h` - AES-CCM support.
- `atlk/ecdsa.h` - ECDSA support.
- `atlk/ecies.h` - ECIES support.

4.1.4 CAN

The CAN bus API enables transmission and reception of CAN frames.

- `atlk/can.h` - Shared CAN API declarations.
- `atlk/can_service.h` - CAN service handling, TX and RX functions.

4.1.5 Navigation:

The Navigation API provides an interface to navigation data sources.

- `atlk/nav.h` - Shared navigation API declarations.
- `atlk/nav_service.h` - Navigation service handling, RX of navigation data.

Note that since the GNSS receiver is external to CRATON, the availability of navigation services is dependent of the platform used.

4.1.6 Network

The CRATON Network API enables direct access to NetX-Duo instances running above CRATON's Ethernet interface and its two WLAN interfaces. Using these instances is done via NetX-Duo's API.

- `craton/net.h` - Access the `trusted` (IPv4 above Ethernet) and `untrusted` (IPv6 above WLAN) NetX-Duo instances.

4.1.7 IMQ

IMQ is the main means of communication between SW components running in different cores when using a MC-SDK version.

- `craton/imq.h` - IMQ socket handling, TX and RX of IMQ messages.
- `craton/imq_user.h` - Available IMQ addresses for user usage.

4.1.8 Debug and logging

Debug and logging facilities include a RFC-5424 compatible system logger and a CLI based on the open source libcli.

Users have the ability to define their own exception handlers and monitor WLAN traffic from 802.11 header and up.

- `craton/syslog.h` - Log messages via CRATON's system logger.
- `craton/debug.h` - Print messages directly to console.
- `craton/cli.h` - Access CRATON's libcli CLI instance.
- `craton/bootparam.h` - Access variables in the boot environment.
- `craton/exception.h` - Defines exception info structs for ARM and ARC CPUs.
- `craton/wd.h` - Control HW watchdog on all CPUs.
- `craton/wlan_driver.h` - Register traffic monitor callback.

4.1.9 Optional modules

Optional modules are packages as separate libraries (all other modules are included in `libcraton`).

- `atlk/sntp_client.h` - SNTP client; requires linking with `libsntp`.
- `atlk/vca.h` - VCA initialization.
- `atlk/vcad.h` - VCA management initialization.

Note that VCA headers are not in their final form and might change or be discontinued in subsequent SDK releases.

4.1.10 Third party

- `tx_api.h` - ThreadX API.
- `tx_execution_profile.h` - ThreadX Execution Profile Kit API.
- `tx_posix.h` - POSIX Compliancy Wrapper for ThreadX API.
- `nx_api.h` - NetX-Duo API.
- `nxd_bsd.h` - BSD 4.3 Socket API Compatible Interface to NetX-Duo.
- `libcli.h` - libcli API.

Please note that libcli is provided for **convenience** to aid in development stages and comes with **limited support**.

4.2 General usage guidelines

Please make sure to initialize API structures in your program with the initializer macros provided for each struct type. This will prevent struct members that will be added in future API versions to be uninitialized, leading to unexpected side-effects. We recommend to enable your compiler's "uninitialized struct member" warning.

Example:

```
// Like this
foo_t foo = FOO_INIT;
foo.bar = 5;

// Not like this
foo_t foo = { .bar = 5 };

// And not like this
foo_t foo;
foo.bar = 5;
```

Please avoid using directly any struct member whose name is prefixed with __. Such struct members could be removed in future API versions.

4.2.1 Function call context

All functions in Autotalks APIs (anything under `include/atlk` and `inlcude/craton`) **should not be called from a non-thread context** such as timer or ISR. The only exception is debug_printf which can be called from any context.

Warning

Calling Autotalks API functions from a non-thread context may result in undefined behavior or lead to a kernel panic.

For allowed function call context of third party APIs please refer to third party documentation.

4.2.2 Using wait option

A wait option is used to control the "blocking" behavior of certain API functions.

Common to its usage is that as a function parameter (e.g. `wait` in v2x_send) it is an optional parameter and that not passing it (i.e. passing `NULL`) means the function call will be **non-blocking**.

For convenience, the predefined blocking wait option atlk_wait_forever has been defined.

5 Examples

API examples are included in the last section of this manual.

Two types of examples are included:

1. Examples which run directly on CRATON on top of the ThreadX RTOS.
2. Examples which run externally to CRATON on top of an OS which supports POSIX.

5.1 CRATON ThreadX examples

The starting point for these examples is the function craton_user_init, which is called at the end of the firmware's initialization sequence. In MC-SDK a craton_user_init function must be defined for each core.

Note that craton_user_init is called in the context of a thread with the highest priority (i.e. priority 0). Users are **not** expected to run code in a loop in this function – doing so will starve all other threads in the system.

Users are expected to run their code in their own thread (or threads) created with an appropriate priority. It is OK to create global objects in craton_user_init such as a V2X service.

Creating threads is done via ThreadX API and is outside the scope of this document.

CRATON ThreadX examples are those contained under `examples/craton-threadx/`.

An example on how to integrate C++ code can be found in `examples/craton-threadx/build/`. An example on how to use the POSIX Compliancy Wrapper on target can be found in `examples/craton-threadx/posix/`.

6 Data Structure Documentation

6.1 aes_cbc_iv_t Struct Reference

AES-CBC initialization vector.

`#include <atlk/aes.h>`

6.1.1 Detailed Description

AES-CBC initialization vector.

Examples:

craton-threadx/crypto/aes-example.c, craton-threadx/crypto/secure-storage-example.c, and remote-posix/crypto/aes-example.c.

The documentation for this struct was generated from the following file:

- atlk/aes.h

6.2 aes_ccm_authentication_tag_t Struct Reference

AES-CCM authentication tag.

`#include <atlk/aes.h>`

6.2.1 Detailed Description

AES-CCM authentication tag.

Examples:

craton-threadx/crypto/ecies-example.c, and remote-posix/crypto/ecies-example.c.

The documentation for this struct was generated from the following file:

- atlk/aes.h

6.3 aes_ccm_nonce_t Struct Reference

AES-CCM nonce.

`#include <atlk/aes.h>`

6.3.1 Detailed Description

AES-CCM nonce.

Examples:

craton-threadx/crypto/ecies-example.c, and remote-posix/crypto/ecies-example.c.

The documentation for this struct was generated from the following file:

- atlk/aes.h

6.4 aes_cmac_tag_t Struct Reference

AES-CMAC authentication tag.

```
#include <atlk/aes.h>
```

6.4.1 Detailed Description

AES-CMAC authentication tag.

Examples:

craton-threadx/crypto/aes-example.c, craton-threadx/crypto/secure-storage-example.c, and remote-posix/crypto/aes-example.c.

The documentation for this struct was generated from the following file:

- atlk/aes.h

6.5 aes_key_t Struct Reference

AES secret key.

```
#include <atlk/aes.h>
```

6.5.1 Detailed Description

AES secret key.

Examples:

craton-threadx/crypto/aes-example.c, craton-threadx/crypto/ecies-example.c, remote-posix/crypto/aes-example.c, and remote-posix/crypto/ecies-example.c.

The documentation for this struct was generated from the following file:

- atlk/aes.h

6.6 atlk_const_fragment_t Struct Reference

Read-only data fragment.

```
#include <atlk/sdk.h>
```

Data Fields

- const void ∗ fragment_ptr

  Pointer to start of fragment.
- size_t fragment_size

  Size of fragment in bytes.

6.6.1 Detailed Description

Read-only data fragment.

The documentation for this struct was generated from the following file:

- atlk/sdk.h

6.7 atlk_fragment_t Struct Reference

Data fragment.

`#include <atlk/sdk.h>`

Data Fields

- void * fragment_ptr

  Pointer to start of fragment.
- size_t fragment_size

  Size of fragment in bytes.

6.7.1 Detailed Description

Data fragment.

The documentation for this struct was generated from the following file:

- atlk/sdk.h

6.8 atlk_thread_sched_t Struct Reference

Thread scheduling parameters.

`#include <atlk/os.h>`

6.8.1 Detailed Description

Thread scheduling parameters.

The documentation for this struct was generated from the following file:

- atlk/os.h

6.9 atlk_wait_t Struct Reference

Wait option.

`#include <atlk/sdk.h>`

Data Fields

- atlk_wait_type_t wait_type

  Wait option type.
- uint32_t wait_usec

  Number of microseconds.

6.9.1 Detailed Description

Wait option.

If atlk_wait_t::wait_type is ATLK_WAIT_INTERVAL then the maximum amount of time spent waiting by the calling thread will be atlk_wait_t::wait_usec microseconds, rounded up to an implementation-dependent timer resolution.

If atlk_wait_t::wait_type is ATLK_WAIT_FOREVER then the calling thread will wait indefinitely.

Examples:

remote-posix/crypto/ecdsa-benchmark.c.

The documentation for this struct was generated from the following file:

- atlk/sdk.h

6.10 can_device_t Struct Reference

CAN device.

`#include <atlk/can_device.h>`

Data Fields

- can_tx_handler_t tx_handler

  Transmission handler function.
- void * context

  Context pointer passed to all device functions.

6.10.1 Detailed Description

CAN device.

The documentation for this struct was generated from the following file:

- atlk/can_device.h

6.11 can_hw_buffer_config_t Struct Reference

CAN HW buffer configuration.

```
#include <craton/can_driver.h>
```

Data Fields

- can_hw_id_t gmask

  Global mask for buffers 0-13.
- can_hw_id_t bmask

  Basic mask for buffer 14.
- can_hw_buffer_t buffers [15]

  CAN HW buffers.

6.11.1 Detailed Description

CAN HW buffer configuration.

Examples:

craton-threadx/can/can-hw-filter-example.c.

The documentation for this struct was generated from the following file:

- craton/can_driver.h

6.12 can_hw_buffer_t Struct Reference

CAN HW buffer (direction + ID)

```
#include <craton/can_driver.h>
```

Data Fields

- can_hw_buffer_direction_t direction

  CAN HW buffer direction.
- can_hw_id_t id

  Buffer ID.

6.12.1 Detailed Description

CAN HW buffer (direction + ID)

The documentation for this struct was generated from the following file:

- craton/can_driver.h

6.13 can_id_filter_t Struct Reference

CAN ID filter.

```
#include <atlk/can_service.h>
```

Data Fields

- can_id_t can_id

  CAN ID value.
- can_id_t can_id_mask

  CAN ID mask.

6.13.1 Detailed Description

CAN ID filter.

Examples:

craton-threadx/can/can-example.c, and craton-threadx/gnss-teseo/gnss-teseo-sou-example.c.

6.13.2 Field Documentation

can_id_t can_id_filter_t::can_id_mask CAN ID mask.

A frame with CAN ID some_can_id matches the filter if (some_can_id & can_id_mask) == (can_id & can_id_mask).

The documentation for this struct was generated from the following file:

- atlk/can_service.h

6.14 can_socket_config_t Struct Reference

CAN socket configuration parameters.

```
#include <atlk/can_service.h>
```

Data Fields

- can_device_id_t device_id

  Ingress/egress CAN device ID.
- const can_id_filter_t ∗ filter_array_ptr

  Receive-side CAN message ID filter array.
- size_t filter_array_size

  Receive-side CAN message ID filter array size.

6.14.1 Detailed Description

CAN socket configuration parameters.

Examples:

craton-threadx/can/can-example.c, and craton-threadx/gnss-teseo/gnss-teseo-sou-example.c.

The documentation for this struct was generated from the following file:

- atlk/can_service.h

6.15 cc3100_config_t Struct Reference

CC3100 configuration parameters.

```
#include <craton/cc3100_driver.h>
```

Data Fields

- gpio_num_t spi_gpio_srdy

  Ready signal from CC3100 towards CRATON.
- gpio_num_t spi_gpio_hibernate

  Hibernate signal from CRATON towards CC3100.
- hdmac_channel_id_t spi_dma_rx_channel

  SPI RX DMA channel.
- hdmac_channel_id_t spi_dma_tx_channel

  SPI TX DMA channel.
- spi_device_id_t spi_device

  SPI device number which interacts with CC3100.
- atlk_thread_sched_t workqueue_sched

  Work queue thread scheduling parameters.
- atlk_thread_sched_t sl_task_sched

  Simple Link task thread scheduling parameters.

6.15.1 Detailed Description

CC3100 configuration parameters.

The documentation for this struct was generated from the following file:

- craton/cc3100_driver.h

6.16 dhcp_client_config_t Struct Reference

DHCP client configuration parameters.

```
#include <atlk/dhcp_client.h>
```

Data Fields

- dhcp_client_bound_handler_t bound_handler

  DHCP client bound handler.
- atlk_thread_sched_t sched_params

  DHCP client thread scheduling parameters.

6.16.1 Detailed Description

DHCP client configuration parameters.

The documentation for this struct was generated from the following file:

- atlk/dhcp_client.h

6.17 ecc_fast_verification_signature_t Struct Reference

ECDSA signature for fast verification.

```
#include <atlk/ecc.h>
```

Data Fields

- ecc_point_t R_point

  R point.
- ecc_scalar_t s_scalar

  s scalar

6.17.1 Detailed Description

ECDSA signature for fast verification.

Examples:

craton-threadx/crypto/ecdsa-example.c, and remote-posix/crypto/ecdsa-example.c.

The documentation for this struct was generated from the following file:

- atlk/ecc.h

6.18 ecc_pma_params_t Struct Reference

Elliptic curve point multiply-add parameters.

```
#include <atlk/ecc_service.h>
```

Data Fields

- ecc_point_t point

  ECC point to be multiplied.
- ecc_scalar_t multiplier

  Scalar to multiply by.
- ecc_point_t addend

  ECC point to add.

6.18.1 Detailed Description

Elliptic curve point multiply-add parameters.

The documentation for this struct was generated from the following file:

- atlk/ecc_service.h

6.19 ecc_point_t Struct Reference

Point on an elliptic curve.

```
#include <atlk/ecc.h>
```

Data Fields

- ecc_point_type_t point_type

  Point representation type.
- ecc_scalar_t x_coordinate

  X coordinate.
- ecc_scalar_t y_coordinate

  Y coordinate.

6.19.1 Detailed Description

Point on an elliptic curve.

Examples:

craton-threadx/crypto/ecdsa-example.c, craton-threadx/crypto/ecies-example.c, remote-posix/crypto/ecdsa-example.-c, and remote-posix/crypto/ecies-example.c.

6.19.2 Field Documentation

ecc_scalar_t ecc_point_t::y_coordinate Y coordinate.
Valid only if `type` is equal to ecc_point_type_t::ECC_POINT_UNCOMPRESSED.

Examples:

craton-threadx/crypto/ecdsa-example.c, craton-threadx/crypto/ecies-example.c, remote-posix/crypto/ecdsa-example.c, and remote-posix/crypto/ecies-example.c.

The documentation for this struct was generated from the following file:

- atlk/ecc.h

6.20 ecc_request_context_t Struct Reference

ECC request context.

```
#include <atlk/ecc_service.h>
```

Data Fields

- ecc_request_id_t request_id
  Request ID.
- ecc_request_type_t request_type
  Request type.
- ecc_curve_t curve
  Elliptic curve identifier.

6.20.1 Detailed Description

ECC request context.
The documentation for this struct was generated from the following file:

- atlk/ecc_service.h

6.21 ecc_request_t Struct Reference

ECC request.

```
#include <atlk/ecc_service.h>
```

Data Fields

- ecc_request_context_t context
  Request context.
- ecc_verify_params_t verify_params
  ECDSA verification parameters.
- ecc_sign_params_t sign_params
  ECDSA signing parameters.
- ecc_pma_params_t pma_params
  Elliptic curve point multiply-add parameters.

6.21.1 Detailed Description

ECC request.

Examples:

craton-threadx/crypto/ecdsa-benchmark.c, craton-threadx/crypto/ecdsa-example.c, remote-posix/crypto/ecdsa-benchmark.c, and remote-posix/crypto/ecdsa-example.c.

The documentation for this struct was generated from the following file:

- atlk/ecc_service.h

6.22 ecc_response_t Struct Reference

ECC response.

```
#include <atlk/ecc_service.h>
```

Data Fields

- ecc_request_context_t context

  Original request context.
- ecc_rc_t rc

  ECC return code.
- union {

  ecc_fast_verification_signature_t sign_result

  ECDSA signature for fast verification.

  ecc_point_t pma_result

  Elliptic curve point after multiply-add operation.

  } result

  Response result.

6.22.1 Detailed Description

ECC response.

Examples:

craton-threadx/crypto/ecdsa-benchmark.c, craton-threadx/crypto/ecdsa-example.c, remote-posix/crypto/ecdsa-benchmark.c, and remote-posix/crypto/ecdsa-example.c.

6.22.2 Field Documentation

ecc_point_t ecc_response_t::pma_result Elliptic curve point after multiply-add operation.

Value is valid only if ecc_response_t::context::request_type is equal to ECC_REQUEST_TYPE_PMA.

ecc_rc_t ecc_response_t::rc ECC return code.

Specifically, if the original request was for ECDSA verification, then the supplied ECDSA signature was found valid if and only if the return code is equal to ECC_OK.

Examples:

craton-threadx/crypto/ecdsa-benchmark.c, craton-threadx/crypto/ecdsa-example.c, remote-posix/crypto/ecdsa-benchmark.c, and remote-posix/crypto/ecdsa-example.c.

union { ... } ecc_response_t::result Response result.

Valid only if ecc_response_t::rc is equal to ECC_OK.

ecc_fast_verification_signature_t ecc_response_t::sign_result ECDSA signature for fast verification.

Value is valid only if ecc_response_t::context::request_type is equal to ECC_REQUEST_TYPE_SIGN.

The documentation for this struct was generated from the following file:

- atlk/ecc_service.h

6.23 ecc_scalar_t Struct Reference

Big integer type for use with ECC.

```
#include <atlk/ecc.h>
```

Data Fields

- uint32_t value [12]

  Unsigned integer in base 2^32.

6.23.1 Detailed Description

Big integer type for use with ECC.

6.23.2 Field Documentation

uint32_t ecc_scalar_t::value[12] Unsigned integer in base 2^32.

Note: Least significant word appears first, most significant word appears last. All words are in native endianness.

The documentation for this struct was generated from the following file:

- atlk/ecc.h

6.24 ecc_sign_params_t Struct Reference

ECDSA signing parameters.

```
#include <atlk/ecc_service.h>
```

Data Fields

- ecc_scalar_t private_key

  Private key.
- sha_digest_t digest

  Hash digest.

6.24.1 Detailed Description

ECDSA signing parameters.

The documentation for this struct was generated from the following file:

- atlk/ecc_service.h

6.25 ecc_signature_t Struct Reference

ECDSA signature.

```
#include <atlk/ecc.h>
```

Data Fields

- ecc_scalar_t r_scalar

  r scalar
- ecc_scalar_t s_scalar

  s scalar

6.25.1 Detailed Description

ECDSA signature.

Examples:

craton-threadx/crypto/ecdsa-example.c, and remote-posix/crypto/ecdsa-example.c.

The documentation for this struct was generated from the following file:

- atlk/ecc.h

6.26 ecc_verify_params_t Struct Reference

ECDSA verification parameters.

```
#include <atlk/ecc_service.h>
```

Data Fields

- ecc_point_t public_key

    Public key.
- sha_digest_t digest

    Hash digest.
- ecc_signature_t signature

    ECDSA signature.

6.26.1 Detailed Description

ECDSA verification parameters.

The documentation for this struct was generated from the following file:

- atlk/ecc_service.h

6.27 ecies_authentication_tag_t Struct Reference

ECIES authentication tag.

```
#include <atlk/ecies.h>
```

6.27.1 Detailed Description

ECIES authentication tag.

Examples:

craton-threadx/crypto/ecies-example.c, and remote-posix/crypto/ecies-example.c.

The documentation for this struct was generated from the following file:

- atlk/ecies.h

6.28 eui48_t Struct Reference

48-bit Extended Unique Identifier

```
#include <atlk/eui48.h>
```

6.28.1 Detailed Description

48-bit Extended Unique Identifier

Examples:

craton-threadx/v2x-emulator/v2x-emulator-over-udp-example.c.

The documentation for this struct was generated from the following file:

- atlk/eui48.h

6.29 exception_arc_info_t Struct Reference

ARC exception structure containing all necessary information upon exception.

```
#include <craton/exception_arc.h>
```

Data Fields

- uint32_t mode

    Exception mode flag.
- exception_arc_type_t type

    Exception type.
- uint32_t address

    Exception address.
- uint32_t watchdog_counter

    SW watchdog counter upon exception.
- exception_arc_regs_t regs

    Exception registers.
- uint32_t thread_address

    Pointer to thread structure upon exception.
- char thread_name [128]

    Thread name.
- size_t thread_stack_size

    Thread stack size.
- uint8_t thread_stack [512]

    Thread stack.

6.29.1 Detailed Description

ARC exception structure containing all necessary information upon exception.

6.29.2 Field Documentation

uint32_t exception_arc_info_t::mode Exception mode flag.
Set when ARC enters exeption mode.
The documentation for this struct was generated from the following file:

- craton/exception_arc.h

6.30 exception_arc_regs_t Struct Reference

ARC exception registers.
`#include <craton/exception_arc.h>`

6.30.1 Detailed Description

ARC exception registers.
The documentation for this struct was generated from the following file:

- craton/exception_arc.h

6.31 exception_arm_info_t Struct Reference

ARM exception structure containing all necessary information upon exception.
`#include <craton/exception_arm.h>`

Data Fields

- exception_arm_fault_operation_t operation

  Operation upon exception.
- exception_arm_type_t type

  Exception type.
- exception_arm_reason_t reason

  Exception reason.
- uint32_t address

  Exception address.
- exception_arm_regs_t arm_regs

  Exception registers.
- uint32_t thread_address

  Pointer to thread structure upon exception.
- char thread_name [128]

  Thread name.
- size_t thread_stack_size

  Thread stack size.
- uint8_t thread_stack [512]

  Thread stack.

6.31.1 Detailed Description

ARM exception structure containing all necessary information upon exception.
The documentation for this struct was generated from the following file:

- craton/exception_arm.h

6.32 exception_arm_regs_t Struct Reference

ARM exception registers.

```
#include <craton/exception_arm.h>
```

6.32.1 Detailed Description

ARM exception registers.
The documentation for this struct was generated from the following file:

- craton/exception_arm.h

6.33 fs_dirstat Struct Reference

Directory statistics structure.

```
#include <craton/fs.h>
```

Data Fields

- unsigned int traversal_depth

  Traversal depth.
- uint64_t size_bytes

  Total size of files in traversed directory in bytes.
- unsigned int num_of_dirs

  Total number of directories in traversed directory.
- unsigned int num_of_files

  Total number of files in traversed directory.

6.33.1 Detailed Description

Directory statistics structure.

Examples:

craton-threadx/fs/fs-example.c.

The documentation for this struct was generated from the following file:

- craton/fs.h

6.34 gnss_config_t Struct Reference

GNSS configuration parameters.
`#include <atlk/gnss.h>`

Data Fields

- gnss_model_t model
  GNSS model.
- gnss_hw_reset_t hw_reset
  GNSS HW reset (optional)
- int wd_enabled
  Whether GNSS watchdog is enabled.
- atlk_thread_sched_t wd_sched_params
  GNSS watchdog thread scheduling parameters.
- const char ∗ dev_name
  I/O device name.
- uart_speed_bps_t nmea_speed_bps
  Expected NMEA speed in bps during init.
- const char ∗ nmea_cycle_ender_10hz
  NMEA sentence address for 10 Hz cycles.
- const char ∗ nmea_cycle_ender_1hz
  NMEA sentence address for 1 Hz cycles.
- atlk_thread_sched_t sched_params
  Input thread scheduling parameters.
- nav_data_handler_t handler
  Navigation data frame handler.
- nav_service_t ∗ service
  Navigation service instance.

6.34.1 Detailed Description

GNSS configuration parameters.

Examples:

remote-posix/gnss/gnss-example.c.

6.34.2 Field Documentation

const char∗ gnss_config_t::nmea_cycle_ender_10hz NMEA sentence address for 10 Hz cycles.

Expected NMEA cycle ender of all cycles except for cycles in which NMEA time-stamp is round. String must include '$' sign at its start.

For example:

```
gnss_config_t config = GNSS_CONFIG_INIT;
config.nmea_cycle_ender_1hz = "$PSTMCPU";
config.nmea_cycle_ender_10hz = "$XXGLL";
```

Note

Setting an invalid talker ID means "don't care" (for example, setting to "$XXGLL", means GLL with any talker ID is cycle ender).

Examples:

remote-posix/gnss/gnss-example.c.

int gnss_config_t::wd_enabled Whether GNSS watchdog is enabled.
When enabled, specifying gnss_config_t.hw_reset is mandatory.
The documentation for this struct was generated from the following file:

- atlk/gnss.h

6.35 gnss_reset_params_t Struct Reference

GNSS reset parameters.

```
#include <atlk/gnss.h>
```

Data Fields

- gnss_reset_type_t reset_type

  GNSS reset type.
- gnss_start_type_t start_type

  GNSS start type.
- uint32_t cold_start_flags

  GNSS cold start flags bitmask.

6.35.1 Detailed Description

GNSS reset parameters.
The documentation for this struct was generated from the following file:

- atlk/gnss.h

6.36 gnss_teseo_fw_update_params_t Struct Reference

Teseo firmware update parameters.

```
#include <atlk/gnss_teseo.h>
```

Data Fields

- const void ∗ fw_image

  Pointer to Teseo firmware image.
- size_t fw_image_size

  Teseo firmware image size in bytes.
- uart_speed_bps_t nmea_speed_bps

  UART speed used to send FW Upgrade command in bits/s.
- uart_speed_bps_t download_speed_bps

  UART speed used to download firmware image in bits/s.
- int erase_nvm_area

  Whether to erase NVM area.
- size_t nvm_area_size_kb

  NVM area size in kilobytes.
- int recovery_mode

  Whether to run in recovery mode.
- atlk_thread_sched_t sched_params

  Input thread scheduling parameters.

6.36.1 Detailed Description

Teseo firmware update parameters.

Examples:

craton-threadx/gnss-teseo/gnss-teseo-fw-update-example.c.

6.36.2 Field Documentation

uart_speed_bps_t gnss_teseo_fw_update_params_t::nmea_speed_bps UART speed used to send FW Upgrade command in bits/s.

Speed should be set to the speed currently configured in Teseo firmware. At the end of the update, UART speed will be reconfigured to this speed.

When speed is set to GNSS_TESEO_FW_UPDATE_NMEA_SPEED_BPS_AUTO, speed is automatically chosen (both during and after the update). Generally, automatically choosing speed is less reliable.

Examples:

craton-threadx/gnss-teseo/gnss-teseo-fw-update-example.c.

size_t gnss_teseo_fw_update_params_t::nvm_area_size_kb NVM area size in kilobytes.

Default value is chosen when gnss_teseo_fw_update_params_t::nvm_area_size_kb equals zero.

Examples:

craton-threadx/gnss-teseo/gnss-teseo-fw-update-example.c.

int gnss_teseo_fw_update_params_t::recovery_mode Whether to run in recovery mode.

Note

Recovery mode will likely fail if gnss_config_t.hw_reset procedure is not registered.

Examples:

craton-threadx/gnss-teseo/gnss-teseo-fw-update-example.c.

The documentation for this struct was generated from the following file:

- atlk/gnss_teseo.h

6.37 gnss_teseo_sou_config_t Struct Reference

Teseo SOU configuration parameters.

```
#include <atlk/gnss_teseo.h>
```

Data Fields

- gnss_teseo_sou_operating_mode_t operating_mode

  SOU operating mode.
- const sensor_value_params_t ∗ gyro_1axis_params_ptr

  Pointer to gyro 1-axis parameters.
- const sensor_value_params_t ∗ wheels_speed_params_ptr

  Pointer to wheels speed parameters.
- atlk_thread_sched_t sched_params

  SOU feeder thread scheduling parameters.

6.37.1 Detailed Description

Teseo SOU configuration parameters.

Examples:

craton-threadx/gnss-teseo/gnss-teseo-sou-example.c.

The documentation for this struct was generated from the following file:

- atlk/gnss_teseo.h

6.38 hsm_capability_info_t Struct Reference

HSM capability information.

```
#include <atlk/hsm_service.h>
```

Data Fields

- uint32_t max_num_of_cells

  Maximum number of NVM cells supported.
- uint32_t current_num_of_cells

  Current number of NVM cells configured.
- uint32_t max_num_of_cell_ranges_for_csr

  Maximum number of cell ranges supported by hsm_csr_ecdsa_public_keys_sign.
- uint32_t max_ecies_key_size

  Maximum ECIES key size supported by hsm_ecies_key_derive.
- uint32_t max_ecies_kdf_param_size

  Maximum ECIES key derivation parameter size supported by hsm_ecies_key_derive.

6.38.1 Detailed Description

HSM capability information.

Examples:

craton-threadx/crypto/ecdsa-example.c, craton-threadx/crypto/ecies-example.c, remote-posix/crypto/ecdsa-example.c, and remote-posix/crypto/ecies-example.c.

The documentation for this struct was generated from the following file:

- atlk/hsm_service.h

6.39 hsm_cell_range_t Struct Reference

HSM secure storage cell range.

```
#include <atlk/hsm.h>
```

Data Fields

- hsm_cell_index_t first_cell_index

  First cell index in range.
- uint32_t num_of_cells

  Number of cells in range.

6.39.1 Detailed Description

HSM secure storage cell range.

The documentation for this struct was generated from the following file:

- atlk/hsm.h

6.40 hsm_csr_random_prefix_t Struct Reference

CSR random prefix.

```
#include <atlk/hsm_service.h>
```

6.40.1 Detailed Description

CSR random prefix.

The documentation for this struct was generated from the following file:

- atlk/hsm_service.h

6.41 hsm_ecc_private_key_info_t Struct Reference

Private key information.

```
#include <atlk/hsm_service.h>
```

Data Fields

- ecc_curve_t key_curve

  Elliptic curve used with this key.
- hsm_private_key_type_t key_type

  Type of key.
- hsm_public_key_algorithm_t key_algorithm

  Intended algorithm for key.

6.41.1 Detailed Description

Private key information.

Examples:

craton-threadx/crypto/ecdsa-example.c, craton-threadx/crypto/ecies-example.c, remote-posix/crypto/ecdsa-example.-c, and remote-posix/crypto/ecies-example.c.

The documentation for this struct was generated from the following file:

- atlk/hsm_service.h

6.42 hsm_emulator_config_t Struct Reference

HSM emulator configuration parameters.

```
#include <atlk/hsm_emulator.h>
```

Data Fields

- ecc_service_t ∗ ecc_service_ptr

  ECC service pointer to be used by the HSM emulator.
- char ∗ nvm_file_path

  NVM filename.
- aes_key_t host_nvm_authentication_key

  AES-128 key to provide host NVM storage integrity.
- aes_key_t host_nvm_encryption_key

  AES-128 key to provide host NVM storage confidentiality.

6.42.1 Detailed Description

HSM emulator configuration parameters.

Examples:

craton-threadx/crypto/ecdsa-example.c, craton-threadx/crypto/ecies-example.c, craton-threadx/crypto/secure-storage-example.c, remote-posix/crypto/ecdsa-example.c, and remote-posix/crypto/ecies-example.c.

6.42.2 Field Documentation

char∗ hsm_emulator_config_t::nvm_file_path NVM filename.

Using NULL means RAM is used for storage of private keys.

Examples:

craton-threadx/crypto/ecdsa-example.c, craton-threadx/crypto/ecies-example.c, craton-threadx/crypto/secure-storage-example.c, remote-posix/crypto/ecdsa-example.c, and remote-posix/crypto/ecies-example.c.

The documentation for this struct was generated from the following file:

- atlk/hsm_emulator.h

6.43 hsm_nvm_config_t Struct Reference

HSM NVM configuration.

```
#include <atlk/hsm_service.h>
```

Data Fields

- uint32_t num_of_cells

    Number of used NVM cells.

6.43.1 Detailed Description

HSM NVM configuration.

Examples:

craton-threadx/crypto/ecdsa-example.c, craton-threadx/crypto/ecies-example.c, craton-threadx/crypto/secure-storage-example.c, remote-posix/crypto/ecdsa-example.c, and remote-posix/crypto/ecies-example.c.

The documentation for this struct was generated from the following file:

- atlk/hsm_service.h

6.44 http_server_config_t Struct Reference

HTTP server configuration parameters.

```
#include <atlk/http_server.h>
```

Data Fields

- const char ∗ default_path_prefix

    Default path prefix (currently unsupported)
- atlk_thread_sched_t sched_params

    HTTP thread scheduling parameters.

6.44.1 Detailed Description

HTTP server configuration parameters.

Examples:

craton-threadx/net/http-example.c.

The documentation for this struct was generated from the following file:

- atlk/http_server.h

6.45 http_url_entry_t Struct Reference

URL entry.

```
#include <atlk/http_server.h>
```

Data Fields

- const char * url

  URL address.
- http_url_handler_t url_handler

  URL handler.

6.45.1 Detailed Description

URL entry.

Examples:

craton-threadx/net/http-example.c.

The documentation for this struct was generated from the following file:

- atlk/http_server.h

6.46 i2c_driver_config_t Struct Reference

I2C initialization configuration descriptor.

```
#include <craton/i2c_driver.h>
```

Data Fields

- uint16_t clock_divisor

  Ratio between the system clock frequency and the desired bus clock frequency.

6.46.1 Detailed Description

I2C initialization configuration descriptor.

6.46.2 Field Documentation

uint16_t i2c_driver_config_t::clock_divisor Ratio between the system clock frequency and the desired bus clock frequency.

Must be a multiple of I2C_CLOCK_DIVISOR_STEP.

E.g. clock divisor value 400 means 100 kHz bus clock frequency.

The documentation for this struct was generated from the following file:

- craton/i2c_driver.h

6.47 i2s_dma_playback_t Struct Reference

I2S DMA playback descriptor.

```
#include <craton/i2s_driver.h>
```

Data Fields

- const void * left_sample_buffer_ptr

  Pointer to sample buffer for left channel playback.
- const void * right_sample_buffer_ptr

  Pointer to sample buffer for right channel playback.
- size_t sample_buffer_size

  Both sample buffers' size in bytes.
- void * context

  User context.
- void(* completion_handler)(struct i2s_dma_playback *playback)

  DMA playback completion handler.
- i2s_playback_status_t playback_status

  Current playback status.

6.47.1 Detailed Description

I2S DMA playback descriptor.

Examples:

craton-threadx/i2s/i2s-example.c.

6.47.2 Field Documentation

void∗ i2s_dma_playback_t::context User context.
Not used by the driver.

const void∗ i2s_dma_playback_t::left_sample_buffer_ptr Pointer to sample buffer for left channel playback.

Remarks

Must be aligned to cache line size.

Examples:

craton-threadx/i2s/i2s-example.c.

const void∗ i2s_dma_playback_t::right_sample_buffer_ptr Pointer to sample buffer for right channel playback.

Remarks

Must be aligned to cache line size.
May be equal to i2s_dma_playback_t::left_sample_buffer_ptr.

Examples:

craton-threadx/i2s/i2s-example.c.

size_t i2s_dma_playback_t::sample_buffer_size Both sample buffers' size in bytes.

Remarks

Must be a multiple of cache line size.

Examples:

craton-threadx/i2s/i2s-example.c.

The documentation for this struct was generated from the following file:

- craton/i2s_driver.h

6.48 i2s_driver_config_t Struct Reference

I2S Driver configuration.
`#include <craton/i2s_driver.h>`

Data Fields

- hdmac_channel_id_t left_dma_channel

  I2S DMA left channel ID.
- hdmac_channel_id_t right_dma_channel

  I2S DMA right channel ID.
- gpio_num_t mute_gpio

  I2S mute audio GPIO.

6.48.1 Detailed Description

I2S Driver configuration.

The documentation for this struct was generated from the following file:

- craton/i2s_driver.h

6.49 imq_queue_config_t Struct Reference

IMQ queue configuration.

```
#include <craton/imq.h>
```

Data Fields

- uint16_t queue_mtu

  IMQ queue MTU.
- uint32_t queue_length

  IMQ queue length.

6.49.1 Detailed Description

IMQ queue configuration.

The documentation for this struct was generated from the following file:

- craton/imq.h

6.50 imq_service_config_t Struct Reference

IMQ service configuration.

```
#include <craton/imq.h>
```

Data Fields

- imq_queue_config_t client_to_server_config

  Client to server queue configuration.
- imq_queue_config_t server_to_client_config

  Server to client queue configuration.
- char * service_name

  IMQ service name.

6.50.1 Detailed Description

IMQ service configuration.

Examples:

craton-threadx/imq/imq-echo-server.c.

The documentation for this struct was generated from the following file:

- craton/imq.h

6.51 imq_socket_t Struct Reference

IMQ socket.

```
#include <craton/imq.h>
```

6.51.1 Detailed Description

IMQ socket.

Examples:

craton-threadx/imq/imq-client.c, and craton-threadx/imq/imq-echo-server.c.

The documentation for this struct was generated from the following file:

- craton/imq.h

6.52 memc_chip_select_info_t Struct Reference

MEMC chip-select information.

```
#include <craton/memc.h>
```

Data Fields

- void ∗ region_ptr

  Pointer to start of mapped region.
- size_t region_size

  Size of mapped region in bytes.

6.52.1 Detailed Description

MEMC chip-select information.

The documentation for this struct was generated from the following file:

- craton/memc.h

6.53 mibstat_canDevEntry_t Struct Reference

CAN device status.

```
#include <atlk/mibs/can-mibstat.h>
```

Data Fields

- uint64_t canTxFrameCnt

  Count of transmitted frames.
- uint64_t canRxFrameCnt

  Count of received frames.
- uint64_t canRxSwFilteredCnt

  Count of received frames which were filtered by SW.
- uint32_t canTxErrorCnt

  Count of errors due to transmitted frames.
- uint32_t canRxErrorCnt

  Count of errors due to received frames.
- uint32_t canTxHwErrorCnt

  Count of HW errors due to transmitted frames.
- uint32_t canRxHwErrorCnt

  Count of HW errors due to received frames.
- uint32_t canInterruptErrorCnt

  Count of generated error interrupts.
- uint32_t canBlockAllocErrorCnt

  Count of block allocation failures.
- uint32_t canTxQueueErrorCnt

  Count of transmission queueing failures.

6.53.1 Detailed Description

CAN device status.

The documentation for this struct was generated from the following file:

- atlk/mibs/can-mibstat.h

6.54 mibstat_canMib_t Struct Reference

CAN status.

```
#include <atlk/mibs/can-mibstat.h>
```

6.54.1 Detailed Description

CAN status.

The documentation for this struct was generated from the following file:

- atlk/mibs/can-mibstat.h

6.55 mibstat_ethMib_t Struct Reference

Ethernet status.

```
#include <atlk/mibs/eth-mibstat.h>
```

Data Fields

- uint32_t ethTxFrameCnt

  Count of transmitted frames.
- uint32_t ethTxGoodFrameCnt

  Count of transmitted frames with no errors.
- uint32_t ethRxFrameCnt

  Count of received frames.
- uint32_t ethRxCrcErrorCnt

  Count of received frames with CRC errors.

6.55.1 Detailed Description

Ethernet status.

The documentation for this struct was generated from the following file:

- atlk/mibs/eth-mibstat.h

6.56 mibstat_profilingMib_t Struct Reference

Profiling status.

```
#include <atlk/mibs/profiling-mibstat.h>
```

Data Fields

- uint64_t profilingTotalArmCnt

  Total number of cycles since last reset on ARM CPU.
- uint64_t profilingIdleArmCnt

  Cycles in idle state on ARM CPU.
- uint64_t profilingIsrArmCnt

  Cycles consumed by ISRs on ARM CPU.
- mibstat_profilingThreadEntry_t profilingThreadsArm [32]

  Status per thread on ARM CPU.
- uint64_t profilingTotalArc1Cnt

  Total number of cycles since last reset on ARC1 CPU.

- uint64_t profilingIdleArc1Cnt

    Cycles in idle state on ARC1 CPU.
- uint64_t profilingIsrArc1Cnt

    Cycles consumed by ISRs on ARC1 CPU.
- mibstat_profilingThreadEntry_t profilingThreadsArc1 [32]

    Status per thread on ARC1 CPU.
- uint64_t profilingTotalArc2Cnt

    Total number of cycles since last reset on ARC2 CPU.
- uint64_t profilingIdleArc2Cnt

    Cycles in idle state on ARC2 CPU.
- uint64_t profilingIsrArc2Cnt

    Cycles consumed by ISRs on ARC2 CPU.
- mibstat_profilingThreadEntry_t profilingThreadsArc2 [32]

    Status per thread on ARC2 CPU.

6.56.1 Detailed Description

Profiling status.

The documentation for this struct was generated from the following file:

- atlk/mibs/profiling-mibstat.h

6.57 mibstat_profilingThreadEntry_t Struct Reference

Profiling thread status.

```
#include <atlk/mibs/profiling-mibstat.h>
```

Data Fields

- uint64_t profilingThreadCyclesCnt

    Count of cycles used by thread.
- char profilingThreadName [32]

    Thread name.

6.57.1 Detailed Description

Profiling thread status.

The documentation for this struct was generated from the following file:

- atlk/mibs/profiling-mibstat.h

6.58 mibstat_slx97Mib_t Struct Reference

SLx97 status.

```
#include <atlk/mibs/slx97-mibstat.h>
```

Data Fields

- uint32_t slx97CmdWriteCnt

    Count of commands written.
- uint32_t slx97CmdExeCnt

    Count of executed commands.
- uint32_t slx97RspReadCnt

    Count of responses read.
- uint32_t slx97CmdCrcErrorCnt

    Count of command CRC errors.

- uint32_t slx97RspCrcErrorCnt

    Count of response CRC errors.
- uint32_t slx97CmdNoSecCnt

    Count of commands with no transport security.
- uint32_t slx97CmdSecCnt

    Count of commands with transport security.
- uint32_t slx97RspNoSecCnt

    Count of responses with no transport security.
- uint32_t slx97RspSecCnt

    Count of responses with transport security.
- uint32_t slx97RspErrorCnt

    Count of error responses.

6.58.1 Detailed Description

SLx97 status.

The documentation for this struct was generated from the following file:

- atlk/mibs/slx97-mibstat.h

6.59 mibstat_spi2uartMib_t Struct Reference

SPI2UART status.

```
#include <atlk/mibs/spi2uart-mibstat.h>
```

Data Fields

- uint32_t spi2uartTxFrameCnt

    Count of transmitted frames.
- uint32_t spi2uartRxFrameCnt

    Count of received frames.
- uint32_t spi2uartRxOverrunErrorCnt

    Count of overrun error due to received frames.
- uint32_t spi2uartRxParityErrorCnt

    Count of parity error due to received frames.
- uint32_t spi2uartRxFramingErrorCnt

    Count of framing error due to received frames.

6.59.1 Detailed Description

SPI2UART status.

The documentation for this struct was generated from the following file:

- atlk/mibs/spi2uart-mibstat.h

6.60 mq_attr Struct Reference

6.60.1 Detailed Description

Examples:

craton-threadx/posix/posix-example.c.

The documentation for this struct was generated from the following file:

- tx_posix.h

6.61 mqd_t Struct Reference

The documentation for this struct was generated from the following file:

- tx_posix.h

6.62 nav_data_t Struct Reference

Navigation data frame.

```
#include <atlk/nav.h>
```

Data Fields

- uint32_t data_type

  Navigation data frame type.
- nav_fix_t fix

  Navigation fix data frame.
- nav_satellite_report_t satellite_report

  Satellite report data frame.

6.62.1 Detailed Description

Navigation data frame.

Examples:

craton-threadx/nav/nav-data-example.c, and remote-posix/gnss/gnss-example.c.

The documentation for this struct was generated from the following file:

- atlk/nav.h

6.63 nav_fix_t Struct Reference

Navigation fix data frame.

```
#include <atlk/nav.h>
```

Data Fields

- nav_time_t time

  Timestamp of the moment when fix was taken.
- double position_latitude_deg

  Latitude in units of one degree.
- double position_longitude_deg

  Longitude in units of one degree.
- double position_altitude_m

  Altitude above the reference ellipsoid in units of one meter.
- double movement_horizontal_direction_deg

  Horizontal movement direction relative to true north (clockwise) in units of one degree.
- double movement_horizontal_speed_mps

  Horizontal movement speed in units of one meter per second.
- double movement_vertical_speed_mps

  Vertical movement speed in units of one meter per second.
- double error_time_s

  Time standard deviation in units of one second.
- double error_position_horizontal_major_axis_direction_deg

  Horizontal position standard deviation ellipse: major axis direction relative to true north (clockwise) in units of one degree.
- double error_position_horizontal_semi_major_axis_length_m

Horizontal position standard deviation ellipse: semi-major axis length in meters.
- double error_position_horizontal_semi_minor_axis_length_m

  Horizontal position standard deviation ellipse: semi-minor axis length in meters.
- double error_position_altitude_m

  Vertical position standard deviation in meters.
- double error_movement_horizontal_direction_deg

  Horizontal movement direction standard deviation in units of one degree.
- double error_movement_horizontal_speed_mps

  Horizontal movement speed standard deviation in units of one meter per second.
- double error_movement_vertical_speed_mps

  Vertical movement speed standard deviation in units of one meter per second.
- nav_fix_mode_t mode

  Navigation fix mode.
- uint32_t data_source

  Navigation fix data source.
- double hdop

  Horizontal Dilution of Precision of combined GNSS solution.
- uint8_t satellites_in_use_num

  Number of satellites in use.
- uint8_t satellites_num [NAV_SATELLITES_MAX+1]

  Number of satellites in view per satellites system.
- nav_fix_user_data_t user_data

  Navigation fix user data.

6.63.1 Detailed Description

Navigation fix data frame.

Time, position, velocity and their error estimates.

The reference ellipsoid used is the one defined by WGS-84.

Any floating-point field may be set to NaN if the navigation data source didn't provide a value for that field. User programs should use the standard isnan() function to check for this condition.

Note that a non-nan position value does not necessarily mean that GNSS is in a locked state. Please check the value of nav_fix_t.mode

Examples:

craton-threadx/gnss/gnss-integration-example.c, craton-threadx/nav/nav-example.c, craton-threadx/nav/nav-trace.h, and remote-posix/gnss/gnss-example.c.

6.63.2 Field Documentation

uint32_t nav_fix_t::data_source Navigation fix data source.

A bitmask of NAV_FIX_USES_GNSS, NAV_FIX_USES_DGNSS, NAV_FIX_USES_DR.

Note

In practice, a single value is always used (i.e. this is not really used as a bitmask). This is in alignment with NMEA data source reporting.

Examples:

craton-threadx/nav/nav-trace.h, and remote-posix/gnss/gnss-example.c.

double nav_fix_t::error_position_horizontal_major_axis_direction_deg Horizontal position standard deviation ellipse: major axis direction relative to true north (clockwise) in units of one degree.

When available, its value is in range [0.0, 360.0]

Examples:

craton-threadx/nav/nav-trace.h, and remote-posix/gnss/gnss-example.c.

double nav_fix_t::movement_horizontal_direction_deg Horizontal movement direction relative to true north (clockwise) in units of one degree.

The angle between true north and the projection of the velocity vector onto the ground plane. Also known as "track made good" or "course over ground".

In case of a skidding vehicle, this direction may differ from the vehicle's heading.

When available, its value is in range [0.0, 360.0]

Examples:

craton-threadx/nav/nav-trace.h, and remote-posix/gnss/gnss-example.c.

double nav_fix_t::movement_horizontal_speed_mps Horizontal movement speed in units of one meter per second.

The magnitude of the projection of the velocity vector onto the ground plane. Also known as "speed made good" or "speed over ground".

Examples:

craton-threadx/nav/nav-trace.h, and remote-posix/gnss/gnss-example.c.

double nav_fix_t::movement_vertical_speed_mps Vertical movement speed in units of one meter per second.

A positive value indicates upward movement, a negative value indicates downward movement. Also known as "rate of climb".

Examples:

craton-threadx/nav/nav-trace.h, and remote-posix/gnss/gnss-example.c.

double nav_fix_t::position_latitude_deg Latitude in units of one degree.

When available, its value is in range: [-90.0, 90.0]

Examples:

craton-threadx/nav/nav-trace.h, and remote-posix/gnss/gnss-example.c.

double nav_fix_t::position_longitude_deg Longitude in units of one degree.

When available, its value is in range: [-180.0, 180.0]

Examples:

craton-threadx/nav/nav-trace.h, and remote-posix/gnss/gnss-example.c.

uint8_t nav_fix_t::satellites_num[NAV_SATELLITES_MAX+1] Number of satellites in view per satellites system.

Examples:

craton-threadx/nav/nav-trace.h, and remote-posix/gnss/gnss-example.c.

The documentation for this struct was generated from the following file:

- atlk/nav.h

6.64 nav_fix_user_data_t Struct Reference

Navigation fix user data.

```
#include <atlk/nav.h>
```

Data Fields

- size_t data_size

  Size of data in user data buffer.
- uint8_t data [100]

  User data buffer.

6.64.1 Detailed Description

Navigation fix user data.

The documentation for this struct was generated from the following file:

- atlk/nav.h

6.65 nav_satellite_info_t Struct Reference

Satellite information.

```
#include <atlk/nav.h>
```

Data Fields

- uint16_t prn_num

  Satellite PRN (Pseudo-Random Noise sequence) number.
- uint16_t azimuth_deg

  Satellite azimuth in degrees, relative to true north.
- uint8_t elevation_deg

  Satellite elevation in degrees.
- uint8_t cnr_db

  Satellite carrier to noise ratio in dB.
- nav_satellites_t satellite_system

  Satellite system which this satellite belongs to.

6.65.1 Detailed Description

Satellite information.

The documentation for this struct was generated from the following file:

- atlk/nav.h

6.66 nav_satellite_report_t Struct Reference

Satellite report data frame.

```
#include <atlk/nav.h>
```

Data Fields

- nav_time_t time

  Timestamp of the moment when satellites info was taken.
- nav_satellite_info_t satellite_info_array [24]

  Satellites information.
- size_t satellite_info_array_size

  Number of satellites for which information is available.

6.66.1 Detailed Description

Satellite report data frame.

Examples:

craton-threadx/nav/nav-trace.h, and remote-posix/gnss/gnss-example.c.

The documentation for this struct was generated from the following file:

- atlk/nav.h

6.67 nav_time_t Struct Reference

Navigation timestamp.

```
#include <atlk/nav.h>
```

Data Fields

- double tai_seconds_since_2004

  Number of TAI seconds since 2004-01-01T00:00:00Z.
- int16_t leap_seconds_since_2004

  Net amount of UTC leap seconds between 2004-01-01T00:00:00Z and the point in time when fix was generated.
- unsigned int positive_leap_second

  Set to "1" iff timestamp refers to a UTC positive leap second.

6.67.1 Detailed Description

Navigation timestamp.

The documentation for this struct was generated from the following file:

- atlk/nav.h

6.68 norfl_part_info_t Struct Reference

Partition table entry.

```
#include <craton/nor_flash.h>
```

Data Fields

- uint32_t part_offset

  Offset of partition start from flash start.
- uint32_t part_size

  Partition size in bytes.
- uint32_t part_type

  Partition type.

6.68.1 Detailed Description

Partition table entry.

The documentation for this struct was generated from the following file:

- craton/nor_flash.h

6.69 norfl_part_table_t Struct Reference

Partition table.

```
#include <craton/nor_flash.h>
```

Data Fields

- norfl_part_info_t part_info [16]

  Partition table entry array.

6.69.1 Detailed Description

Partition table.

Examples:

craton-threadx/firmware/fw-update-example.c, and craton-threadx/otp/otp-example.c.

The documentation for this struct was generated from the following file:

- craton/nor_flash.h

6.70 POSIX_MSG_QUEUE Struct Reference

The documentation for this struct was generated from the following file:

- tx_posix.h

6.71 POSIX_TCB Struct Reference

The documentation for this struct was generated from the following file:

- tx_posix.h

6.72 pthread_attr_t Struct Reference

6.72.1 Detailed Description

Examples:

craton-threadx/posix/posix-example.c.

The documentation for this struct was generated from the following file:

- tx_posix.h

6.73 pthread_cond_t Struct Reference

6.73.1 Detailed Description

Examples:

craton-threadx/posix/posix-example.c.

The documentation for this struct was generated from the following file:

- tx_posix.h

6.74 pthread_condattr_t Struct Reference

The documentation for this struct was generated from the following file:

- tx_posix.h

6.75 pthread_mutex_t Struct Reference

6.75.1 Detailed Description

Examples:

craton-threadx/posix/posix-example.c.

The documentation for this struct was generated from the following file:

- tx_posix.h

6.76 pthread_mutexattr_t Struct Reference

6.76.1 Detailed Description

Examples:

craton-threadx/posix/posix-example.c.

The documentation for this struct was generated from the following file:

- tx_posix.h

6.77 pthread_once_t Struct Reference

6.77.1 Detailed Description

Examples:

craton-threadx/posix/posix-example.c.

The documentation for this struct was generated from the following file:

- tx_posix.h

6.78 remote_ip_transport_config_t Struct Reference

IP remote transport configuration.

```
#include <atlk/remote.h>
```

Data Fields

- uint32_t remote_ipv4_address

  IPv4 server address in network order.
- uint32_t max_rtt_ms

  Max round-trip time in milliseconds.
- uint32_t local_ipv4_address

  Local IPv4 address in network order.

6.78.1 Detailed Description

IP remote transport configuration.

Examples:

remote-posix/crypto/ecdsa-benchmark.c, remote-posix/crypto/ecdsa-example.c, remote-posix/crypto/ecies-example.c, remote-posix/mibs/mibs-example.c, and remote-posix/v2x/v2x-example.c.

The documentation for this struct was generated from the following file:

- atlk/remote.h

6.79 remote_ll_device_ops_t Struct Reference

Link layer driver operations.

```
#include <atlk/remote.h>
```

Data Fields

- atlk_rc_t(∗ ll_send)(void ∗device, const void ∗pdu, size_t pdu_size)

  Send link layer PDU.
- atlk_rc_t(∗ ll_receive)(void ∗device, void ∗pdu, size_t ∗pdu_size_ptr)

  Receive link layer PDU.
- atlk_rc_t(∗ ll_address_get)(void ∗device, eui48_t ∗address)

  Get MAC address of link layer device.

6.79.1 Detailed Description

Link layer driver operations.

6.79.2 Field Documentation

atlk_rc_t(∗ remote_ll_device_ops_t::ll_address_get)(void ∗device, eui48_t ∗address) Get MAC address of link layer device.

Parameters

in	*device*	Link layer device context
out	*address*	Device MAC address

atlk_rc_t(∗ remote_ll_device_ops_t::ll_receive)(void ∗device, void ∗pdu, size_t ∗pdu_size_ptr) Receive link layer PDU.
Parameters

in	*device*	Link layer device context
out	*pdu*	Link layer PDU pointer
in,out	*pdu_size_ptr*	Maximum (in) and actual (out) size of link layer PDU

atlk_rc_t(∗ remote_ll_device_ops_t::ll_send)(void ∗device, const void ∗pdu, size_t pdu_size) Send link layer PDU.
Parameters

in	*device*	Link layer device context
in	*pdu*	Link layer PDU pointer
in	*pdu_size*	Link layer PDU size

The documentation for this struct was generated from the following file:

- atlk/remote.h

6.80 remote_ll_transport_config_t Struct Reference

Link layer remote transport configuration.

```
#include <atlk/remote.h>
```

Data Fields

- const remote_ll_device_ops_t ∗ device_ops

  Link layer device operations.
- void ∗ device

  Link layer device context.
- uint32_t max_rtt_ms

  Max round-trip time in milliseconds.
- eui48_t remote_address

  MAC address of remote device.

6.80.1 Detailed Description

Link layer remote transport configuration.

The documentation for this struct was generated from the following file:

- atlk/remote.h

6.81 sem_t Struct Reference

6.81.1 Detailed Description

Examples:

craton-threadx/posix/posix-example.c.

The documentation for this struct was generated from the following file:

- tx_posix.h

6.82 sensor_value_params_t Struct Reference

Sensor value parameters.

```
#include <atlk/sensor.h>
```

Data Fields

- sensor_units_t units

  Units of (scaled) sensor value.
- sensor_value_t min

  Minimum (unscaled) sensor value.
- sensor_value_t max

  Maximum (unscaled) sensor value.
- uint16_t inverse_scaling

  Sensor value inverse scaling.

6.82.1 Detailed Description

Sensor value parameters.

Note

Sensor values which represent rotational movement (e.g. gyro) are expected to be clockwise-positive.

Examples:

craton-threadx/gnss-teseo/poti-hil.c, and craton-threadx/gnss-teseo/poti-hil.h.

6.82.2 Field Documentation

uint16_t sensor_value_params_t::inverse_scaling Sensor value inverse scaling.

Sensor value must be divided by `inverse_scaling` to be expressed in units of `units`.

The documentation for this struct was generated from the following file:

- atlk/sensor.h

6.83 sensor_wheels_speed_t Struct Reference

Vehicle wheels speed.

```
#include <atlk/sensor.h>
```

Data Fields

- sensor_value_t rear_left

  Speed of rear left wheel.
- sensor_value_t rear_right

  Speed of rear right wheel.
- sensor_value_t front_left

  Speed of front left wheel.
- sensor_value_t front_right

  Speed of front right wheel.

6.83.1 Detailed Description

Vehicle wheels speed.

Examples:

craton-threadx/gnss-teseo/gnss-teseo-sou-example.c, craton-threadx/gnss-teseo/poti-hil.c, and craton-threadx/gnss-teseo/poti-hil.h.

The documentation for this struct was generated from the following file:

- atlk/sensor.h

6.84 sha_digest_t Struct Reference

SHA digest.

```
#include <atlk/sha.h>
```

6.84.1 Detailed Description

SHA digest.

Examples:

craton-threadx/crypto/ecdsa-example.c, and remote-posix/crypto/ecdsa-example.c.

The documentation for this struct was generated from the following file:

- atlk/sha.h

6.85 signal_info Struct Reference

The documentation for this struct was generated from the following file:

- tx_posix.h

6.86 slx97_chip_info_t Struct Reference

SLx97 chip information.

```
#include <atlk/slx97.h>
```

Data Fields

- uint8_t cimIdentifier

  Chip Ident Mode Identification Byte.
- uint8_t platformIdentifier

  Platform Identifier.
- uint8_t modelIdentifier

  Chip Mode Identifier.
- uint8_t romCode [2]

  Individual coding of each ROM mask.
- uint8_t chipType_1 [2]

  Silicon Identification Number.
- uint8_t chipType_2 [4]

  Blocked Variants.
- uint8_t designStep [2]

  Design Step Code.
- uint8_t batchNumber_1

  Fab Number.
- uint8_t batchNumber_2

  Production Year.
- uint8_t batchNumber_3 [2]

  Business Week + Lot Number.
- uint8_t batchNumber_4

  rfu (Extension of Lot Number)
- uint8_t batchNumber_5

  Wafer Number.
- uint8_t chipPositionX [2]

  X-position of die on wafer.
- uint8_t chipPositionY [2]

Y-position of die on wafer.

- uint8_t trackingInfo [8]

  Tracking information.
- uint8_t firmwareIdentifier [4]

  Firmware Version Identifier.
- uint8_t bosRomVersion [4]

  Boot System ROM Version.
- uint8_t bosPatchVersion [4]

  Boot System Patch Version.
- uint8_t individualLength

  Amount of individual bytes.
- uint8_t indLowerTempLimit

  Lower limit of temperature range.
- uint8_t indUpperTempLimit

  Upper limit of temperature range.
- uint8_t indOscFreq

  Oscillator frequency in MHz.
- uint8_t indFeatures

  Individuell Features in Generic CIM.
- uint8_t indFeatures1

  Individuell Features in Generic CIM.
- uint8_t rfu [27]

  Reserved for future coding of individual CIM data.
- uint8_t atlk_firmware_ver_tuple [4]

  Autotalks firmware version tuple (little-endian 32-bit unsigned integer).
- uint8_t atlk_firmware_build_num [2]

  Autotalks firmware build number (little-endian 16-bit unsigned integer)
- uint8_t atlk_firmware_build_info [4]

  Autotalks firmware build information.

6.86.1 Detailed Description

SLx97 chip information.
Reference: SLE97 Programmer Reference User Manual, 10.2.1 "IFX-Mailbox Area".

6.86.2 Field Documentation

uint8_t slx97_chip_info_t::atlk_firmware_ver_tuple[4] Autotalks firmware version tuple (little-endian 32-bit unsigned integer).
Encoded as: (major ∗ 1e7) + (minor ∗ 1e3) + patch.
The documentation for this struct was generated from the following file:

- atlk/slx97.h

6.87 slx97_dsk_t Struct Reference

Device specific key used for SLx97 communication security.
`#include <atlk/slx97.h>`

6.87.1 Detailed Description

Device specific key used for SLx97 communication security.
The documentation for this struct was generated from the following file:

- atlk/slx97.h

6.88 slx97_host_io_config_t Struct Reference

SLx97 host I/O configuration parameters.

```
#include <craton/slx97_host.h>
```

Data Fields

- spi_device_id_t spi_device_id

  SPI device ID.
- hdmac_channel_id_t tx_dma_channel

  Tx DMA channel.
- hdmac_channel_id_t rx_dma_channel

  Rx DMA channel.
- gpio_num_t gpio_num

  GPIO number.

6.88.1 Detailed Description

SLx97 host I/O configuration parameters.

The documentation for this struct was generated from the following file:

- craton/slx97_host.h

6.89 slx97_host_sec_config_t Struct Reference

SLx97 host communication security parameters.

```
#include <atlk/slx97.h>
```

Data Fields

- uint32_t sec_version_min

  SLx97 communication security minimum version support.
- uint8_t sec_session_key_usage_limit

  SLx97 communication security session key usage limit.
- int sec_master_key_external

  Whether SLx97 communication security master key is externally generated.

6.89.1 Detailed Description

SLx97 host communication security parameters.

6.89.2 Field Documentation

uint8_t slx97_host_sec_config_t::sec_session_key_usage_limit SLx97 communication security session key usage limit.

Valid values are between 8 and 63 (both inclusive). Will only apply after hsm_nvm_init is called.

uint32_t slx97_host_sec_config_t::sec_version_min SLx97 communication security minimum version support.

Valid values are 1 or 2.

The documentation for this struct was generated from the following file:

- atlk/slx97.h

6.90 slx97_host_sec_key_t Struct Reference

SLx97 host communication security key.

```
#include <atlk/slx97.h>
```

6.90.1 Detailed Description

SLx97 host communication security key.

The documentation for this struct was generated from the following file:

- atlk/slx97.h

6.91 sntp_client_config_t Struct Reference

SNTP client configuration parameters.

```
#include <atlk/sntp_client.h>
```

Data Fields

- atlk_thread_sched_t sched_params

  SNTP thread scheduling parameters.
- sntp_client_update_handler_t update_handler

  SNTP client update callback.
- uint32_t ntp_server_address

  NTP IPv4 server address in network byte order.
- sntp_connection_type_t type

  SNTP connection type.
- uint32_t max_root_dispersion_us

  Upper limit of server clock dispersion in microseconds the client will accept.
- uint8_t min_server_stratum

  Minimum (numerically highest) stratum the client will accept.
- uint16_t unicast_poll_interval_s

  SNTP client unicast poll interval in seconds.

6.91.1 Detailed Description

SNTP client configuration parameters.

Examples:

craton-threadx/nav/system-time-benchmark.c, and craton-threadx/sntp/sntp-example.c.

6.91.2 Field Documentation

uint32_t sntp_client_config_t::max_root_dispersion_us Upper limit of server clock dispersion in microseconds the client will accept.

To disable this check, set this parameter to 0.

Examples:

craton-threadx/nav/system-time-benchmark.c, and craton-threadx/sntp/sntp-example.c.

uint8_t sntp_client_config_t::min_server_stratum Minimum (numerically highest) stratum the client will accept.

Valid range defined by SNTP_SERVER_STRATUM_MIN and SNTP_SERVER_STRATUM_MAX.

Examples:

craton-threadx/nav/system-time-benchmark.c, and craton-threadx/sntp/sntp-example.c.

uint16_t sntp_client_config_t::unicast_poll_interval_s SNTP client unicast poll interval in seconds.

RFC-4330 section 10.1: "A client MUST NOT under any conditions use a poll interval less than 15 seconds."

For testing purposes it might be desirable to set polling interval to less than 15 seconds. Please do so only when using a local server (i.e. a server in your own Network).

Examples:

craton-threadx/nav/system-time-benchmark.c, and craton-threadx/sntp/sntp-example.c.

The documentation for this struct was generated from the following file:

- atlk/sntp_client.h

6.92 sntp_info_t Struct Reference

NTP update info.

```
#include <atlk/sntp_client.h>
```

Data Fields

- uint32_t seconds

  NTP time seconds.
- uint32_t fraction

  NTP time fractions of a second.

6.92.1 Detailed Description

NTP update info.

Examples:

craton-threadx/nav/system-time-benchmark.c, and craton-threadx/sntp/sntp-example.c.

The documentation for this struct was generated from the following file:

- atlk/sntp_client.h

6.93 spi_config_t Struct Reference

SPI device configuration.

```
#include <craton/spi_driver.h>
```

Data Fields

- spi_device_id_t device_id

  SPI device ID.
- spi_mode_t device_mode

  SPI device mode.
- uint8_t data_bits

  SPI data size in bits.
- spi_clock_polarity_t clock_polarity

  Clock polarity.
- spi_clock_phase_t clock_phase

  Clock phase.
- uint8_t clock_rate_divisor1

  Clock rate divisor #1.
- uint8_t clock_rate_divisor2

  Clock rate divisor #2.
- hdmac_channel_id_t tx_dma_channel

  Tx DMA channel.
- hdmac_channel_id_t rx_dma_channel

  Rx DMA channel.

6.93.1 Detailed Description

SPI device configuration.

Effective SPI clock rate (in Hz) is:
base_rate / (clock_rate_divisor1 ∗ (1 + clock_rate_divisor2))

Examples:

craton-threadx/spi/spi-master-example.c, and craton-threadx/spi/spi-slave-example.c.

6.93.2 Field Documentation

uint8_t spi_config_t::clock_rate_divisor1 Clock rate divisor #1.

Remarks

Must be even and in the range [2, 254]. Reference: [1] clause 3.3.1.

uint8_t spi_config_t::clock_rate_divisor2 Clock rate divisor #2.

Remarks

Can be any value in the range [0, 255]. Reference: [1] clause 3.3.5.

The documentation for this struct was generated from the following file:

- craton/spi_driver.h

6.94 spi_dma_transfer_t Struct Reference

SPI DMA transfer descriptor.

```
#include <craton/spi_driver.h>
```

Data Fields

- spi_device_t ∗ device_ptr

  Pointer to SPI device object.
- const void ∗ tx_buffer_ptr

  Pointer to Tx buffer.
- size_t tx_buffer_size

  Tx buffer size in bytes.
- void ∗ rx_buffer_ptr

  Pointer to Rx buffer.
- size_t rx_buffer_size

  Rx buffer size in bytes.
- size_t tx_data_offset

  Offset (in bytes) from 'tx_buffer_ptr' to start of data to transmit.
- size_t rx_data_offset

  Offset (in bytes) from 'rx_buffer_ptr' to start of data to receive.
- size_t data_size

  Size (in bytes) of data to receive and/or transmit.
- void ∗ context

  User context.
- void(∗ completion_handler)(struct spi_dma_transfer ∗transfer)

  DMA transfer completion handler.

6.94.1 Detailed Description

SPI DMA transfer descriptor.

Examples:

craton-threadx/spi/spi-master-example.c, and craton-threadx/spi/spi-slave-example.c.

6.94.2 Field Documentation

void(∗ spi_dma_transfer_t::completion_handler)(struct spi_dma_transfer ∗transfer) DMA transfer completion handler.
Important note: This function will be executed in ISR context.

Examples:

craton-threadx/spi/spi-master-example.c, and craton-threadx/spi/spi-slave-example.c.

void∗ spi_dma_transfer_t::context User context.
Not used by the driver.

size_t spi_dma_transfer_t::data_size Size (in bytes) of data to receive and/or transmit.

Remarks

Must be divisible by 2 if spi_config_t::data_bits of used SPI device is set to 9 or more.

Examples:

craton-threadx/spi/spi-master-example.c, and craton-threadx/spi/spi-slave-example.c.

void∗ spi_dma_transfer_t::rx_buffer_ptr Pointer to Rx buffer.

Remarks

Set to NULL to disable Rx.
Must be aligned to cache line size.
If spi_config_t::data_bits of used SPI device is in [4, 7] then (8 - data_bits) most significant bits of each data byte will be zeroed. If spi_config_t::data_bits of used SPI device is in [9, 15] then (16 - data_bits) most significant bits of each data byte with odd offset (1, 3, 5, etc) will be zeroed.

Examples:

craton-threadx/spi/spi-master-example.c, and craton-threadx/spi/spi-slave-example.c.

size_t spi_dma_transfer_t::rx_buffer_size Rx buffer size in bytes.

Remarks

Ignored if spi_dma_transfer_t::rx_buffer_ptr is NULL.
Must be equal or greater than spi_dma_transfer_t::data_size.
Must be a multiple of cache line size.

Warning

The area marked with asterisks below may be modified in an arbitrary fashion by the DMA transfer.

```
                                  rx_buffer_size
                  |<------------------------------------------------>|
                  |<--rx_data_offset-->|<---data_size--->|<---***--->|
rx_buffer_ptr     ^
```

Examples:

craton-threadx/spi/spi-master-example.c, and craton-threadx/spi/spi-slave-example.c.

size_t spi_dma_transfer_t::rx_data_offset Offset (in bytes) from 'rx_buffer_ptr' to start of data to receive.

Remarks

Must be divisible by 2 if spi_config_t::data_bits of used SPI device is set to 9 or more.

const void∗ spi_dma_transfer_t::tx_buffer_ptr Pointer to Tx buffer.

Remarks

Set to NULL to disable Tx.
Must be aligned to cache line size.
If spi_config_t::data_bits of used SPI device is in [4, 7] then (8 - data_bits) most significant bits of each data byte will be ignored. If spi_config_t::data_bits of used SPI device is in [9, 15] then (16 - data_bits) most significant bits of each data byte with odd offset (1, 3, 5, etc) will be ignored.

Examples:

craton-threadx/spi/spi-master-example.c, and craton-threadx/spi/spi-slave-example.c.

size_t spi_dma_transfer_t::tx_buffer_size Tx buffer size in bytes.

Remarks

Ignored if spi_dma_transfer_t::tx_buffer_ptr is NULL.
Must be equal or greater than spi_dma_transfer_t::data_size.
Must be a multiple of cache line size.

Warning

The area marked with asterisks below shouldn't overlap with any DMA Rx buffer.

```
                                   tx_buffer_size
               |<------------------------------------------------->|
               |<--tx_data_offset-->|<---data_size--->|<---***--->|
tx_buffer_ptr  ^
```

Examples:

craton-threadx/spi/spi-master-example.c, and craton-threadx/spi/spi-slave-example.c.

size_t spi_dma_transfer_t::tx_data_offset Offset (in bytes) from 'tx_buffer_ptr' to start of data to transmit.

Remarks

Must be divisible by 2 if spi_config_t::data_bits of used SPI device is set to 9 or more.

The documentation for this struct was generated from the following file:

- craton/spi_driver.h

6.95 sys_alarm_config_t Struct Reference

System alarm configuration parameters.
`#include <craton/sys_alarm.h>`

Data Fields

- sys_alarm_gauges_t alarm_thresholds
  System alarm thresholds.
- void(∗ alarm_handler)(const sys_alarm_gauges_t ∗gauges)
  Alarm handler.
- atlk_thread_sched_t sched_params
  System alarm thread scheduling parameters.

6.95.1 Detailed Description

System alarm configuration parameters.

Examples:

craton-threadx/sys-alarm/sys-alarm-example.c.

6.95.2 Field Documentation

void(∗ sys_alarm_config_t::alarm_handler)(const sys_alarm_gauges_t ∗gauges) Alarm handler.

Handler is called at the end of a polling interval when configured thresholds are breached. Measured gague values of last polling interval are available at `gauges`.

Examples:

craton-threadx/sys-alarm/sys-alarm-example.c.

sys_alarm_gauges_t sys_alarm_config_t::alarm_thresholds System alarm thresholds.

Valid values for sys_alarm_gauges_t.cpu_utilization_percent and sys_alarm_gauges_t.heap_utilization_percent is in the range [1..99].

Examples:

craton-threadx/sys-alarm/sys-alarm-example.c.

The documentation for this struct was generated from the following file:

- craton/sys_alarm.h

6.96 sys_alarm_gauges_t Struct Reference

System alarm gauges.

```
#include <craton/sys_alarm.h>
```

Data Fields

- uint8_t cpu_utilization_percent

  Percent of non-idle CPU time in the last polling interval (5 seconds by default).
- uint8_t heap_utilization_percent

  Percent of allocated heap memory area, maximum achieved in the last polling interval (5 seconds by default).

6.96.1 Detailed Description

System alarm gauges.

Examples:

craton-threadx/sys-alarm/sys-alarm-example.c.

The documentation for this struct was generated from the following file:

- craton/sys_alarm.h

6.97 v2x_channel_id_t Struct Reference

V2X radio channel identifier.

```
#include <atlk/v2x.h>
```

Data Fields

- v2x_op_class_t op_class

  Operating class.
- v2x_channel_num_t channel_num

  Radio channel number.

6.97.1 Detailed Description

V2X radio channel identifier.

The documentation for this struct was generated from the following file:

- atlk/v2x.h

6.98 v2x_config_t Struct Reference

V2X configuration.

`#include <craton/v2x_config.h>`

Data Fields

- size_t socket_pool_size

  Number of sockets in V2X socket pool.
- size_t packet_pool_size

  Number of packets in V2X packet pool.

6.98.1 Detailed Description

V2X configuration.

The documentation for this struct was generated from the following file:

- craton/v2x_config.h

6.99 v2x_dot4_channel_end_indication_t Struct Reference

IEEE Std 1609.4-2016 service primitive MLMEX-CHEND.indication parameters.

`#include <atlk/v2x.h>`

Data Fields

- v2x_if_index_t if_index

  V2X physical interface index on which access to channel was provided.
- v2x_channel_id_t channel_id

  Radio channel identifier for which access is no longer provided.
- v2x_dot4_channel_end_reason_t reason

  Reason code.

6.99.1 Detailed Description

IEEE Std 1609.4-2016 service primitive MLMEX-CHEND.indication parameters.

Examples:

craton-threadx/dot4/dot4-channel-switching-example.c.

The documentation for this struct was generated from the following file:

- atlk/v2x.h

6.100 v2x_dot4_channel_end_request_t Struct Reference

IEEE Std 1609.4-2016 service primitive MLMEX-CHEND.request parameters.

`#include <atlk/v2x.h>`

Data Fields

- v2x_if_index_t if_index

  V2X physical interface index on which access is no longer required.
- v2x_channel_id_t channel_id

  Radio channel identifier for which access is no longer required.

6.100.1 Detailed Description

IEEE Std 1609.4-2016 service primitive MLMEX-CHEND.request parameters.
The documentation for this struct was generated from the following file:

- atlk/v2x.h

6.101 v2x_dot4_channel_start_request_t Struct Reference

IEEE Std 1609.4-2016 service primitive MLMEX-CHSTART.request parameters.

```
#include <atlk/v2x.h>
```

Data Fields

- v2x_if_index_t if_index

  V2X physical interface index on which access to channel is requested.
- v2x_channel_id_t channel_id

  Radio channel identifier to be made available for communications.
- v2x_time_slot_t time_slot

  The time slot in which alternating access is requested.
- uint8_t immediate_access

  Number of sync intervals to immediately remain on the selected channel before starting channel switching schedule.

6.101.1 Detailed Description

IEEE Std 1609.4-2016 service primitive MLMEX-CHSTART.request parameters.

Examples:

craton-threadx/dot4/dot4-channel-switching-example.c.

6.101.2 Field Documentation

uint8_t v2x_dot4_channel_start_request_t::immediate_access Number of sync intervals to immediately remain on the selected channel before starting channel switching schedule.
The value 0 means "immediate access not requested". The value 255 means "indefinite access".
The documentation for this struct was generated from the following file:

- atlk/v2x.h

6.102 v2x_emulator_config_t Struct Reference

V2X emulator configuration descriptor.

```
#include <craton/v2x_emulator_init.h>
```

Data Fields

- atlk_thread_sched_t sched_params

  Thread scheduling parameters.
- imq_address_t imq_address

  IMQ address.

6.102.1 Detailed Description

V2X emulator configuration descriptor.

Examples:

craton-threadx/v2x-emulator/v2x-emulator-over-udp-example.c, and craton-threadx/v2x-emulator/v2x-service-user.c.

The documentation for this struct was generated from the following file:

- craton/v2x_emulator_init.h

6.103 v2x_netif_profile_t Struct Reference

Network interface V2X access profile.

```
#include <atlk/v2x.h>
```

Data Fields

- v2x_if_index_t if_index

  V2X physical interface index to attach to.
- v2x_channel_id_t channel_id

  Radio channel identifier to use.
- v2x_datarate_t datarate

  Transmission data rate for outgoing packets.
- v2x_power_dbm8_t power_dbm8

  Transmission power level for outgoing packets.

6.103.1 Detailed Description

Network interface V2X access profile.

See Also

IEEE Std 1609.4-2016 MLMEX-REGISTERTXPROFILE.request and MLMEX-DELETETXPROFILE.request.

Examples:

craton-threadx/wave-ipv6/wave-ipv6-client-example.c, craton-threadx/wave-ipv6/wave-ipv6-example.c, and craton-threadx/wave-ipv6/wave-ipv6-server-example.c.

6.103.2 Field Documentation

v2x_channel_id_t v2x_netif_profile_t::channel_id Radio channel identifier to use.

If `channel_id.op_class` is equal to V2X_OP_CLASS_NA, then the network interface will use not just one specific channel, but any channel accessed by the V2X physical interface.

v2x_if_index_t v2x_netif_profile_t::if_index V2X physical interface index to attach to.

If equal to V2X_IF_INDEX_NA then the network interface is detached from any V2X physical interface it was attached to previously.

Examples:

craton-threadx/wave-ipv6/wave-ipv6-client-example.c, craton-threadx/wave-ipv6/wave-ipv6-example.c, and craton-threadx/wave-ipv6/wave-ipv6-server-example.c.

The documentation for this struct was generated from the following file:

- atlk/v2x.h

6.104 v2x_protocol_t Struct Reference

V2X protocol descriptor.

```
#include <atlk/v2x.h>
```

Data Fields

- v2x_frame_type_t frame_type

  V2X MAC frame type.
- uint64_t protocol_id

  Protocol identifier.

6.104.1 Detailed Description

V2X protocol descriptor.

For example:

```
static const v2x_protocol_t wsmp_protocol = {
  .frame_type = V2X_FRAME_TYPE_DATA,
  .protocol_id = 0x88dc
};
```

Examples:

craton-threadx/v2x-emulator/v2x-emulator-over-udp-example.c.

6.104.2 Field Documentation

uint64_t v2x_protocol_t::protocol_id Protocol identifier.

If `frame_type` is v2x_frame_type_t::V2X_FRAME_TYPE_DATA then this is the 5-octet SNAP protocol identifer.

If `frame_type` is v2x_frame_type_t::V2X_FRAME_TYPE_VSA then this is the organizational identifier (OUI-36) and 4 least-significant bits specified by the identified organization.

Examples:

craton-threadx/bridge/v2x-udp-bridge-example.c, craton-threadx/dot4/dot4-channel-switching-example.c, craton-threadx/v2x-emulator/v2x-service-user.c, craton-threadx/v2x/v2x-example.c, craton-threadx/wlan-driver/traffic-monitor-example.c, and remote-posix/v2x/v2x-example.c.

The documentation for this struct was generated from the following file:

- atlk/v2x.h

6.105 v2x_receive_params_t Struct Reference

V2X receive parameters.

`#include <atlk/v2x_service.h>`

Data Fields

- eui48_t source_address

  Source MAC address.
- eui48_t dest_address

  Destination MAC address.
- v2x_user_priority_t user_priority

  MAC User Priority.
- v2x_channel_id_t channel_id

  Radio channel on which the frame was received.
- v2x_datarate_t datarate

  Data rate.
- v2x_power_dbm8_t power_dbm8

  Average input power of frame in units of 1/8 dBm.
- uint64_t receive_time_us

  Receive time in microseconds.

6.105.1 Detailed Description

V2X receive parameters.

Examples:

craton-threadx/bridge/v2x-udp-bridge-example.c, craton-threadx/v2x-emulator/v2x-emulator-over-udp-example.c, craton-threadx/v2x-emulator/v2x-service-user.c, craton-threadx/v2x/v2x-example.c, craton-threadx/wlan-driver/traffic-monitor-example.c, and remote-posix/v2x/v2x-example.c.

6.105.2 Field Documentation

v2x_datarate_t v2x_receive_params_t::datarate Data rate.

Examples:

craton-threadx/v2x-emulator/v2x-emulator-over-udp-example.c.

uint64_t v2x_receive_params_t::receive_time_us Receive time in microseconds.
Format: number of TAI microseconds since 2004-01-01T00:00:00Z (UTC).

v2x_user_priority_t v2x_receive_params_t::user_priority MAC User Priority.
The documentation for this struct was generated from the following file:

- atlk/v2x_service.h

6.106 v2x_sample_subscriber_config_t Struct Reference

V2X sample subscriber configuration.

```
#include <atlk/v2x_service.h>
```

Data Fields

- v2x_if_index_t if_index

  Ingress/egress physical interface index.
- v2x_sample_type_t type

  Subscription sample type.

6.106.1 Detailed Description

V2X sample subscriber configuration.
The documentation for this struct was generated from the following file:

- atlk/v2x_service.h

6.107 v2x_send_params_t Struct Reference

V2X send parameters.

```
#include <atlk/v2x_service.h>
```

Data Fields

- eui48_t source_address

  Source MAC address.
- eui48_t dest_address

  Destination MAC address.
- v2x_user_priority_t user_priority

  MAC User Priority.
- v2x_channel_id_t channel_id

Radio channel on which the frame should be transmitted.
- v2x_datarate_t datarate

  Transmission data rate.
- v2x_power_dbm8_t power_dbm8

  Transmission power level in units of 1/8 dBm.
- v2x_expiry_time_ms_t expiry_time_ms

  Expiration time in milliseconds.

6.107.1 Detailed Description

V2X send parameters.

Examples:

craton-threadx/bridge/v2x-udp-bridge-example.c, craton-threadx/dot4/dot4-channel-switching-example.c, craton-threadx/v2x-emulator/v2x-emulator-over-udp-example.c, craton-threadx/v2x-emulator/v2x-service-user.c, craton-threadx/v2x/v2x-example.c, craton-threadx/wlan-driver/traffic-monitor-example.c, and remote-posix/v2x/v2x-example.c.

6.107.2 Field Documentation

v2x_expiry_time_ms_t v2x_send_params_t::expiry_time_ms Expiration time in milliseconds.
Given v2x_send is called at time t0, frames which were not transmitted until t0 + expiry_time_ms will be dropped.

Note

Only supported when non-blocking wait option is used.

eui48_t v2x_send_params_t::source_address Source MAC address.
If not set to all zeros (EUI48_ZERO_INIT) and wlanTxSaOverrideEnabled is set to "true", the value of this field will determine the MAC frame's source address. Otherwise it has no effect.

v2x_user_priority_t v2x_send_params_t::user_priority MAC User Priority.
The documentation for this struct was generated from the following file:

- atlk/v2x_service.h

6.108 v2x_socket_config_t Struct Reference

V2X socket configuration.
```
#include <atlk/v2x_service.h>
```

Data Fields

- v2x_if_index_t if_index

  Ingress/egress physical interface index.
- v2x_protocol_t protocol

  V2X protocol descriptor.

6.108.1 Detailed Description

V2X socket configuration.

Examples:

craton-threadx/bridge/v2x-udp-bridge-example.c, craton-threadx/dot4/dot4-channel-switching-example.c, craton-threadx/v2x-emulator/v2x-service-user.c, craton-threadx/v2x/v2x-example.c, craton-threadx/wlan-driver/traffic-monitor-example.c, and remote-posix/v2x/v2x-example.c.

The documentation for this struct was generated from the following file:

- atlk/v2x_service.h

6.109 vca_connection_ops_t Struct Reference

The documentation for this struct was generated from the following file:

- atlk/vca.h

6.110 vca_connection_t Struct Reference

The documentation for this struct was generated from the following file:

- atlk/vca.h

6.111 vca_srv_config_t Struct Reference

VCA service configuration descriptor.

```
#include <atlk/vca.h>
```

6.111.1 Detailed Description

VCA service configuration descriptor.

The documentation for this struct was generated from the following file:

- atlk/vca.h

6.112 wd_arc_config_t Struct Reference

WD configuration for ARC.

```
#include <craton/wd.h>
```

Data Fields

- int wd_enabled

  Enable WD on ARC.
- wd_arc_exception_handler_t handler

  WD expiration callback on ARC.

6.112.1 Detailed Description

WD configuration for ARC.

6.112.2 Field Documentation

wd_arc_exception_handler_t wd_arc_config_t::handler WD expiration callback on ARC.

Default handler is used when set to NULL and WD is enabled.

The documentation for this struct was generated from the following file:

- craton/wd.h

6.113 wd_config_t Struct Reference

WD configuration for ARM.

```
#include <craton/wd.h>
```

Data Fields

- wd_mode_t wd_mode

  WD expiration mode.
- wd_arm_exception_handler_t handler

  WD expiration callback on ARM.
- wd_arc_config_t arc_configs [2]

  ARC cores WD configuration.

6.113.1 Detailed Description

WD configuration for ARM.

6.113.2 Field Documentation

wd_arm_exception_handler_t wd_config_t::handler WD expiration callback on ARM.

Default handler is used when set to NULL and WD is enabled.

The documentation for this struct was generated from the following file:

- craton/wd.h

6.114 wlan_frame_t Struct Reference

WLAN frame.

```
#include <craton/wlan_driver.h>
```

Data Fields

- const void ∗ frame_header_ptr

  Pointer to frame header.
- size_t frame_header_size

  Frame header size.
- const void ∗ frame_body_ptr

  Frame body pointer.
- size_t frame_body_size

  Frame body size.

6.114.1 Detailed Description

WLAN frame.

Examples:

craton-threadx/wlan-driver/traffic-monitor-example.c.

6.114.2 Field Documentation

const void∗ wlan_frame_t::frame_body_ptr Frame body pointer.

Remarks

In TX mode, Sequence Control field in MAC header is calculated by HW upon frame transmission, hence Sequence Control field does not reflect true value in TX mode.

In TX mode, last 4 bytes are reserved for FCS calculation which is done by HW upon frame transmission. MAC level does not see real FCS calculation, hence last 4 bytes do not reflect true FCS in TX mode.

The documentation for this struct was generated from the following file:

- craton/wlan_driver.h

6.115 wlan_rx_frame_info_t Struct Reference

WLAN RX frame info.

```
#include <craton/wlan_driver.h>
```

Data Fields

- uint8_t device_id

  WLAN device ID on which frame was received.
- v2x_datarate_t datarate

  Data rate in units of 500 kbit/s.
- v2x_power_dbm8_t power_dbm8

  Average input power of frame in units of 1/8 dBm.
- uint64_t rx_time_us

  Receive time in microseconds.
- uint64_t rx_isr_time_us

  Time at RX complete interrupt in microseconds.

6.115.1 Detailed Description

WLAN RX frame info.

Examples:

craton-threadx/wlan-driver/traffic-monitor-example.c.

6.115.2 Field Documentation

uint64_t wlan_rx_frame_info_t::rx_isr_time_us Time at RX complete interrupt in microseconds.
Format: number of TAI microseconds since 2004-01-01T00:00:00Z (UTC).

Examples:

craton-threadx/wlan-driver/traffic-monitor-example.c.

uint64_t wlan_rx_frame_info_t::rx_time_us Receive time in microseconds.
Format: number of TAI microseconds since 2004-01-01T00:00:00Z (UTC).

Examples:

craton-threadx/wlan-driver/traffic-monitor-example.c.

The documentation for this struct was generated from the following file:

- craton/wlan_driver.h

6.116 wlan_tx_frame_info_t Struct Reference

WLAN TX frame info.

```
#include <craton/wlan_driver.h>
```

Data Fields

- uint8_t device_id

  WLAN device ID on which frame was transmitted.
- v2x_datarate_t datarate

  Data rate in units of 500 kbit/s.
- v2x_power_dbm8_t power_dbm8

  Transmission power level in units of 1/8 dBm.
- uint64_t tx_queue_time_us

  Time when frame was added to transmit queue in microseconds.
- uint64_t tx_isr_time_us

  Time at TX complete interrupt in microseconds.

6.116.1 Detailed Description

WLAN TX frame info.

Examples:

craton-threadx/wlan-driver/traffic-monitor-example.c.

6.116.2 Field Documentation

uint64_t wlan_tx_frame_info_t::tx_isr_time_us Time at TX complete interrupt in microseconds.
Format: number of TAI microseconds since 2004-01-01T00:00:00Z (UTC).

Examples:

craton-threadx/wlan-driver/traffic-monitor-example.c.

uint64_t wlan_tx_frame_info_t::tx_queue_time_us Time when frame was added to transmit queue in microseconds.
Format: number of TAI microseconds since 2004-01-01T00:00:00Z (UTC).

Examples:

craton-threadx/wlan-driver/traffic-monitor-example.c.

The documentation for this struct was generated from the following file:

- craton/wlan_driver.h

7 File Documentation

7.1 atlk/aes.h File Reference

AES API.

```
#include <atlk/sdk.h>
```

Data Structures

- struct aes_key_t

  AES secret key.
- struct aes_ccm_nonce_t

  AES-CCM nonce.
- struct aes_ccm_authentication_tag_t

  AES-CCM authentication tag.
- struct aes_cbc_iv_t

  AES-CBC initialization vector.
- struct aes_cmac_tag_t

  AES-CMAC authentication tag.

Macros

- #define AES_KEY_SIZE 16

  AES secret key size in octets.
- #define AES_KEY_INIT { .value = { 0 } }

  AES secret key default initializer.
- #define AES_CCM_NONCE_SIZE 12

  AES-CCM nonce size in octets as specified in IEEE 1609.2-2016 clause 5.3.8.
- #define AES_CCM_NONCE_INIT { .value = { 0 } }

  AES-CCM nonce default initializer.

- #define AES_CCM_AUTHENTICATION_TAG_SIZE 16

  AES-CCM authentication tag size in octets as specified in IEEE 1609.2-2016 clause 5.3.8.
- #define AES_CCM_AUTHENTICATION_TAG_INIT { .value = { 0 } }

  AES-CCM authentication tag default initializer.
- #define AES_BLOCK_SIZE 16

  AES block size in octets.
- #define AES_CMAC_TAG_SIZE 16

  AES-CMAC authentication tag size in octets.
- #define AES_CMAC_TAG_INIT { .value = { 0 } }

  AES-CMAC authentication tag default initializer.

Functions

- atlk_rc_t aes_ccm_encrypt (const aes_key_t ∗key, const aes_ccm_nonce_t ∗nonce, const void ∗plaintext, size_t plaintext_size, void ∗ciphertext, size_t ∗ciphertext_size, aes_ccm_authentication_tag_t ∗tag)

  Encrypt with AES-CCM.
- atlk_rc_t aes_ccm_decrypt (const aes_key_t ∗key, const aes_ccm_nonce_t ∗nonce, const void ∗ciphertext, size_t ciphertext_size, void ∗plaintext, size_t ∗plaintext_size, aes_ccm_authentication_tag_t ∗tag)

  Decrypt with AES-CCM.
- atlk_rc_t aes_ccm_encrypt_2 (const aes_key_t ∗key, const void ∗nonce, size_t nonce_size, const void ∗header, size_t header_size, const void ∗plaintext, size_t plaintext_size, void ∗ciphertext, size_t ∗ciphertext_size, void ∗tag, size_t ∗tag_size)

  Encrypt with AES-CCM.
- atlk_rc_t aes_ccm_decrypt_2 (const aes_key_t ∗key, const void ∗nonce, size_t nonce_size, const void ∗header, size_t header_size, const void ∗ciphertext, size_t ciphertext_size, void ∗plaintext, size_t ∗plaintext_size, void ∗tag, size_t ∗tag_size)

  Decrypt with AES-CCM.
- atlk_rc_t aes_ecb_encrypt (const aes_key_t ∗key, const void ∗plaintext, size_t plaintext_size, void ∗ciphertext, size_t ∗ciphertext_size)

  Encrypt with AES-ECB.
- atlk_rc_t aes_ecb_decrypt (const aes_key_t ∗key, const void ∗ciphertext, size_t ciphertext_size, void ∗plaintext, size_t ∗plaintext_size)

  Decrypt with AES-ECB.
- atlk_rc_t aes_cbc_encrypt (const aes_key_t ∗key, const aes_cbc_iv_t ∗iv, const void ∗plaintext, size_t plaintext_size, void ∗ciphertext, size_t ∗ciphertext_size)

  Encrypt with AES-CBC.
- atlk_rc_t aes_cbc_decrypt (const aes_key_t ∗key, const aes_cbc_iv_t ∗iv, const void ∗ciphertext, size_t ciphertext_size, void ∗plaintext, size_t ∗plaintext_size)

  Decrypt with AES-CBC.
- atlk_rc_t aes_cmac_compute (const aes_key_t ∗key, const void ∗message, size_t message_size, aes_cmac_tag_t ∗tag)

  Compute AES-CMAC authentication tag.
- atlk_rc_t aes_ccmptmac_compute (const aes_key_t ∗key, const void ∗nonce, size_t nonce_size, const void ∗header, size_t header_size, const void ∗message, size_t message_size, void ∗tag, size_t ∗tag_size)

  Compute AES CCM Plain-Text MAC.

7.1.1 Detailed Description

AES API. Provides AES-CCM, AES-CBC, AES-ECB and AES-CMAC APIs.

References:

- IEEE 1609.2-2016: IEEE Standard for Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages
- NIST Special Publication 800-38A: Recommendation for Block Cipher Modes of Operation: Methods and Techniques

- NIST Special Publication 800-38B: Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication
- NIST Special Publication 800-38C: Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality

7.1.2 Function Documentation

atlk_rc_t aes_cbc_decrypt (const aes_key_t ∗ *key*, const aes_cbc_iv_t ∗ *iv*, const void ∗ *ciphertext*, size_t *ciphertext_size*, void ∗ *plaintext*, size_t ∗ *plaintext_size*) Decrypt with AES-CBC.

It is allowed that `plaintext` be equal to `ciphertext` in order to decrypt data in-place. Any other overlapping input and output buffers would result in undefined behavior.

`ciphertext_size` must be a multiple of AES_BLOCK_SIZE.

Parameters

`in`	*key*	Decryption key
`in`	*iv*	Initialization vector
`in`	*ciphertext*	Ciphertext to decrypt
`in`	*ciphertext_size*	Size of the ciphertext in octets
`out`	*plaintext*	Plaintext
`in,out`	*plaintext_size*	The maximum size (in) and resulting size (out) of the plaintext in octets

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/aes-example.c, and remote-posix/crypto/aes-example.c.

atlk_rc_t aes_cbc_encrypt (const aes_key_t ∗ *key*, const aes_cbc_iv_t ∗ *iv*, const void ∗ *plaintext*, size_t *plaintext_size*, void ∗ *ciphertext*, size_t ∗ *ciphertext_size*) Encrypt with AES-CBC.

It is allowed that `plaintext` be equal to `ciphertext` in order to encrypt data in-place. Any other overlapping input and output buffers would result in undefined behavior.

`plaintext_size` must be a multiple of AES_BLOCK_SIZE.

Remarks

According to NIST SP 800-38A clause 5.3: "The IV need not be secret; however, for the CBC and CFB modes, the IV for any particular execution of the encryption process must be unpredictable."
According to NIST SP 800-38A Appendix C: "There are two recommended methods for generating unpredictable IVs. The first method is to apply the forward cipher function, under the same key that is used for the encryption of the plaintext, to a nonce. The nonce must be a data block that is unique to each execution of the encryption operation. For example, the nonce may be a counter, as described in Appendix B, or a message number. The second method is to generate a random data block using a FIPS approved random number generator."

Parameters

`in`	*key*	Encryption key
`in`	*iv*	Initialization vector
`in`	*plaintext*	Plaintext to encrypt
`in`	*plaintext_size*	Size of the plaintext in octets

out	*ciphertext*	Ciphertext
in,out	*ciphertext_size*	The maximum size (in) and resulting size (out) of the ciphertext in octets

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/aes-example.c, and remote-posix/crypto/aes-example.c.

atlk_rc_t aes_ccm_decrypt (const aes_key_t ∗ *key,* const aes_ccm_nonce_t ∗ *nonce,* const void ∗ *ciphertext,* size_t *ciphertext_size,* void ∗ *plaintext,* size_t ∗ *plaintext_size,* aes_ccm_authentication_tag_t ∗ *tag*) Decrypt with AES-CCM.

Parameters were chosen as specified in IEEE 1609.2-2016 clause 5.3.8:

- No session header (associated data).
- The message authentication tag length is 16 octets.
- The nonce length is 12 octets.

It is allowed that `plaintext` be equal to `ciphertext` in order to decrypt data in-place. Any other overlapping input and output buffers would result in undefined behavior.

User should compare the resulting authentication tag to the expected authentication tag in order to verify message integrity. In the case that tags don't match this function would still return ATLK_OK.

Parameters

in	*key*	Decryption key
in	*nonce*	Session nonce
in	*ciphertext*	Ciphertext to decrypt
in	*ciphertext_size*	Size of the ciphertext in octets
out	*plaintext*	Plaintext
in,out	*plaintext_size*	The maximum size (in) and resulting size (out) of the plaintext in octets
out	*tag*	Authentication tag

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecies-example.c, and remote-posix/crypto/ecies-example.c.

atlk_rc_t aes_ccm_decrypt_2 (const aes_key_t ∗ *key,* const void ∗ *nonce,* size_t *nonce_size,* const void ∗ *header,* size_t *header_size,* const void ∗ *ciphertext,* size_t *ciphertext_size,* void ∗ *plaintext,* size_t ∗ *plaintext_size,* void ∗ *tag,* size_t ∗ *tag_size*)
Decrypt with AES-CCM.

It is allowed that `plaintext` be equal to `ciphertext` in order to decrypt data in-place. Any other overlapping input and output buffers would result in undefined behavior.

User should compare the resulting authentication tag to the expected authentication tag in order to verify message integrity. In the case that tags don't match this function would still return ATLK_OK.

Parameters

in	*key*	Decryption key
in	*nonce*	Session nonce (must be used once!)
in	*nonce_size*	Session nonce size in octets
in	*header*	Session header (optional)
in	*header_size*	Size of the session header in octets
in	*ciphertext*	Ciphertext to decrypt
in	*ciphertext_size*	Size of the ciphertext in octets
out	*plaintext*	Plaintext
in,out	*plaintext_size*	The maximum size (in) and resulting size (out) of the plaintext in octets
out	*tag*	Authentication tag
in,out	*tag_size*	The maximum size (in) and resulting size (out) of the authentication tag in octets

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t aes_ccm_encrypt (const aes_key_t ∗ *key*, const aes_ccm_nonce_t ∗ *nonce*, const void ∗ *plaintext*, size_t *plaintext_size*, void ∗ *ciphertext*, size_t ∗ *ciphertext_size*, aes_ccm_authentication_tag_t ∗ *tag*) Encrypt with AES-CCM.

Parameters were chosen as specified in IEEE 1609.2-2016 clause 5.3.8:

- No session header (associated data).
- The message authentication tag length is 16 octets.
- The nonce length is 12 octets.

It is allowed that `plaintext` be equal to `ciphertext` in order to encrypt data in-place. Any other overlapping input and output buffers would result in undefined behavior.

Parameters

in	*key*	Encryption key
in	*nonce*	Session nonce (must be used once!)
in	*plaintext*	Plaintext to encrypt
in	*plaintext_size*	Size of the plaintext in octets
out	*ciphertext*	Ciphertext
in,out	*ciphertext_size*	The maximum size (in) and resulting size (out) of the ciphertext in octets
out	*tag*	Authentication tag

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecies-example.c, and remote-posix/crypto/ecies-example.c.

atlk_rc_t aes_ccm_encrypt_2 (const aes_key_t ∗ *key*, const void ∗ *nonce*, size_t *nonce_size*, const void ∗ *header*, size_t *header_size*, const void ∗ *plaintext*, size_t *plaintext_size*, void ∗ *ciphertext*, size_t ∗ *ciphertext_size*, void ∗ *tag*, size_t ∗ *tag_size*)

Encrypt with AES-CCM.

It is allowed that `plaintext` be equal to `ciphertext` in order to encrypt data in-place. Any other overlapping input and output buffers would result in undefined behavior.

Parameters

in	*key*	Encryption key
in	*nonce*	Session nonce (must be used once!)
in	*nonce_size*	Session nonce size in octets
in	*header*	Session header (optional)
in	*header_size*	Size of the session header in octets
in	*plaintext*	Plaintext to encrypt
in	*plaintext_size*	Size of the plaintext in octets
out	*ciphertext*	Ciphertext
in,out	*ciphertext_size*	The maximum size (in) and resulting size (out) of the ciphertext in octets
out	*tag*	Authentication tag
in,out	*tag_size*	The maximum size (in) and resulting size (out) of the authentication tag in octets

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t aes_ccmptmac_compute (const aes_key_t ∗ *key*, const void ∗ *nonce*, size_t *nonce_size*, const void ∗ *header*, size_t *header_size*, const void ∗ *message*, size_t *message_size*, void ∗ *tag*, size_t ∗ *tag_size*) Compute AES CCM Plain-Text MAC.

Calculate an authentication tag which is the value T described in step 4 of section 6.1 in NIST SP 800-38C.

Parameters

in	*key*	Secret key
in	*nonce*	Session nonce (must be used once!)
in	*nonce_size*	Session nonce size in octets
in	*header*	Session header (optional)
in	*header_size*	Size of the session header in octets
in	*message*	Message to compute authentication tag
in	*message_size*	Size of the message in octets
out	*tag*	Authentication tag
in,out	*tag_size*	The maximum size (in) and resulting size (out) of the authentication tag in octets

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t aes_cmac_compute (const aes_key_t ∗ *key*, const void ∗ *message*, size_t *message_size*, aes_cmac_tag_t ∗ *tag*)
Compute AES-CMAC authentication tag.

Parameters

in	*key*	Secret key
in	*message*	Message to compute CMAC
in	*message_size*	Size of the message in octets
out	*tag*	AES-CMAC authentication tag

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/aes-example.c, and remote-posix/crypto/aes-example.c.

atlk_rc_t aes_ecb_decrypt (const aes_key_t ∗ *key*, const void ∗ *ciphertext*, size_t *ciphertext_size*, void ∗ *plaintext*, size_t ∗ *plaintext_size*) Decrypt with AES-ECB.

It is allowed that `plaintext` be equal to `ciphertext` in order to decrypt data in-place. Any other overlapping input and output buffers would result in undefined behavior.

`ciphertext_size` must be a multiple of AES_BLOCK_SIZE.

Parameters

`in`	*key*	Decryption key
`in`	*ciphertext*	Ciphertext to decrypt
`in`	*ciphertext_size*	Size of the ciphertext in octets
`out`	*plaintext*	Plaintext
`in,out`	*plaintext_size*	The maximum size (in) and resulting size (out) of the plaintext in octets

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t aes_ecb_encrypt (const aes_key_t ∗ *key*, const void ∗ *plaintext*, size_t *plaintext_size*, void ∗ *ciphertext*, size_t ∗ *ciphertext_size*) Encrypt with AES-ECB.

It is allowed that `plaintext` be equal to `ciphertext` in order to encrypt data in-place. Any other overlapping input and output buffers would result in undefined behavior.

`plaintext_size` must be a multiple of AES_BLOCK_SIZE.

Parameters

`in`	*key*	Encryption key
`in`	*plaintext*	Plaintext to encrypt
`in`	*plaintext_size*	Size of the plaintext in octets
`out`	*ciphertext*	Ciphertext
`in,out`	*ciphertext_size*	The maximum size (in) and resulting size (out) of the ciphertext in octets

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/aes-example.c.

7.2 atlk/can.h File Reference

CAN common definitions.

`#include <atlk/sdk.h>`

Macros

- #define CAN_DEVICE_ID_NA 0xffU

    Value indicating that CAN device ID is N/A.
- #define CAN_DATA_SIZE_MAX 8

    Maximum size of CAN message data in octets.
- #define CAN_ID_NUM_BITS_BASE 11

    CAN ID length for base frame format.
- #define CAN_ID_NUM_BITS_EXTENDED 29

    CAN ID length for extended frame format.
- #define CAN_ID_ERROR_BIT 29

    If this CAN ID bit is set, the frame is an error frame.
- #define CAN_ID_RTR_BIT 30

    If this CAN ID bit is set, the frame is a remote transmission request (RTR) frame.
- #define CAN_ID_EXTENDED_BIT 31

    If this CAN ID bit is set, the frame is an extended format frame.

Typedefs

- typedef uint8_t can_device_id_t

    CAN device ID (starts at zero)
- typedef uint32_t can_id_t

    CAN ID + flag bits.
- typedef struct can_service can_service_t

    CAN service instance.

7.2.1 Detailed Description

CAN common definitions.

7.3 atlk/can_device.h File Reference

CAN device API.

```
   #include <atlk/sdk.h>
#include <atlk/can.h>
```

Data Structures

- struct can_device_t

    CAN device.

Macros

- #define CAN_DEVICE_INIT

    CAN device default initializer.

Typedefs

- typedef atlk_rc_t(∗ can_tx_handler_t)(void ∗context, const void ∗data_ptr, size_t data_size, can_id_t can_id, const atlk_wait_t ∗wait)

    CAN device transmission function prototype.

Functions

- atlk_rc_t can_device_attach (can_service_t *service, can_device_id_t device_id, const can_device_t *device)

    Attach CAN device object to CAN device ID.
- atlk_rc_t can_rx_handler (can_service_t *service, can_device_id_t device_id, const void *data_ptr, size_t data_size, can_id_t can_id)

    Handle CAN frame reception.

7.3.1 Detailed Description

CAN device API.

7.3.2 Typedef Documentation

typedef atlk_rc_t(* can_tx_handler_t)(void *context, const void *data_ptr, size_t data_size, can_id_t can_id, const atlk_wait_t *wait) CAN device transmission function prototype.

Parameters

in	*context*	Context pointer
in	*data_ptr*	Pointer to start of data frame
in	*data_size*	Data frame size in bytes
in	*can_id*	CAN ID of frame
in	*wait*	Wait option (optional)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.3.3 Function Documentation

atlk_rc_t can_device_attach (can_service_t * *service*, can_device_id_t *device_id*, const can_device_t * *device*) Attach CAN device object to CAN device ID.

Parameters

in	*service*	CAN service
in	*device_id*	Device ID
in	*device*	CAN device

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t can_rx_handler (can_service_t * *service*, can_device_id_t *device_id*, const void * *data_ptr*, size_t *data_size*, can_id_t *can_id*) Handle CAN frame reception.

Parameters

in	*service*	CAN service
in	*device_id*	CAN device ID
in	*data_ptr*	Pointer to start of frame data

in	*data_size*	Frame data size in bytes
in	*can_id*	Frame's CAN ID

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.4 atlk/can_service.h File Reference

CAN service API.

```
   #include <atlk/sdk.h>
#include <atlk/can.h>
```

Data Structures

- struct can_id_filter_t

  CAN ID filter.
- struct can_socket_config_t

  CAN socket configuration parameters.

Macros

- #define CAN_ID_MASK_ALL_BITS 0xffffffffU

  CAN ID mask that masks all ID bits.
- #define CAN_ID_FILTER_ONE_ID(id)

  Initializer of CAN ID filter that filters one ID.
- #define CAN_SOCKET_CONFIG_INIT

  CAN socket configuration parameters default initializer.

Typedefs

- typedef struct can_socket can_socket_t

  CAN socket.

Functions

- atlk_rc_t can_default_service_get (can_service_t ∗∗service_ptr)

  Get pointer to default CAN service.
- atlk_rc_t can_service_delete (can_service_t ∗service)

  Delete CAN service instance.
- atlk_rc_t can_socket_create (can_service_t ∗service, can_socket_t ∗∗socket_ptr, const can_socket_config_t ∗config)

  Create CAN socket.
- atlk_rc_t can_socket_delete (can_socket_t ∗socket)

  Delete CAN socket.
- atlk_rc_t can_send (can_socket_t ∗socket, const void ∗data_ptr, size_t data_size, can_id_t can_id, const atlk_wait_t ∗wait)

  Send CAN frame.
- atlk_rc_t can_receive (can_socket_t ∗socket, void ∗data_ptr, size_t ∗data_size_ptr, can_id_t ∗can_id_ptr, const atlk_wait_t ∗wait)

  Receive CAN frame.

7.4.1 Detailed Description

CAN service API.

7.4.2 Function Documentation

atlk_rc_t can_default_service_get (can_service_t ∗∗ *service_ptr*) Get pointer to default CAN service.

Parameters

`out`	*service_ptr*	Pointer to CAN service

Note

New implementation of this getter will override default getter (declared as a weak symbol).

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/can/can-example.c, and craton-threadx/gnss-teseo/gnss-teseo-sou-example.c.

atlk_rc_t can_receive (can_socket_t ∗ *socket,* void ∗ *data_ptr,* size_t ∗ *data_size_ptr,* can_id_t ∗ *can_id_ptr,* const atlk_wait_t ∗ *wait*) Receive CAN frame.

See Also

Using wait option.

Parameters

`in`	*socket*	CAN socket
`out`	*data_ptr*	Pointer to start of frame data
`in,out`	*data_size_ptr*	Maximum (in) and actual (out) frame data size in bytes
`out`	*can_id_ptr*	Pointer to received frame's CAN ID
`in`	*wait*	Wait specification (optional)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/can/can-example.c, and craton-threadx/gnss-teseo/gnss-teseo-sou-example.c.

atlk_rc_t can_send (can_socket_t ∗ *socket,* const void ∗ *data_ptr,* size_t *data_size,* can_id_t *can_id,* const atlk_wait_t ∗ *wait*) Send CAN frame.

See Also

Using wait option.

Parameters

in	*socket*	CAN socket
in	*data_ptr*	Pointer to start of data
in	*data_size*	Size of data in bytes
in	*can_id*	CAN ID of frame
in	*wait*	Wait specification (optional)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/can/can-example.c.

atlk_rc_t can_service_delete (can_service_t ∗ *service*) Delete CAN service instance.

Parameters

in	*service*	CAN service instance

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/can/can-example.c, and craton-threadx/gnss-teseo/gnss-teseo-sou-example.c.

atlk_rc_t can_socket_create (can_service_t ∗ *service*, can_socket_t ∗∗ *socket_ptr*, const can_socket_config_t ∗ *config*)
Create CAN socket.

When can_socket_config_t::device_id is set to CAN_DEVICE_ID_NA, can_receive will receive frames from both CAN devices and can_send cannot be used.

Parameters

in	*service*	CAN service instance
out	*socket_ptr*	CAN socket pointer
in	*config*	CAN socket configuration

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/can/can-example.c, and craton-threadx/gnss-teseo/gnss-teseo-sou-example.c.

atlk_rc_t can_socket_delete (can_socket_t ∗ *socket*) Delete CAN socket.

Parameters

in	*socket*	CAN socket to delete

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/can/can-example.c, and craton-threadx/gnss-teseo/gnss-teseo-sou-example.c.

7.5 atlk/compiler.h File Reference

Compiler specific attributes, declarations and macros.

Macros

- #define atlk_must_check

  Attribute to mark functions whose return code must be checked.
- #define atlk_likely(x) (x)

  Compiler branch hint support.
- #define atlk_inline static

  Explicit function inlining support.
- #define atlk_format_printf(format_index, value_index)

  Format string checking support.
- #define atlk_no_return

  No-return function attribute.

7.5.1 Detailed Description

Compiler specific attributes, declarations and macros.

7.6 atlk/dhcp_client.h File Reference

DHCP client API.

```
   #include <atlk/sdk.h>
#include <atlk/os.h>
```

Data Structures

- struct dhcp_client_config_t

  DHCP client configuration parameters.

Macros

- #define DHCP_CLIENT_CONFIG_INIT

  DHCP client configuration parameters default initializer.

Typedefs

- typedef void(∗ dhcp_client_bound_handler_t)(void)

  DHCP client bound handler.

Functions

- atlk_rc_t dhcp_client_init (const dhcp_client_config_t *config)

  Initialize DHCP client.

7.6.1 Detailed Description

DHCP client API.

7.6.2 Typedef Documentation

typedef void(* dhcp_client_bound_handler_t)(void) DHCP client bound handler.
Handler is called when leasing address was established.

7.6.3 Function Documentation

atlk_rc_t dhcp_client_init (const dhcp_client_config_t * *config*) Initialize DHCP client.
Parameters

in	*config*	DHCP client configuration parameters

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.7 atlk/ecc.h File Reference

ECC API declarations.

```
#include <atlk/sdk.h>
```

Data Structures

- struct ecc_scalar_t

  Big integer type for use with ECC.
- struct ecc_point_t

  Point on an elliptic curve.
- struct ecc_signature_t

  ECDSA signature.
- struct ecc_fast_verification_signature_t

  ECDSA signature for fast verification.

Macros

- #define ECC_CURVE_P224_KEY_SIZE 28

  Size of ECC P-224 private key in octets.
- #define ECC_CURVE_P256_KEY_SIZE 32

  Size of ECC P-256 private key in octets.
- #define ECC_CURVE_P384_KEY_SIZE 48

  Size of ECC P-384 private key in octets.
- #define ECC_SCALAR_NUM_OF_UINT32 12

  Number of 32-bit words in ECC big integer.
- #define ECC_SCALAR_INIT { .value = { 0 } }

  ECC scalar default initializer.
- #define ECC_POINT_INIT

ECC point default initializer.

- #define ECC_SIGNATURE_INIT

  ECDSA signature default initializer.
- #define ECC_FAST_VERIFICATION_SIGNATURE_INIT

  ECDSA signature for fast verification default initializer.

Typedefs

- typedef struct ecc_service ecc_service_t

  ECC service.

Enumerations

- enum ecc_curve_t {
  ECC_CURVE_NIST_P224 = 0, ECC_CURVE_P224 = ECC_CURVE_NIST_P224, ECC_CURVE_NIST_P256 = 1, EC-C_CURVE_P256 = ECC_CURVE_NIST_P256,
  ECC_CURVE_BRAINPOOL_P256t1 = 2, ECC_CURVE_BRAINPOOL_P256r1 = 3, ECC_CURVE_NIST_P384 = 4, E-CC_CURVE_BRAINPOOL_P384t1 = 5,
  ECC_CURVE_BRAINPOOL_P384r1 = 6 }

  Supported elliptic curves.
- enum ecc_point_type_t { ECC_POINT_COMPRESSED_LSB_Y_0 = 2, ECC_POINT_COMPRESSED_LSB_Y_1 = 3, ECC_POINT_UNCOMPRESSED = 4 }

  Elliptic curve point representation type.
- enum ecc_rc_t {
  ECC_OK = 0, ECC_E_UNSPECIFIED = 1, ECC_E_VERIFY_FAILED = 2, ECC_E_SIGN_FAILED = 3,
  ECC_E_INVALID_POINT = 4, ECC_E_POINT_AT_INFINITY = 5, ECC_E_DECOMPRESSION_FAILED = 6 }

  ECC return code.

7.7.1 Detailed Description

ECC API declarations.

7.7.2 Enumeration Type Documentation

enum ecc_curve_t Supported elliptic curves.

Enumerator

ECC_CURVE_NIST_P224	NIST P-224.
ECC_CURVE_P224	NIST P-224 alias.
ECC_CURVE_NIST_P256	NIST P-256.
ECC_CURVE_P256	NIST P-256 alias.
ECC_CURVE_BRAINPOOL_P256t1	Brainpool P-256t1.
ECC_CURVE_BRAINPOOL_P256r1	Brainpool P-256r1.
ECC_CURVE_NIST_P384	NIST P-384.
ECC_CURVE_BRAINPOOL_P384t1	Brainpool P-384t1.
ECC_CURVE_BRAINPOOL_P384r1	Brainpool P-384r1.

enum ecc_point_type_t Elliptic curve point representation type.

Enumerator

ECC_POINT_COMPRESSED_LSB_Y_0	Compressed, LSB of Y coordinate is 0.
ECC_POINT_COMPRESSED_LSB_Y_1	Compressed, LSB of Y coordinate is 1.
ECC_POINT_UNCOMPRESSED	Uncompressed, Y coordinate is stored as-is.

enum ecc_rc_t ECC return code.

Enumerator

ECC_OK Operation successful.

ECC_E_UNSPECIFIED Unspecified error.

ECC_E_VERIFY_FAILED Signature verification failure.

ECC_E_SIGN_FAILED Signature generation failure ($r = 0$ or $s = 0$)

ECC_E_INVALID_POINT Point validation failure - point not on the curve.

ECC_E_POINT_AT_INFINITY Point at infinity.

ECC_E_DECOMPRESSION_FAILED Invalid modulus for point decompression.

7.8 atlk/ecc_math.h File Reference

ECC API functions.

```
   #include <atlk/sdk.h>
#include <atlk/ecc.h>
```

Functions

- int ecc_point_valid (ecc_curve_t curve, const ecc_point_t *point)

  Check if elliptic curve point is valid.
- atlk_rc_t ecc_point_compress (const ecc_point_t *uncompressed, ecc_point_t *compressed)

  Compress representation of point on an elliptic curve.
- atlk_rc_t ecc_point_decompress (ecc_curve_t curve, const ecc_point_t *compressed, ecc_point_t *decompressed)

  Decompress representation of point on an elliptic curve.
- atlk_rc_t ecc_point_multiply_add (ecc_curve_t curve, const ecc_point_t *P, const ecc_scalar_t *e, const ecc_point_t *R, ecc_point_t *Q)

  Perform a multiply-add on elliptic curve point.
- atlk_rc_t ecc_private_key_multiply_add (ecc_curve_t curve, const ecc_scalar_t *private_key, const ecc_scalar_t *addend, const ecc_scalar_t *multiplier, ecc_scalar_t *result)

  Perform a modular multiply-add a private key.

7.8.1 Detailed Description

ECC API functions.

7.8.2 Function Documentation

atlk_rc_t ecc_point_compress (const ecc_point_t * *uncompressed*, ecc_point_t * *compressed*) Compress representation of point on an elliptic curve.

If point is already compressed, copy it's X coordinate value as-is. It is allowed that `uncompressed` be equal to `compressed` in order to compress a point in-place.

Parameters

`in`	*uncompressed*	Elliptic curve point to compress
`out`	*compressed*	Compressed elliptic curve point

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t ecc_point_decompress (ecc_curve_t *curve*, const ecc_point_t ∗ *compressed*, ecc_point_t ∗ *decompressed*) Decompress representation of point on an elliptic curve.

If point is already uncompressed, copy it's value as-is. It is allowed that `compressed` be equal to `decompressed` in order to decompress a point in-place.

Parameters

in	*curve*	Elliptic curve associated with the points
in	*compressed*	Compressed elliptic curve point
out	*decompressed*	Decompressed elliptic curve point

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t ecc_point_multiply_add (ecc_curve_t *curve*, const ecc_point_t ∗ *P*, const ecc_scalar_t ∗ *e*, const ecc_point_t ∗ *R*, ecc_point_t ∗ *Q*) Perform a multiply-add on elliptic curve point.

Does the following operation: Q := R + (e ∗ P)

This operation can be used for public key extraction in ECQV PKI scheme.

Parameters

in	*curve*	Elliptic curve associated with the points
in	*P*	Elliptic curve point
in	*e*	Scalar for multiplication
in	*R*	Elliptic curve point for addition
out	*Q*	Calculated elliptic curve point

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

int ecc_point_valid (ecc_curve_t *curve*, const ecc_point_t ∗ *point*) Check if elliptic curve point is valid.

Parameters

in	*curve*	Elliptic curve associated with the point
in	*point*	Elliptic curve point to check

Return values

1	if `point` is valid on `curve`, 0 otherwise

atlk_rc_t ecc_private_key_multiply_add (ecc_curve_t *curve*, const ecc_scalar_t ∗ *private_key*, const ecc_scalar_t ∗ *addend*, const ecc_scalar_t ∗ *multiplier*, ecc_scalar_t ∗ *result*) Perform a modular multiply-add a private key.

Does the following operation: k' := b + (a ∗ k) mod n

Where: k is `private_key` k' is `result` b is `addend` a is `multiplier` n is order of the elliptic curve group that is specified by `curve`

This operation can be used to implement PKI schemes such as SCMS.

Remarks

`multiplier` is not allowed to be zero modulo the elliptic curve order (n).

Parameters

in	*curve*	Elliptic curve
in	*private_key*	Private key
in	*addend*	Scalar for addition
in	*multiplier*	Scalar for multiplication
out	*result*	Calculated private key

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.9 atlk/ecc_remote.h File Reference

ECC remote service API.

```
   #include <atlk/sdk.h>
#include <atlk/remote.h>
#include <atlk/ecc.h>
```

Typedefs

- typedef struct
  ecc_remote_service_config ecc_remote_service_config_t

  ECC remote service configuration parameters.

Functions

- atlk_rc_t ecc_remote_service_create (remote_transport_t ∗transport, const ecc_remote_service_config_t ∗config, ecc_service_t ∗∗service_ptr)

  Create ECC remote service.

7.9.1 Detailed Description

ECC remote service API.

7.9.2 Function Documentation

atlk_rc_t ecc_remote_service_create (remote_transport_t ∗ *transport*, const ecc_remote_service_config_t ∗ *config*, ecc_service_t ∗∗ *service_ptr*) Create ECC remote service.

Parameters

in	*transport*	Remote transport instance
in	*config*	ECC remote service configuration (optional)
out	*service_ptr*	ECC service

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

remote-posix/crypto/ecdsa-benchmark.c, remote-posix/crypto/ecdsa-example.c, and remote-posix/crypto/ecies-example.c.

7.10 atlk/ecc_service.h File Reference

ECC service API.

```
   #include <atlk/sdk.h>
#include <atlk/ecc.h>
#include <atlk/ecdsa.h>
#include <atlk/sha.h>
```

Data Structures

- struct ecc_request_context_t

  ECC request context.
- struct ecc_verify_params_t

  ECDSA verification parameters.
- struct ecc_sign_params_t

  ECDSA signing parameters.
- struct ecc_pma_params_t

  Elliptic curve point multiply-add parameters.
- struct ecc_request_t

  ECC request.
- struct ecc_response_t

  ECC response.

Macros

- #define ECC_REQUEST_INIT

  ECC request default initializer.
- #define ECC_RESPONSE_INIT

  ECC response default initializer.

Typedefs

- typedef uint32_t ecc_request_id_t

  ECC request ID.
- typedef struct ecc_socket ecc_socket_t

  ECC socket.

Enumerations

- enum ecc_request_type_t { ECC_REQUEST_TYPE_VERIFY = 0, ECC_REQUEST_TYPE_SIGN = 1, ECC_REQUEST_TYPE_PMA = 2 }

  ECC request type.

Functions

- atlk_rc_t ecc_default_service_get (ecc_service_t **service_ptr)

  Get pointer to default ECC service.
- atlk_rc_t ecc_service_delete (ecc_service_t *service)

  Delete ECC service.
- atlk_rc_t ecc_socket_create (ecc_service_t *service, ecc_socket_t **socket_ptr)

  Create ECC socket.
- atlk_rc_t ecc_socket_delete (ecc_socket_t *socket)

  Delete ECC socket.
- atlk_rc_t ecc_request_send (ecc_socket_t *socket, const ecc_request_t *request, const atlk_wait_t *wait)

Send ECC request.

- atlk_rc_t ecc_response_receive (ecc_socket_t *socket, ecc_response_t *response, const atlk_wait_t *wait)

  Receive ECC response.

7.10.1 Detailed Description

ECC service API.

7.10.2 Enumeration Type Documentation

enum ecc_request_type_t ECC request type.

Enumerator

ECC_REQUEST_TYPE_VERIFY	ECDSA verify.
ECC_REQUEST_TYPE_SIGN	ECDSA sign.
ECC_REQUEST_TYPE_PMA	Elliptic curve point multiply-add.

7.10.3 Function Documentation

atlk_rc_t ecc_default_service_get (ecc_service_t ** *service_ptr*) Get pointer to default ECC service.

Parameters

out	*service_ptr*	Pointer to ECC service

Note

Not supported by remote service library.

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecdsa-benchmark.c, craton-threadx/crypto/ecdsa-example.c, craton-threadx/crypto/ecies-example.c, and craton-threadx/crypto/secure-storage-example.c.

atlk_rc_t ecc_request_send (ecc_socket_t * *socket*, const ecc_request_t * *request*, const atlk_wait_t * *wait*) Send ECC request.

See Also

Using wait option.

Parameters

in	*socket*	ECC socket
in	*request*	ECC request
in	*wait*	Wait specification (optional)

See Also

::ecdsa_verify_digest

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecdsa-benchmark.c, craton-threadx/crypto/ecdsa-example.c, remote-posix/crypto/ecdsa-benchmark.c, and remote-posix/crypto/ecdsa-example.c.

atlk_rc_t ecc_response_receive (ecc_socket_t * *socket*, ecc_response_t * *response*, const atlk_wait_t * *wait*) Receive ECC response.

See Also

Using wait option.

Parameters

in	*socket*	ECC socket
out	*response*	ECC response
in	*wait*	Wait specification (optional)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecdsa-benchmark.c, craton-threadx/crypto/ecdsa-example.c, remote-posix/crypto/ecdsa-benchmark.c, and remote-posix/crypto/ecdsa-example.c.

atlk_rc_t ecc_service_delete (ecc_service_t * *service*) Delete ECC service.

Parameters

in	*service*	ECC service to delete

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecdsa-example.c, craton-threadx/crypto/secure-storage-example.c, remote-posix/crypto/ecdsa-benchmark.c, and remote-posix/crypto/ecdsa-example.c.

atlk_rc_t ecc_socket_create (ecc_service_t * *service*, ecc_socket_t ** *socket_ptr*) Create ECC socket.

Parameters

in	*service*	ECC service
out	*socket_ptr*	ECC socket

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecdsa-benchmark.c, craton-threadx/crypto/ecdsa-example.c, remote-posix/crypto/ecdsa-benchmark.c, and remote-posix/crypto/ecdsa-example.c.

atlk_rc_t ecc_socket_delete (ecc_socket_t * *socket*) Delete ECC socket.

Parameters

in	*socket*	ECC socket to delete

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecdsa-example.c, remote-posix/crypto/ecdsa-benchmark.c, and remote-posix/crypto/ecdsa-example.c.

7.11 atlk/ecdsa.h File Reference

ECDSA API.

```
   #include <atlk/sdk.h>
#include <atlk/ecc.h>
#include <atlk/sha.h>
```

Macros

- #define ECDSA_SIGNATURE_INIT ECC_SIGNATURE_INIT

  ECDSA signature default initializer alias.
- #define ECDSA_FAST_VERIFICATION_SIGNATURE_INIT ECC_FAST_VERIFICATION_SIGNATURE_INIT

  ECDSA signature for fast verification default initializer alias.

Typedefs

- typedef ecc_signature_t ecdsa_signature_t

  ECDSA signature alias.
- typedef
  ecc_fast_verification_signature_t ecdsa_fast_verification_signature_t

  ECDSA signature for fast verification alias.

Functions

- atlk_rc_t ecdsa_signature_convert (ecc_curve_t curve, const ecc_fast_verification_signature_t *fv_signature, ecc_signature_t *signature)

  Convert ECDSA signature for fast verification to a regular ECDSA signature.
- atlk_rc_t ecdsa_digest_sign (ecc_curve_t curve, const ecc_scalar_t *private_key, const sha_digest_t *digest, ecc_fast_verification_signature_t *fv_signature)

  Generate ECDSA fast verification signature.
- atlk_rc_t ecdsa_digest_verify (ecc_curve_t curve, const ecc_point_t *public_key, const sha_digest_t *digest, const ecc_signature_t *signature, ecc_rc_t *rc)

  Verify ECDSA signature.

7.11.1 Detailed Description

ECDSA API.

7.11.2 Function Documentation

atlk_rc_t ecdsa_digest_sign (ecc_curve_t *curve*, const ecc_scalar_t * *private_key*, const sha_digest_t * *digest*, ecc_fast_verification_signature_t * *fv_signature*) Generate ECDSA fast verification signature.

Parameters

in	*curve*	Elliptic curve associated with the private key
in	*private_key*	ECDSA private key to use for signing
in	*digest*	SHA digest to be signed
out	*fv_signature*	ECDSA signature for fast verification

Remarks

This function may be implemented using just software or accelerated using dedicated hardware, depending on library implementation.
This function expects to receive the private key in plain and thus is not suitable for a tamper-resistant implementation. Tamper-resistant implementations should use hsm_ecdsa_sign instead.

See Also

hsm_ecdsa_sign
ecc_request_send
ecc_response_receive

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t ecdsa_digest_verify (ecc_curve_t *curve*, const ecc_point_t * *public_key*, const sha_digest_t * *digest*, const ecc_signature_t * *signature*, ecc_rc_t * *rc*) Verify ECDSA signature.

Parameters

in	*curve*	Elliptic curve associated with the public key
in	*public_key*	ECDSA public key to use for verification

in	*digest*	SHA digest to be verified
in	*signature*	ECDSA signature to be verified
in	*rc*	ECC return code

Remarks

A return value of ATLK_OK only indicates that the verification completed with some result. The user must inspect ∗rc and compare it against ECC_OK in order to know whether the signature is correct or incorrect.
This function may be implemented using just software or accelerated using dedicated hardware, depending on library implementation.

See Also

ecc_request_send
ecc_response_receive

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t ecdsa_signature_convert (ecc_curve_t *curve,* const ecc_fast_verification_signature_t ∗ *fv_signature,* ecc_-signature_t ∗ *signature*) Convert ECDSA signature for fast verification to a regular ECDSA signature.

Parameters

in	*curve*	Elliptic curve associated with the signature
in	*fv_signature*	ECDSA signature for fast verification
out	*signature*	ECDSA signature

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecdsa-example.c, and remote-posix/crypto/ecdsa-example.c.

7.12 atlk/ecies.h File Reference

ECIES API.

```
  #include <atlk/sdk.h>
#include <atlk/ecc.h>
#include <atlk/sha.h>
```

Data Structures

- struct ecies_authentication_tag_t

  ECIES authentication tag.

Macros

- #define ECIES_MAX_TEXT_SIZE 16

  Maximum size of plaintext/ciphertext (in octets) for ECIES encryption/decryption.
- #define ECIES_AUTHENTICATION_TAG_SIZE 16

  ECIES authentication tag size in octets as specified in IEEE 1609.2-2016 clause 5.3.5.
- #define ECIES_AUTHENTICATION_TAG_INIT { .value = { 0 } }

  ECIES authentication tag default initializer.

Functions

- atlk_rc_t ecies_key_create (ecc_curve_t curve, const ecc_point_t *peer_public_key, ecc_point_t *public_key, void *key, size_t key_size, const void *kdf_param, size_t kdf_param_size)

  Generate ECIES shared secret key from a public key.
- atlk_rc_t ecies_encrypt (sha_algorithm_t sha_algorithm, const void *key, size_t key_size, const void *plaintext, size_t plaintext_size, void *ciphertext, size_t *ciphertext_size, ecies_authentication_tag_t *tag)

  Encrypt with ECIES.
- atlk_rc_t ecies_decrypt (sha_algorithm_t sha_algorithm, const void *key, size_t key_size, const void *ciphertext, size_t ciphertext_size, void *plaintext, size_t *plaintext_size, ecies_authentication_tag_t *tag)

  Decrypt with ECIES.

7.12.1 Detailed Description

ECIES API. Provides ECIES encryption/decryption API reflected from IEEE 1609.2-2016 (clause 5.3.5) and IEEE 1363a (clause 11.3).

References:

- IEEE 1609.2-2016: IEEE Standard for Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages.
- IEEE Std 1363a: IEEE Standard Specifications for Public-Key Cryptography - Amendment 1: Additional Techniques.

7.12.2 Function Documentation

atlk_rc_t ecies_decrypt (sha_algorithm_t *sha_algorithm*, const void * *key*, size_t *key_size*, const void * *ciphertext*, size_t *ciphertext_size*, void * *plaintext*, size_t * *plaintext_size*, ecies_authentication_tag_t * *tag*) Decrypt with ECIES.

Overlapping input and output buffers would result in undefined behavior.

User should compare the resulting authentication tag to the expected authentication tag in order to verify message integrity. In the case that tags don't match this function would still return ATLK_OK.

must not be greater than ECIES_MAX_TEXT_SIZE.

Parameters

in	*sha_algorithm*	SHA algorithm to be used in MAC calculation
in	*key*	ECIES key
in	*key_size*	ECIES key size in octets
in	*ciphertext*	Ciphertext to decrypt
in	*ciphertext_size*	Size of the ciphertext in octets
out	*plaintext*	Plaintext
in,out	*plaintext_size*	The maximum size (in) and resulting size (out) of the plaintext in octets
out	*tag*	Authentication tag

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecies-example.c, and remote-posix/crypto/ecies-example.c.

atlk_rc_t ecies_encrypt (sha_algorithm_t *sha_algorithm,* const void ∗ *key,* size_t *key_size,* const void ∗ *plaintext,* size_t *plaintext_size,* void ∗ *ciphertext,* size_t ∗ *ciphertext_size,* ecies_authentication_tag_t ∗ *tag*) Encrypt with ECIES.

Overlapping input and output buffers would result in undefined behavior.

must not be greater than ECIES_MAX_TEXT_SIZE.

Parameters

`in`	*sha_algorithm*	SHA algorithm to be used in MAC calculation
`in`	*key*	ECIES key
`in`	*key_size*	ECIES key size in octets
`in`	*plaintext*	Plaintext to encrypt
`in`	*plaintext_size*	Size of the plaintext in octets
`out`	*ciphertext*	Ciphertext
`in,out`	*ciphertext_size*	The maximum size (in) and resulting size (out) of the ciphertext in octets
`out`	*tag*	Authentication tag

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecies-example.c, and remote-posix/crypto/ecies-example.c.

atlk_rc_t ecies_key_create (ecc_curve_t *curve,* const ecc_point_t ∗ *peer_public_key,* ecc_point_t ∗ *public_key,* void ∗ *key,* size_t *key_size,* const void ∗ *kdf_param,* size_t *kdf_param_size*) Generate ECIES shared secret key from a public key.

`curve` must belong to an elliptic curve with more than 224 bits (e.g. NIST P-256).

`kdf_param` is an optional octet string used as a key derivation parameter. In order for the key derivation parameter to be the empty string, `kdf_param_size` should be 0. The key derivation parameter can be used to prevent misbinding attacks. Please refer to IEEE Std 1363a-2004 clause 11.3.2 where the key derivation parameter is denoted by P1.

Parameters

`in`	*curve*	Elliptic curve
`in`	*peer_public_key*	Public key of ECIES peer
`out`	*public_key*	Ephemeral public key to be sent to ECIES peer
`out`	*key*	Derived ECIES key
`in`	*key_size*	ECIES key size in octets
`in`	*kdf_param*	Key derivation parameter (optional)
`in`	*kdf_param_size*	Key derivation parameter size in octets

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecies-example.c, and remote-posix/crypto/ecies-example.c.

7.13 atlk/eui48.h File Reference

48-bit Extended Unique Identifier declarations and macros

```
#include <atlk/sdk.h>
```

Data Structures

- struct eui48_t

    48-bit Extended Unique Identifier

Macros

- #define EUI48_LEN 6

    Length of EUI-48 in octets.
- #define EUI48_ZERO_INIT { .octets = { 0 } }

    Initializer that represents an invalid MAC address.
- #define EUI48_BCAST_INIT

    Initializer that represents a broadcast MAC address.
- #define EUI48_INIT(_0, _1, _2, _3, _4, _5)

    Initializer that takes the EUI-48 octets as arguments.
- #define EUI48_FMT "%02x:%02x:%02x:%02x:%02x:%02x"

    Format string for EUI-48.

7.13.1 Detailed Description

48-bit Extended Unique Identifier declarations and macros

7.13.2 Macro Definition Documentation

#define EUI48_INIT(*_0,* *_1,* *_2,* *_3,* *_4,* *_5*) **Value:**

```
{ \
  .octets = { _0, _1, _2, _3, _4, _5 } }
```

Initializer that takes the EUI-48 octets as arguments.
Example:

```
eui48_t my_addr = EUI48_INIT(0x90, 0x56, 0x92, 0x0, 0x0, 0x1);
```

7.14 atlk/ftp_server.h File Reference

FTP server API.

```
#include <atlk/sdk.h>
```

Functions

- atlk_rc_t ftp_server_start (void)

    Start FTP server.
- atlk_rc_t ftp_server_stop (void)

    Stop FTP server.

7.14.1 Detailed Description

FTP server API.

7.14.2 Function Documentation

atlk_rc_t ftp_server_start (void) Start FTP server.
Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t ftp_server_stop (void) Stop FTP server.

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.15 atlk/gnss.h File Reference

GNSS API.

```
   #include <atlk/sdk.h>
#include <atlk/os.h>
#include <atlk/uart.h>
#include <atlk/nav.h>
```

Data Structures

- struct gnss_config_t

  GNSS configuration parameters.
- struct gnss_reset_params_t

  GNSS reset parameters.

Macros

- #define GNSS_CONFIG_INIT

  GNSS configuration parameters default initializer.
- #define GNSS_COLD_START_F_CLEAR_ALMANAC (1U << 0)

  Clear almanac during cold GNSS start.
- #define GNSS_COLD_START_F_CLEAR_EPHEMERIS (1U << 1)

  Clear ephemeris during cold GNSS start.
- #define GNSS_COLD_START_F_CLEAR_POSITION (1U << 2)

  Clear position during cold GNSS start.
- #define GNSS_COLD_START_F_CLEAR_TIME (1U << 3)

  Clear time during cold GNSS start.
- #define GNSS_COLD_START_F_CLEAR_ALL UINT32_MAX

  Clear all during cold reset.
- #define GNSS_RESET_PARAMS_INIT

  GNSS reset parameters default initializer.

Typedefs

- typedef atlk_rc_t(∗ gnss_hw_reset_t)(void)

  GNSS HW reset procedure.

Enumerations

- enum gnss_model_t {
  GNSS_MODEL_STMICRO_TESEO_II = 0, GNSS_MODEL_STMICRO_TESEO_III = 1, GNSS_MODEL_UBLOX_MAX_7 = 2, GNSS_MODEL_UBLOX_MAX_M8 = 3,
  GNSS_MODEL_NA = 255 }

  GNSS model.
- enum gnss_reset_type_t { GNSS_RESET_TYPE_HW = 0, GNSS_RESET_TYPE_SW = 1, GNSS_RESET_TYPE_GNSS_ONLY = 2, GNSS_RESET_TYPE_NA = 3 }

  GNSS reset type.
- enum gnss_start_type_t { GNSS_START_TYPE_HOT = 0, GNSS_START_TYPE_WARM = 1, GNSS_START_TYPE_COLD = 2, GNSS_START_TYPE_NA = 3 }

  GNSS start type.

Functions

- atlk_rc_t gnss_init (const gnss_config_t ∗config)

  Initialize GNSS.
- atlk_rc_t gnss_fw_version_get (char ∗fw_version, size_t ∗fw_version_size, const atlk_wait_t ∗wait)

  Get GNSS firmware version.
- atlk_rc_t gnss_reset (const gnss_reset_params_t ∗params, const atlk_wait_t ∗wait)

  GNSS reset.

Variables

- const atlk_wait_t gnss_default_wait

  Predefined GNSS default wait option.

7.15.1 Detailed Description

GNSS API.

7.15.2 Typedef Documentation

typedef atlk_rc_t(∗ gnss_hw_reset_t)(void) GNSS HW reset procedure.

Note

When using a Teseo device, this procedure should either toggle the reset pin or toggle the device's power supply. Performing Teseo firmware update procedure in recovery mode will likely fail without HW reset.

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.15.3 Enumeration Type Documentation

enum gnss_model_t GNSS model.

Enumerator

GNSS_MODEL_STMICRO_TESEO_II STMicroelectronics Teseo-II.

GNSS_MODEL_STMICRO_TESEO_III STMicroelectronics Teseo-III.

GNSS_MODEL_UBLOX_MAX_7 u-blox MAX-7 series

GNSS_MODEL_UBLOX_MAX_M8 u-blox MAX-M8 series

GNSS_MODEL_NA GNSS is not available.

enum gnss_reset_type_t GNSS reset type.

Enumerator

GNSS_RESET_TYPE_HW Reset hardware.

GNSS_RESET_TYPE_SW Reset software.

GNSS_RESET_TYPE_GNSS_ONLY Reset GNSS engine only.

GNSS_RESET_TYPE_NA Reset type is N/A.

enum gnss_start_type_t GNSS start type.

Enumerator

GNSS_START_TYPE_HOT Hot start.

GNSS_START_TYPE_WARM Warm start.

GNSS_START_TYPE_COLD Cold start.

GNSS_START_TYPE_NA Start type is N/A.

7.15.4 Function Documentation

atlk_rc_t gnss_fw_version_get (char ∗ *fw_version*, size_t ∗ *fw_version_size*, const atlk_wait_t ∗ *wait*) Get GNSS firmware version.

Parameters

`out`	*fw_version*	GNSS firmware version buffer
`in,out`	*fw_version_size*	Maximum size (in) and actual (out) in chars
`in`	*wait*	Wait specification (optional)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t gnss_init (const gnss_config_t ∗ *config*) Initialize GNSS.

Parameters

`in`	*config*	GNSS configuration parameters

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

remote-posix/gnss/gnss-example.c.

atlk_rc_t gnss_reset (const gnss_reset_params_t ∗ *params*, const atlk_wait_t ∗ *wait*) GNSS reset.

Parameters

in	*params*	GNSS reset parameters
in	*wait*	Wait specification (optional)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.15.5 Variable Documentation

const atlk_wait_t gnss_default_wait Predefined GNSS default wait option.
This is the default recommended amount of time to wait in all GNSS API functions which receive a wait option.

Waiting less than this amount of time might result in subsequent function call receiving a reply from GNSS which was not meant for it.

Waiting more than this amount of time is pointless.

7.16 atlk/gnss_teseo.h File Reference

GNSS Teseo API.

```
   #include <atlk/sdk.h>
#include <atlk/os.h>
#include <atlk/uart.h>
#include <atlk/sensor.h>
```

Data Structures

- struct gnss_teseo_fw_update_params_t

  Teseo firmware update parameters.
- struct gnss_teseo_sou_config_t

  Teseo SOU configuration parameters.

Macros

- #define GNSS_TESEO_FW_UPDATE_NMEA_SPEED_BPS_AUTO 0

  UART speed used to send FW Upgrade command is automatically chosen.
- #define GNSS_TESEO_FW_UPDATE_PARAMS_INIT

  Teseo firmware update parameters default initializer.
- #define GNSS_TESEO_SOU_CONFIG_INIT

  Teseo SOU configuration parameters default initializer.

Enumerations

- enum gnss_teseo_sou_operating_mode_t { GNSS_TESEO_SOU_OPERATING_MODE_20 = 0x14, GNSS_TESEO_SOU_OPERATING_MODE_NA = 0xff }

  Teseo SOU operating mode.

Functions

- atlk_rc_t gnss_teseo_fw_update (const gnss_teseo_fw_update_params_t ∗params)

  Update Teseo firmware.
- atlk_rc_t gnss_teseo_sou_init (const gnss_teseo_sou_config_t ∗config)

  Init Teseo SOU.
- atlk_rc_t gnss_teseo_sou_reverse_gear_data_feed (int value)

  Feed reverse gear data via SOU.

- atlk_rc_t gnss_teseo_sou_gyro_1axis_data_feed (sensor_value_t value)

  Feed gyro 1-axis data via SOU.
- atlk_rc_t gnss_teseo_sou_wheels_speed_data_feed (sensor_wheels_speed_t value)

  Feed wheels speed data via SOU.

7.16.1 Detailed Description

GNSS Teseo API.

7.16.2 Enumeration Type Documentation

enum gnss_teseo_sou_operating_mode_t Teseo SOU operating mode.

Chosen operating mode must be aligned with the operating mode configured in DR firmware. Which sensors to feed depends on chosen mode.

Please contact Autotalks support for further details.

Enumerator

GNSS_TESEO_SOU_OPERATING_MODE_20 CAN gyro, DWP and reverse from CAN bus are selected as DR inputs. The following sensor feeders shall be used with this mode: reverse gear status, gyro_1axis and wheels speed.

GNSS_TESEO_SOU_OPERATING_MODE_NA GNSS operating mode is not available.

7.16.3 Function Documentation

atlk_rc_t gnss_teseo_fw_update (const gnss_teseo_fw_update_params_t ∗ *params*) Update Teseo firmware.

Function call is blocking and may take several minutes to return. It is recommended to set gnss_teseo_fw_update_params_t.sched_params to the same scheduling parameters of the thread from which the function is called.

Parameters

in	*params*	Teseo firmware update parameters

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/gnss-teseo/gnss-teseo-fw-update-example.c.

atlk_rc_t gnss_teseo_sou_gyro_1axis_data_feed (sensor_value_t *value*) Feed gyro 1-axis data via SOU.

Parameters

in	*value*	Gyro 1-axis rate value

Return values

ATLK_OK	if succeeded
ATLK_E_OUT_OF_DOMAIN	if fed value is N/A

Returns

Error code if failed

Examples:

craton-threadx/gnss-teseo/gnss-teseo-sou-example.c.

atlk_rc_t gnss_teseo_sou_init (const gnss_teseo_sou_config_t * *config*) Init Teseo SOU.

Note

Teseo SOU requires a Teseo device flashed with DR firmware (whether DR firmware is flashed or not is not checked in code).

Parameters

in	*config*	Teseo SOU configuration parameters

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/gnss-teseo/gnss-teseo-sou-example.c.

atlk_rc_t gnss_teseo_sou_reverse_gear_data_feed (int *value*) Feed reverse gear data via SOU.

If `value` equals 0, reverse gear is not enabled.

Parameters

in	*value*	Reverse gear status value (boolean)

Return values

ATLK_OK	if succeeded
ATLK_E_OUT_OF_DOMAIN	if fed value is N/A

Returns

Error code if failed

Examples:

craton-threadx/gnss-teseo/gnss-teseo-sou-example.c.

atlk_rc_t gnss_teseo_sou_wheels_speed_data_feed (sensor_wheels_speed_t *value*) Feed wheels speed data via SOU.

Parameters

in	*value*	Wheels speed value

Return values

ATLK_OK	if succeeded
ATLK_E_OUT_OF_DOMAIN	if fed value is N/A

Returns

Error code if failed

Examples:

craton-threadx/gnss-teseo/gnss-teseo-sou-example.c.

7.17 atlk/hsm.h File Reference

HSM API declarations.

```
#include <atlk/sdk.h>
```

Data Structures

- struct hsm_cell_range_t
  HSM secure storage cell range.

Macros

- #define HSM_CELL_INDEX_NA 0xffffffffUL
  Value indicating that HSM cell index is N/A.
- #define HSM_CELL_RANGE_INIT
  HSM cell range default initializer.

Typedefs

- typedef uint32_t hsm_cell_index_t
  HSM secure storage cell index.
- typedef struct hsm_service hsm_service_t
  HSM service instance.

7.17.1 Detailed Description

HSM API declarations.

7.18 atlk/hsm_emulator.h File Reference

HSM emulator API.

```
  #include <atlk/sdk.h>
#include <atlk/aes.h>
#include <atlk/hsm.h>
#include <atlk/ecc_service.h>
```

Data Structures

- struct hsm_emulator_config_t
  HSM emulator configuration parameters.

Macros

- #define HSM_EMULATOR_CONFIG_INIT
  HSM emulator configuration default initializer.

Functions

- atlk_rc_t hsm_emulator_create (const hsm_emulator_config_t *config, hsm_service_t **service_ptr)
  Create HSM emulator service.

7.18.1 Detailed Description

HSM emulator API.

7.18.2 Function Documentation

atlk_rc_t hsm_emulator_create (const hsm_emulator_config_t * *config*, hsm_service_t ** *service_ptr*) Create HSM emulator service.

Parameters

in	*config*	HSM emulator configuration parameters
out	*service_ptr*	HSM emulator service

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecdsa-example.c, craton-threadx/crypto/ecies-example.c, craton-threadx/crypto/secure-storage-example.c, remote-posix/crypto/ecdsa-example.c, and remote-posix/crypto/ecies-example.c.

7.19 atlk/hsm_service.h File Reference

HSM service API.

```
  #include <atlk/sdk.h>
#include <atlk/ecc.h>
#include <atlk/ecdsa.h>
#include <atlk/hsm.h>
#include <atlk/sha.h>
#include <atlk/aes.h>
```

Data Structures

- struct hsm_capability_info_t

  HSM capability information.
- struct hsm_nvm_config_t

  HSM NVM configuration.
- struct hsm_ecc_private_key_info_t

  Private key information.
- struct hsm_csr_random_prefix_t

  CSR random prefix.

Macros

- #define HSM_CAPABILITY_INFO_INIT

  HSM capability info default initializer.
- #define HSM_NVM_CONFIG_INIT

  HSM NVM configuration default initializer.
- #define HSM_ECC_PRIVATE_KEY_INFO_INIT

  Private key information default initializer.
- #define HSM_CSR_RANDOM_PREFIX_SIZE 16

  CSR random prefix size in bytes.
- #define HSM_CSR_RANDOM_PREFIX_INIT { .value = { 0 } }

  CSR random prefix default initializer.

Enumerations

- enum hsm_private_key_type_t {
  HSM_PRIVATE_KEY_TYPE_ISOLATED = 0, HSM_PRIVATE_KEY_TYPE_CSR_MEMBER = 1, HSM_PRIVATE_KEY_TYPE_CSR_SIGNER = 2, HSM_PRIVATE_KEY_TYPE_MA_INPUT = 3,
  HSM_PRIVATE_KEY_TYPE_MA_OUTPUT = 4 }

  Private key type.
- enum hsm_public_key_algorithm_t { HSM_PUBLIC_KEY_ALGORITHM_ECDSA = 0, HSM_PUBLIC_KEY_ALGORITHM_ECIES = 1 }

  Public key algorithm.

Functions

- atlk_rc_t hsm_default_service_get (hsm_service_t **service_ptr)

  Get pointer to default HSM service.
- atlk_rc_t hsm_service_delete (hsm_service_t *service)

  Delete HSM service.
- atlk_rc_t hsm_capability_info_get (hsm_service_t *service, hsm_capability_info_t *capability_info)

  Get HSM capability information.
- atlk_rc_t hsm_nvm_init (hsm_service_t *service, const hsm_nvm_config_t *config)

  Initialize or re-initialize HSM NVM.
- atlk_rc_t hsm_ecc_private_key_import (hsm_service_t *service, hsm_cell_index_t private_key_index, const hsm_ecc_private_key_info_t *private_key_info, const ecc_scalar_t *private_key)

  Import ECC private key.
- atlk_rc_t hsm_ecc_private_key_create (hsm_service_t *service, hsm_cell_index_t private_key_index, const hsm_ecc_private_key_info_t *private_key_info)

  Create ECC private key.
- atlk_rc_t hsm_ecc_private_key_info_get (hsm_service_t *service, hsm_cell_index_t private_key_index, hsm_ecc_private_key_info_t *private_key_info)

  Get private key information.
- atlk_rc_t hsm_ecc_public_key_get (hsm_service_t *service, hsm_cell_index_t private_key_index, ecc_point_t *public_key)

  Return ECC public key that matches a stored private key.
- atlk_rc_t hsm_ecdsa_sign (hsm_service_t *service, hsm_cell_index_t private_key_index, const sha_digest_t *digest, ecc_fast_verification_signature_t *signature)

  Generate ECDSA signature from a given hash digest.
- atlk_rc_t hsm_ecies_key_derive (hsm_service_t *service, hsm_cell_index_t private_key_index, const ecc_point_t *peer_public_key, void *key, size_t key_size, const void *kdf_param, size_t kdf_param_size)

  Derive ECIES key from a private key and peer public key.
- atlk_rc_t hsm_ecc_private_key_multiply_add (hsm_service_t *service, hsm_cell_index_t input_key_index, hsm_cell_index_t output_key_index, const ecc_scalar_t *key_addend, const ecc_scalar_t *key_multiplier)

  Perform a modular multiply-add on stored private key and store the result.
- atlk_rc_t hsm_csr_ecdsa_external_sign (hsm_service_t *service, hsm_cell_index_t private_key_index, const sha_digest_t *digest, hsm_csr_random_prefix_t *prefix, ecc_fast_verification_signature_t *signature)

  ECDSA sign externally on CSR (Certificate Signing Request).
- atlk_rc_t hsm_csr_ecdsa_public_keys_sign (hsm_service_t *service, hsm_cell_index_t private_key_index, const hsm_cell_range_t *range_array_ptr, size_t range_array_size, hsm_csr_random_prefix_t *prefix, ecc_fast_verification_signature_t *signature)

  ECDSA sign internally on public keys.
- atlk_rc_t hsm_host_nvm_aes_cbc_encrypt (hsm_service_t *service, const void *plaintext, size_t plaintext_size, aes_cbc_iv_t *iv, void *ciphertext, size_t *ciphertext_size)

  Encrypt with AES-128-CBC data to be stored on a non-secure NVM.
- atlk_rc_t hsm_host_nvm_aes_cbc_decrypt (hsm_service_t *service, const aes_cbc_iv_t *iv, const void *ciphertext, size_t ciphertext_size, void *plaintext, size_t *plaintext_size)

  Decrypt with AES-128-CBC data that is stored on a non-secure NVM.

- atlk_rc_t hsm_host_nvm_aes_cmac_compute (hsm_service_t *service, const void *message, size_t message_size, aes_cmac_tag_t *tag)

  Compute AES-128-CMAC authentication tag of data stored on a non-secure NVM.

7.19.1 Detailed Description

HSM service API.

7.19.2 Enumeration Type Documentation

enum hsm_private_key_type_t Private key type.

Enumerator

HSM_PRIVATE_KEY_TYPE_ISOLATED	Private key that cannot interact with other private keys.
HSM_PRIVATE_KEY_TYPE_CSR_MEMBER	Private key whose public key conterpart can be a member of a CSR.
HSM_PRIVATE_KEY_TYPE_CSR_SIGNER	Private key that can be used to sign a CSR.
HSM_PRIVATE_KEY_TYPE_MA_INPUT	Private key that can be input to hsm_ecc_private_key_multiply_add.
HSM_PRIVATE_KEY_TYPE_MA_OUTPUT	Private key that is the output of hsm_ecc_private_key_multiply_add.

enum hsm_public_key_algorithm_t Public key algorithm.

Enumerator

HSM_PUBLIC_KEY_ALGORITHM_ECDSA	Public key for ECDSA.
HSM_PUBLIC_KEY_ALGORITHM_ECIES	Public key for ECIES.

7.19.3 Function Documentation

atlk_rc_t hsm_capability_info_get (hsm_service_t * *service*, hsm_capability_info_t * *capability_info*) Get HSM capability information.

Parameters

in	*service*	HSM service instance
out	*capability_info*	HSM capability information

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecdsa-example.c, craton-threadx/crypto/ecies-example.c, remote-posix/crypto/ecdsa-example.c, and remote-posix/crypto/ecies-example.c.

atlk_rc_t hsm_csr_ecdsa_external_sign (hsm_service_t * *service*, hsm_cell_index_t *private_key_index*, const sha_digest_t * *digest*, hsm_csr_random_prefix_t * *prefix*, ecc_fast_verification_signature_t * *signature*) ECDSA sign externally on CSR (Certificate Signing Request).

The ecc_curve_t of the key at `private_key_index` must belong to at least a 256-bit elliptic curve (e.g. NIST P-256).

The hsm_private_key_type_t of the key at `private_key_index` must be equal to HSM_PRIVATE_KEY_TYPE_CSR_SIGNER.

The hsm_public_key_algorithm_t of the key at `private_key_index` must be equal to HSM_PUBLIC_KEY_ALGORITHM_ECDSA.

Reference: "Pseudonym CSRs in ITS (Europe)", June 4, 2014.

Parameters

in	*service*	HSM service instance
in	*private_key_index*	Index of private key that should be used
in	*digest*	To-be-signed hash digest of CSR
out	*prefix*	Random prefix
out	*signature*	ECDSA fast verification signature

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t hsm_csr_ecdsa_public_keys_sign (hsm_service_t ∗ *service*, hsm_cell_index_t *private_key_index*, const hsm_cell_range_t ∗ *range_array_ptr*, size_t *range_array_size*, hsm_csr_random_prefix_t ∗ *prefix*, ecc_fast_verification_signature_t ∗ *signature*) ECDSA sign internally on public keys.

The ecc_curve_t of the key at `private_key_index` must belong to at least a 256-bit elliptic curve (e.g. NIST P-256).

The hsm_private_key_type_t of the key at `private_key_index` must be equal to HSM_PRIVATE_KEY_TYPE_CSR_SIGNER.

The hsm_public_key_algorithm_t of the key at `private_key_index` must be equal to HSM_PUBLIC_KEY_ALGORITHM_ECDSA.

All the private keys in the specified cell ranges must have hsm_private_key_type_t equal to HSM_PRIVATE_KEY_TYPE_CSR_MEMBER.

Reference: "Pseudonym CSRs in ITS (Europe)", June 4, 2014.

Parameters

in	*service*	HSM service instance
in	*private_key_index*	Index of private key that should be used
in	*range_array_ptr*	Array of cell ranges of private keys
in	*range_array_size*	Size of cell ranges array
out	*prefix*	Random prefix
out	*signature*	ECDSA fast verification signature

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t hsm_default_service_get (hsm_service_t ∗∗ *service_ptr*) Get pointer to default HSM service.

Parameters

out	*service_ptr*	Pointer to HSM service

Remarks

Should be implemented by system integration code.

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t hsm_ecc_private_key_create (hsm_service_t ∗ *service*, hsm_cell_index_t *private_key_index*, const hsm_ecc_private_key_info_t ∗ *private_key_info*) Create ECC private key.

Private key is stored in the secure storage cell at `private_key_index`.

`private_key_info::key_type` must be equal to one of HSM_PRIVATE_KEY_TYPE_ISOLATED, HSM_PRIVATE_KEY_TYPE_CSR_MEMBER, HSM_PRIVATE_KEY_TYPE_CSR_SIGNER or HSM_PRIVATE_KEY_TYPE_MA_INPUT.

If `private_key_info::key_type` is equal to HSM_PRIVATE_KEY_TYPE_CSR_SIGNER then `private_key_info::key_curve` must be at least a 256-bit elliptic curve (e.g. NIST P-256) and `private_key_info::key_algorithm` must be equal to HSM_PUBLIC_KEY_ALGORITHM_ECDSA.

Parameters

`in`	*service*	HSM service instance
`in`	*private_key_index*	Index where private key should be stored
`in`	*private_key_info*	Key information

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecdsa-example.c, craton-threadx/crypto/ecies-example.c, remote-posix/crypto/ecdsa-example.c, and remote-posix/crypto/ecies-example.c.

atlk_rc_t hsm_ecc_private_key_import (hsm_service_t ∗ *service*, hsm_cell_index_t *private_key_index*, const hsm_ecc_private_key_info_t ∗ *private_key_info*, const ecc_scalar_t ∗ *private_key*) Import ECC private key.

Store private key in the secure storage cell at `private_key_index`.

`private_key_info::key_type` must be equal to HSM_PRIVATE_KEY_TYPE_ISOLATED.

Parameters

`in`	*service*	HSM service instance
`in`	*private_key_index*	Index where private key should be stored
`in`	*private_key_info*	Key information
`in`	*private_key*	Private key value to be stored

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t hsm_ecc_private_key_info_get (hsm_service_t ∗ *service*, hsm_cell_index_t *private_key_index*, hsm_ecc_private_key_info_t ∗ *private_key_info*) Get private key information.

Parameters

`in`	*service*	HSM service instance
`in`	*private_key_index*	Index of private key
`out`	*private_key_info*	Key information

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t hsm_ecc_private_key_multiply_add (hsm_service_t ∗ *service,* hsm_cell_index_t *input_key_index,* hsm_cell_index_t *output_key_index,* const ecc_scalar_t ∗ *key_addend,* const ecc_scalar_t ∗ *key_multiplier*) Perform a modular multiply-add on stored private key and store the result.

Does the following operation: k' := b + (a ∗ k) mod n

Where: k is value at `input_key_index` k' is value at `output_key_index` b is `key_addend` a is `key_multiplier` n is order of the elliptic curve group that is specified by ecc_curve_t of the key at `input_key_index`

The hsm_private_key_type_t of the key at `input_key_index` must be HSM_PRIVATE_KEY_TYPE_MA_INPUT.

The newly created key at `output_key_index` will have the same ecc_curve_t and hsm_public_key_algorithm_t as the key at `input_key_index` but will have hsm_private_key_type_t equal to HSM_PRIVATE_KEY_TYPE_MA_OUTPUT.

This operation can be used to implement PKI schemes such as SCMS.

Remarks

`key_multiplier` is not allowed to be zero modulo the elliptic curve order (n) since allowing it will provide a method for importing private keys with hsm_private_key_type_t other than HSM_PRIVATE_KEY_TYPE_ISOLATED.
`input_key_index` is not allowed to be equal to `output_key_index` due to the idempotency requirement; i.e. invoking a procedure once and invoking the same procedure twice or more with the same inputs (without any other intervening procedure invocations) should be indistinguishable to the user.

Parameters

in	*service*	HSM service instance
in	*input_key_index*	Index of input private key
in	*output_key_index*	Index of output private key
in	*key_addend*	Scalar to add
in	*key_multiplier*	Scalar to multiply

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t hsm_ecc_public_key_get (hsm_service_t ∗ *service,* hsm_cell_index_t *private_key_index,* ecc_point_t ∗ *public_key*) Return ECC public key that matches a stored private key.

Parameters

in	*service*	HSM service instance
in	*private_key_index*	Index of stored private key
out	*public_key*	Public key that matches the private key

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecdsa-example.c, craton-threadx/crypto/ecies-example.c, remote-posix/crypto/ecdsa-example.-c, and remote-posix/crypto/ecies-example.c.

atlk_rc_t hsm_ecdsa_sign (hsm_service_t ∗ *service,* hsm_cell_index_t *private_key_index,* const sha_digest_t ∗ *digest,* ecc_-fast_verification_signature_t ∗ *signature*) Generate ECDSA signature from a given hash digest.

Note: when implementing standard ECDSA variants, if you use elliptic curve P-256 then `digest` should be computed using SHA-256; if you use P-224 then `digest` should be computed using SHA-224.

The hsm_private_key_type_t of the key at `private_key_index` must be equal to one of HSM_PRIVATE_KEY_TYP-E_CSR_MEMBER, HSM_PRIVATE_KEY_TYPE_MA_OUTPUT, or HSM_PRIVATE_KEY_TYPE_ISOLATED.

The hsm_public_key_algorithm_t of the key at `private_key_index` must be equal to HSM_PUBLIC_KEY_ALGORIT-HM_ECDSA.

Parameters

in	*service*	HSM service instance
in	*private_key_index*	Index of private key that should be used
in	*digest*	To-be-signed hash digest
out	*signature*	ECDSA fast verification signature

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecdsa-example.c, and remote-posix/crypto/ecdsa-example.c.

atlk_rc_t hsm_ecies_key_derive (hsm_service_t ∗ *service,* hsm_cell_index_t *private_key_index,* const ecc_point_t ∗ *peer_public_key,* void ∗ *key,* size_t *key_size,* const void ∗ *kdf_param,* size_t *kdf_param_size*) Derive ECIES key from a private key and peer public key.

The ecc_curve_t of the key at `private_key_index` must be at least a 256-bit elliptic curve (e.g. NIST P-256).

The hsm_private_key_type_t of the key at `private_key_index` must be equal to one of HSM_PRIVATE_KEY_TYPE_CSR_MEMBER, HSM_PRIVATE_KEY_TYPE_MA_OUTPUT, or HSM_PRIVATE_KEY_TYPE_ISOLATED.

The hsm_public_key_algorithm_t of the key at `private_key_index` must be equal to HSM_PUBLIC_KEY_ALGORITHM_ECIES.

`kdf_param` is an optional octet string used as a key derivation parameter. In order for the key derivation parameter to be the empty string, `kdf_param_size` must be 0. The key derivation parameter can be used to prevent misbinding attacks. Please refer to IEEE Std 1363a-2004 clause 11.3.2 where the key derivation parameter is denoted by P1.

Parameters

in	*service*	HSM service instance
in	*private_key_index*	Index of private key that should be used
in	*peer_public_key*	Public key of ECIES peer
out	*key*	Derived ECIES key
in	*key_size*	ECIES key size in octets
in	*kdf_param*	Key derivation parameter (optional)
in	*kdf_param_size*	Key derivation parameter size in octets

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecies-example.c, and remote-posix/crypto/ecies-example.c.

atlk_rc_t hsm_host_nvm_aes_cbc_decrypt (hsm_service_t ∗ *service,* const aes_cbc_iv_t ∗ *iv,* const void ∗ *ciphertext,* size_t *ciphertext_size,* void ∗ *plaintext,* size_t ∗ *plaintext_size*) Decrypt with AES-128-CBC data that is stored on a non-secure NVM.

Intended use case is to provide confidentiality for data (e.g. pseudonym certificates) that is stored on a non-secure host NVM.

The decryption key is internally generated with the following properties:

- Non-volatile: stays the same across power-cycles.
- Unique per unit: discovering one will not put other units at risk.

- Not stored on a non-secure NVM.

It is allowed that `plaintext` be equal to `ciphertext` in order to decrypt data in-place. Any other overlapping input and output buffers would result in undefined behavior.

`ciphertext_size` must be a multiple of AES_BLOCK_SIZE.

Parameters

`in`	*service*	HSM service instance
`in`	*iv*	Initialization vector
`in`	*ciphertext*	Ciphertext to decrypt
`in`	*ciphertext_size*	Size of the ciphertext in octets
`out`	*plaintext*	Plaintext
`in,out`	*plaintext_size*	The maximum size (in) and resulting size (out) of the plaintext in octets

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/secure-storage-example.c.

atlk_rc_t hsm_host_nvm_aes_cbc_encrypt (hsm_service_t ∗ *service*, const void ∗ *plaintext*, size_t *plaintext_size*, aes_cbc_iv_t ∗ *iv*, void ∗ *ciphertext*, size_t ∗ *ciphertext_size*) Encrypt with AES-128-CBC data to be stored on a non-secure NVM.

Intended use case is to provide confidentiality for data (e.g. pseudonym certificates) to be stored on a non-secure host NVM.

The encryption key is internally generated with the following properties:

- Non-volatile: stays the same across power-cycles.
- Unique per unit: discovering one will not put other units at risk.
- Not stored on a non-secure NVM.

It is allowed that `plaintext` be equal to `ciphertext` in order to encrypt data in-place. Any other overlapping input and output buffers would result in undefined behavior.

`plaintext_size` must be a multiple of AES_BLOCK_SIZE.

Parameters

`in`	*service*	HSM service instance
`in`	*plaintext*	Plaintext to encrypt
`in`	*plaintext_size*	Size of the plaintext in octets
`out`	*iv*	Initialization vector
`out`	*ciphertext*	Ciphertext
`in,out`	*ciphertext_size*	The maximum size (in) and resulting size (out) of the ciphertext in octets

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/secure-storage-example.c.

atlk_rc_t hsm_host_nvm_aes_cmac_compute (hsm_service_t ∗ *service*, const void ∗ *message*, size_t *message_size*, aes_cmac_tag_t ∗ *tag*) Compute AES-128-CMAC authentication tag of data stored on a non-secure NVM.

Intended use case is to provide integrity protection for data (e.g. root certificates) that is stored on a non-secure host NVM.

The authentication key is internally generated with the following properties:

- Non-volatile: stays the same across power-cycles.
- Unique per unit: discovering one will not put other units at risk.
- Not stored on a non-secure NVM.

Parameters

in	*service*	HSM service instance
in	*message*	Message to compute CMAC
in	*message_size*	Size of the message in octets
out	*tag*	AES-CMAC authentication tag

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/secure-storage-example.c.

atlk_rc_t hsm_nvm_init (hsm_service_t ∗ *service*, const hsm_nvm_config_t ∗ *config*) Initialize or re-initialize HSM NVM.

Parameters

in	*service*	HSM service instance
in	*config*	HSM NVM configuration

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecdsa-example.c, craton-threadx/crypto/ecies-example.c, craton-threadx/crypto/secure-storage-example.c, remote-posix/crypto/ecdsa-example.c, and remote-posix/crypto/ecies-example.c.

atlk_rc_t hsm_service_delete (hsm_service_t ∗ *service*) Delete HSM service.

Parameters

in	*service*	HSM service to be deleted

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecdsa-example.c, craton-threadx/crypto/ecies-example.c, craton-threadx/crypto/secure-storage-example.c, remote-posix/crypto/ecdsa-example.c, and remote-posix/crypto/ecies-example.c.

7.20 atlk/http_server.h File Reference

HTTP Server API.

```
   #include <atlk/sdk.h>
#include <atlk/os.h>
```

Data Structures

- struct http_url_entry_t

  URL entry.
- struct http_server_config_t

  HTTP server configuration parameters.

Macros

- #define HTTP_RESPONSE_CONTENT_MAX_LEN 1024

  Maximum response content size.
- #define HTTP_SERVER_CONFIG_INIT

  HTTP server configuration parameters default initializer.

Typedefs

- typedef atlk_rc_t(∗ http_url_handler_t)(const char ∗resource, http_request_type_t request_type, const void ∗request_-content, size_t request_content_size, void ∗response_content, size_t ∗response_content_size, uint16_t ∗status_code)

  URL handler.

Enumerations

- enum http_request_type_t {
  HTTP_REQUEST_TYPE_GET = 0, HTTP_REQUEST_TYPE_POST = 1, HTTP_REQUEST_TYPE_HEAD = 2, H-TTP_REQUEST_TYPE_PUT = 3,
  HTTP_REQUEST_TYPE_DELETE = 4 }

  HTTP request type.

Functions

- atlk_rc_t http_server_init (const http_server_config_t ∗config)

  Initialize HTTP server.
- atlk_rc_t http_server_module_register (const char ∗module_name, const http_url_entry_t ∗module_url_entry_array_ptr, size_t module_url_entry_array_count)

  Register HTTP module URL entries.

7.20.1 Detailed Description

HTTP Server API. Support is limited to HTTP/1.0. The following are not supported:

- Persistent connections
- Request pipelining
- Content compression
- TRACE, OPTIONS and CONNECT requests

7.20.2 Typedef Documentation

typedef atlk_rc_t(∗ http_url_handler_t)(const char ∗resource, http_request_type_t request_type, const void ∗request_content, size_t request_content_size, void ∗response_content, size_t ∗response_content_size, uint16_t ∗status_code) URL handler.

When request is not handled (ATLK_E_NOT_FOUND is returned), HTTP server will look for a page stored on FS with the same URL.

Parameters

in	*resource*	Resource name
in	*request_type*	HTTP request type
in	*request_content*	HTTP request content
in	*request_content_-size*	HTTP request content size
out	*response_content*	HTTP response content
in,out	*response_content-_size*	HTTP response content size
out	*status_code*	HTTP status code

Return values

ATLK_OK	if succeeded
ATLK_E_NOT_FOUND	if request is not handled by handler

Returns

Error code if failed

7.20.3 Enumeration Type Documentation

enum http_request_type_t HTTP request type.

Enumerator

HTTP_REQUEST_TYPE_GET HTTP GET request.

HTTP_REQUEST_TYPE_POST HTTP POST request.

HTTP_REQUEST_TYPE_HEAD HTTP HEAD request.

HTTP_REQUEST_TYPE_PUT HTTP PUT request.

HTTP_REQUEST_TYPE_DELETE HTTP DELETE request.

7.20.4 Function Documentation

atlk_rc_t http_server_init (const http_server_config_t ∗ *config*) Initialize HTTP server.

Parameters

in	*config*	HTTP server configuration parameters (optional)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/net/http-example.c.

atlk_rc_t http_server_module_register (const char ∗ *module_name*, const http_url_entry_t ∗ *module_url_entry_array_ptr*, size_t *module_url_entry_array_count*) Register HTTP module URL entries.

Parameters

in	*module_name*	Module name (must be unique)

in	*module_url_entry_-array_ptr*	Pointer to URL entries array
in	*module_url_entry_-array_count*	URL entries array count

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/net/http-example.c.

7.21 atlk/mib_remote.h File Reference

MIB remote service API.

```
#include <atlk/sdk.h>
#include <atlk/remote.h>
#include <atlk/mib_service.h>
```

Typedefs

- typedef struct mib_remote_service_config mib_remote_service_config_t

  MIB remote service configuration parameters.

Functions

- atlk_rc_t mib_remote_service_create (remote_transport_t *transport, const mib_remote_service_config_t *config, mib_service_t **service_ptr)

  Create MIB remote service.

7.21.1 Detailed Description

MIB remote service API.

7.21.2 Function Documentation

atlk_rc_t mib_remote_service_create (remote_transport_t * *transport*, const mib_remote_service_config_t * *config*, mib_service_t ** *service_ptr*) Create MIB remote service.

Parameters

in	*transport*	Remote transport instance
in	*config*	MIB remote service configuration (optional)
out	*service_ptr*	MIB service

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

remote-posix/mibs/mibs-example.c.

7.22 atlk/mib_service.h File Reference

MIB service API.

```
#include <atlk/sdk.h>
```

Typedefs

- typedef struct mib_service mib_service_t
  MIB service.

Functions

- atlk_rc_t mib_service_delete (mib_service_t *service)
  Delete MIB service instance.
- atlk_rc_t mib_default_service_get (mib_service_t **service_ptr)
  Get pointer to default MIB service.

7.22.1 Detailed Description

MIB service API.

7.22.2 Function Documentation

atlk_rc_t mib_default_service_get (mib_service_t ** *service_ptr*) Get pointer to default MIB service.

Parameters

out	*service_ptr*	MIB service

Note

Not supported by remote service library.

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/dot4/dot4-channel-switching-example.c, craton-threadx/gnss-teseo/gnss-teseo-sou-example.c, craton-threadx/gnss/gnss-integration-example.c, craton-threadx/mibs/mibs-edca-example.c, craton-threadx/mibs/mibs-example.c, craton-threadx/nav/system-time-benchmark.c, and craton-threadx/sntp/sntp-example.c.

atlk_rc_t mib_service_delete (mib_service_t * *service*) Delete MIB service instance.

Parameters

in	*service*	MIB service instance

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/gnss-teseo/gnss-teseo-sou-example.c, craton-threadx/gnss/gnss-integration-example.c, craton-threadx/mibs/mibs-edca-example.c, craton-threadx/mibs/mibs-example.c, craton-threadx/sntp/sntp-example.c, and remote-posix/mibs/mibs-example.c.

7.23 atlk/mibs/can-mibstat.h File Reference

CAN Status API.

```
#include <atlk/mib_service.h>
```

Data Structures

- struct mibstat_canDevEntry_t

  CAN device status.
- struct mibstat_canMib_t

  CAN status.

Macros

- #define MIB_canDevTable_SIZE 2

  CAN device table size.

Functions

- atlk_rc_t mibstat_get_canMib (mib_service_t ∗service, mibstat_canMib_t ∗value)

  Get CAN status.
- atlk_rc_t mibstat_reset_canMib (mib_service_t ∗service)

  Reset CAN status.

7.23.1 Detailed Description

CAN Status API.

7.23.2 Function Documentation

atlk_rc_t mibstat_get_canMib (mib_service_t ∗ *service*, mibstat_canMib_t ∗ *value*) Get CAN status.

Parameters

in	*service*	Instance of MIB service
out	*value*	CAN status value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mibstat_reset_canMib (mib_service_t ∗ *service*) Reset CAN status.

Parameters

in	*service*	Instance of MIB service

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.24 atlk/mibs/eth-mibstat.h File Reference

Ethernet Status API.

```
#include <atlk/mib_service.h>
```

Data Structures

- struct mibstat_ethMib_t

  Ethernet status.

Functions

- atlk_rc_t mibstat_get_ethMib (mib_service_t ∗service, mibstat_ethMib_t ∗value)

  Get Ethernet status.
- atlk_rc_t mibstat_reset_ethMib (mib_service_t ∗service)

  Reset Ethernet status.

7.24.1 Detailed Description

Ethernet Status API.

7.24.2 Function Documentation

atlk_rc_t mibstat_get_ethMib (mib_service_t ∗ *service*, mibstat_ethMib_t ∗ *value*) Get Ethernet status.
Parameters

in	*service*	Instance of MIB service
out	*value*	Ethernet status value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mibstat_reset_ethMib (mib_service_t ∗ *service*) Reset Ethernet status.
Parameters

in	*service*	Instance of MIB service

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.25 atlk/mibs/if-mib.h File Reference

Network interface MIB API.

```
   #include <atlk/mib_service.h>
#include <atlk/eui48.h>
```

Functions

- atlk_rc_t mib_get_ifPhysAddress (mib_service_t ∗service, int32_t if_index, eui48_t ∗value)

  Get physical address.

7.25.1 Detailed Description

Network interface MIB API.

7.25.2 Function Documentation

atlk_rc_t mib_get_ifPhysAddress (mib_service_t ∗ *service*, int32_t *if_index*, eui48_t ∗ *value*) Get physical address.

The Ethernet interface index number is 20.

Parameters

in	*service*	Instance of MIB service
in	*if_index*	Interface index
in	*value*	Physical address

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.26 atlk/mibs/inet-address-mib.h File Reference

Inet address MIB API.

Enumerations

- enum mib_InetAddressType_t {
  MIB_InetAddressType_unknown = 0, MIB_InetAddressType_ipv4 = 1, MIB_InetAddressType_ipv6 = 2, MIB_InetAddressType_ipv4z = 3,
  MIB_InetAddressType_ipv6z = 4 }

  Internet address type.

7.26.1 Detailed Description

Inet address MIB API.

7.26.2 Enumeration Type Documentation

enum mib_InetAddressType_t Internet address type.

Enumerator

MIB_InetAddressType_unknown An unknown address type.

MIB_InetAddressType_ipv4 An IPv4 address as defined by the InetAddressIPv4 textual convention.

MIB_InetAddressType_ipv6 An IPv6 address as defined by the InetAddressIPv6 textual convention.

MIB_InetAddressType_ipv4z A non-global IPv4 address including a zone index as defined by the InetAddressIPv4z textual convention.

MIB_InetAddressType_ipv6z A non-global IPv6 address including a zone index as defined by the InetAddressIPv6z textual convention.

7.27 atlk/mibs/nav-mib.h File Reference

AUTOTALKS-NAV-MIB access API.

```
  #include <atlk/mib_service.h>
#include <atlk/mibs/inet-address-mib.h>
#include <atlk/mibs/tc.h>
```

Macros

- #define MIB_navGpsSatelliteCnt_MIN 0

  Minimum navGpsSatelliteCnt value.
- #define MIB_navGpsSatelliteCnt_MAX 255

  Maximum navGpsSatelliteCnt value.
- #define MIB_navGlonassSatelliteCnt_MIN 0

  Minimum navGlonassSatelliteCnt value.
- #define MIB_navGlonassSatelliteCnt_MAX 255

  Maximum navGlonassSatelliteCnt value.

Enumerations

- enum mib_navDataSource_t

  Enumeration of navDataSource values.
- enum mib_navSysTimeStatus_t

  Enumeration of navSysTimeStatus values.
- enum mib_navSysTimeAccuracy_t

  Enumeration of navSysTimeAccuracy values.

Functions

- atlk_rc_t mib_get_navGpsSatelliteCnt (mib_service_t ∗service, uint32_t ∗value)

  Get navGpsSatelliteCnt value.
- atlk_rc_t mib_set_navGpsSatelliteCnt (mib_service_t ∗service, uint32_t value)

  Set navGpsSatelliteCnt value.
- atlk_rc_t mib_get_navGlonassSatelliteCnt (mib_service_t ∗service, uint32_t ∗value)

  Get navGlonassSatelliteCnt value.
- atlk_rc_t mib_set_navGlonassSatelliteCnt (mib_service_t ∗service, uint32_t value)

  Set navGlonassSatelliteCnt value.
- atlk_rc_t mib_get_navGnssRxBytesCnt (mib_service_t ∗service, uint32_t ∗value)

  Get navGnssRxBytesCnt value.
- atlk_rc_t mib_get_navGnssRxNmeaSentencesCnt (mib_service_t ∗service, uint32_t ∗value)

  Get navGnssRxNmeaSentencesCnt value.
- atlk_rc_t mib_get_navGnssTxBytesCnt (mib_service_t ∗service, uint32_t ∗value)

  Get navGnssTxBytesCnt value.
- atlk_rc_t mib_get_navGnssTxNmeaSentencesCnt (mib_service_t ∗service, uint32_t ∗value)

  Get navGnssTxNmeaSentencesCnt value.
- atlk_rc_t mib_get_navGnssRxNmeaSentencesErrorCnt (mib_service_t ∗service, uint32_t ∗value)

  Get navGnssRxNmeaSentencesErrorCnt value.
- atlk_rc_t mib_get_navGnssTxNmeaSentencesErrorCnt (mib_service_t ∗service, uint32_t ∗value)

  Get navGnssTxNmeaSentencesErrorCnt value.
- atlk_rc_t mib_get_navGnssAntennaStatus (mib_service_t ∗service, mib_AntennaStatus_t ∗value)

  Get navGnssAntennaStatus value.
- atlk_rc_t mib_get_navGnssAntennaOffsetX (mib_service_t ∗service, int32_t ∗value)

  Get navGnssAntennaOffsetX value (units: cm).
- atlk_rc_t mib_set_navGnssAntennaOffsetX (mib_service_t ∗service, int32_t value)

  Set navGnssAntennaOffsetX value (units: cm).
- atlk_rc_t mib_get_navGnssAntennaOffsetY (mib_service_t ∗service, int32_t ∗value)

  Get navGnssAntennaOffsetY value (units: cm).
- atlk_rc_t mib_set_navGnssAntennaOffsetY (mib_service_t ∗service, int32_t value)

  Set navGnssAntennaOffsetY value (units: cm).
- atlk_rc_t mib_get_navGnssAntennaOffsetZ (mib_service_t ∗service, int32_t ∗value)

  Get navGnssAntennaOffsetZ value (units: cm).
- atlk_rc_t mib_set_navGnssAntennaOffsetZ (mib_service_t ∗service, int32_t value)

  Set navGnssAntennaOffsetZ value (units: cm).
- atlk_rc_t mib_get_navFixAvailable (mib_service_t ∗service, int ∗value)

  Get navFixAvailable value.
- atlk_rc_t mib_set_navFixAvailable (mib_service_t ∗service, int value)

  Set navFixAvailable value.
- atlk_rc_t mib_get_navDataSource (mib_service_t ∗service, mib_navDataSource_t ∗value)

  Get navDataSource value.
- atlk_rc_t mib_set_navDataSource (mib_service_t ∗service, mib_navDataSource_t value)

Set navDataSource value.
- atlk_rc_t mib_get_navGpsdServerPort (mib_service_t ∗service, uint16_t ∗value)

  Get navGpsdServerPort value.
- atlk_rc_t mib_set_navGpsdServerPort (mib_service_t ∗service, uint16_t value)

  Set navGpsdServerPort value.
- atlk_rc_t mib_get_navGpsdServerIpAddressType (mib_service_t ∗service, mib_InetAddressType_t ∗value)

  Get navGpsdServerIpAddressType value.
- atlk_rc_t mib_get_navGpsdServerIpAddressIPv4 (mib_service_t ∗service, uint32_t ∗value)

  Get navGpsdServerIpAddressIPv4 value.
- atlk_rc_t mib_set_navGpsdServerIpAddressIPv4 (mib_service_t ∗service, uint32_t value)

  Set navGpsdServerIpAddressIPv4 value.
- atlk_rc_t mib_get_navSysTimeUpdateEnabled (mib_service_t ∗service, int ∗value)

  Get navSysTimeUpdateEnabled value.
- atlk_rc_t mib_set_navSysTimeUpdateEnabled (mib_service_t ∗service, int value)

  Set navSysTimeUpdateEnabled value.
- atlk_rc_t mib_get_navSysTimePpsSyncEnabled (mib_service_t ∗service, int ∗value)

  Get navSysTimePpsSyncEnabled value.
- atlk_rc_t mib_set_navSysTimePpsSyncEnabled (mib_service_t ∗service, int value)

  Set navSysTimePpsSyncEnabled value.
- atlk_rc_t mib_get_navSysTimeStatus (mib_service_t ∗service, mib_navSysTimeStatus_t ∗value)

  Get navSysTimeStatus value.
- atlk_rc_t mib_set_navSysTimeStatus (mib_service_t ∗service, mib_navSysTimeStatus_t value)

  Set navSysTimeStatus value.
- atlk_rc_t mib_get_navSysTimeLeapSeconds (mib_service_t ∗service, int32_t ∗value)

  Get navSysTimeLeapSeconds value.
- atlk_rc_t mib_set_navSysTimeLeapSeconds (mib_service_t ∗service, int32_t value)

  Set navSysTimeLeapSeconds value.
- atlk_rc_t mib_get_navSysTimePpsCnt (mib_service_t ∗service, uint32_t ∗value)

  Get navSysTimePpsCnt value.
- atlk_rc_t mib_get_navSysTimeAccuracy (mib_service_t ∗service, mib_navSysTimeAccuracy_t ∗value)

  Get navSysTimeAccuracy value.
- atlk_rc_t mib_set_navSysTimeAccuracy (mib_service_t ∗service, mib_navSysTimeAccuracy_t value)

  Set navSysTimeAccuracy value.
- atlk_rc_t mib_get_navConfigSaveStatus (mib_service_t ∗service, mib_ConfigSaveStatus_t ∗value)

  Get navConfigSaveStatus value.
- atlk_rc_t mib_set_navConfigSaveStatus (mib_service_t ∗service, mib_ConfigSaveStatus_t value)

  Set navConfigSaveStatus value.
- atlk_rc_t mib_get_navGnssInputDeviceIndex (mib_service_t ∗service, uint16_t ∗value)

  Get navGnssInputDeviceIndex value.
- atlk_rc_t mib_set_navGnssInputDeviceIndex (mib_service_t ∗service, uint16_t value)

  Set navGnssInputDeviceIndex value.

7.27.1 Detailed Description

AUTOTALKS-NAV-MIB access API. Navigation MIB.

7.27.2 Function Documentation

atlk_rc_t mib_get_navConfigSaveStatus (mib_service_t ∗ *service*, mib_ConfigSaveStatus_t ∗ *value*) Get navConfigSave-Status value.

Navigation MIB configuration save status.

Parameters

in	*service*	Instance of MIB service
out	*value*	navConfigSaveStatus value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_navDataSource (mib_service_t ∗ *service*, mib_navDataSource_t ∗ *value*) Get navDataSource value.

Source of navigation data fix.

Parameters

in	*service*	Instance of MIB service
out	*value*	navDataSource value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_navFixAvailable (mib_service_t ∗ *service*, int ∗ *value*) Get navFixAvailable value.

Whether a navigation fix is available (via satellite navigation and/or another method).

Parameters

in	*service*	Instance of MIB service
out	*value*	navFixAvailable value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/nav/system-time-benchmark.c.

atlk_rc_t mib_get_navGlonassSatelliteCnt (mib_service_t ∗ *service*, uint32_t ∗ *value*) Get navGlonassSatelliteCnt value.

Number of GLONASS satellites in view.

Parameters

in	*service*	Instance of MIB service
out	*value*	navGlonassSatelliteCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_navGnssAntennaOffsetX (mib_service_t ∗ *service*, int32_t ∗ *value*) Get navGnssAntennaOffsetX value (units: cm).

Antenna offset on axis X in centimeters, relative to vehicles position reference. Axis X is positive towards vehicles front.

Parameters

in	*service*	Instance of MIB service
out	*value*	navGnssAntennaOffsetX value (units: cm)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_navGnssAntennaOffsetY (mib_service_t * *service,* int32_t * *value*) Get navGnssAntennaOffsetY value (units: cm).

Antenna offset on axis Y in centimeters, relative to vehicles position reference. Axis Y is positive towards vehicles right hand side.

Parameters

in	*service*	Instance of MIB service
out	*value*	navGnssAntennaOffsetY value (units: cm)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_navGnssAntennaOffsetZ (mib_service_t * *service,* int32_t * *value*) Get navGnssAntennaOffsetZ value (units: cm).

Antenna offset on axis Z in centimeters, relative to vehicles position reference. Axis Z is positive towards ground (i.e. down).

Parameters

in	*service*	Instance of MIB service
out	*value*	navGnssAntennaOffsetZ value (units: cm)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_navGnssAntennaStatus (mib_service_t * *service,* mib_AntennaStatus_t * *value*) Get navGnssAntenna-Status value.

Current status of GNSS antenna.

Parameters

in	*service*	Instance of MIB service
out	*value*	navGnssAntennaStatus value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_navGnssInputDeviceIndex (mib_service_t ∗ *service*, uint16_t ∗ *value*) Get navGnssInputDeviceIndex value.
Index of currently selected GNSS input device.
Parameters

in	*service*	Instance of MIB service
out	*value*	navGnssInputDeviceIndex value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_navGnssRxBytesCnt (mib_service_t ∗ *service*, uint32_t ∗ *value*) Get navGnssRxBytesCnt value.
Count of bytes read via NMEA I/O.
Parameters

in	*service*	Instance of MIB service
out	*value*	navGnssRxBytesCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_navGnssRxNmeaSentencesCnt (mib_service_t ∗ *service*, uint32_t ∗ *value*) Get navGnssRxNmeaSentencesCnt value.
Count of valid NMEA sentences read via NMEA I/O.
Parameters

in	*service*	Instance of MIB service
out	*value*	navGnssRxNmeaSentencesCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_navGnssRxNmeaSentencesErrorCnt (mib_service_t ∗ *service*, uint32_t ∗ *value*) Get navGnssRxNmeaSentencesErrorCnt value.
Count of invalid NMEA sentences dropped at NMEA I/O before being parsed.

Parameters

in	*service*	Instance of MIB service
out	*value*	navGnssRxNmeaSentencesErrorCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_navGnssTxBytesCnt (mib_service_t ∗ *service*, uint32_t ∗ *value*) Get navGnssTxBytesCnt value.

Count of bytes written via NMEA I/O.

Parameters

in	*service*	Instance of MIB service
out	*value*	navGnssTxBytesCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_navGnssTxNmeaSentencesCnt (mib_service_t ∗ *service*, uint32_t ∗ *value*) Get navGnssTxNmeaSentences-Cnt value.

Count of valid NMEA sentences written via NMEA I/O.

Parameters

in	*service*	Instance of MIB service
out	*value*	navGnssTxNmeaSentencesCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_navGnssTxNmeaSentencesErrorCnt (mib_service_t ∗ *service*, uint32_t ∗ *value*) Get navGnssTxNmea-SentencesErrorCnt value.

Count of invalid NMEA sentences dropped at NMEA I/O before being written.

Parameters

in	*service*	Instance of MIB service
out	*value*	navGnssTxNmeaSentencesErrorCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_navGpsdServerIpAddressIPv4 (mib_service_t ∗ *service*, uint32_t ∗ *value*) Get navGpsdServerIpAddressI-Pv4 value.

IPv4 address used by GPSD server.

Parameters

in	*service*	Instance of MIB service
out	*value*	navGpsdServerIpAddressIPv4 value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_navGpsdServerIpAddressType (mib_service_t * *service*, mib_InetAddressType_t * *value*) Get navGpsd-ServerIpAddressType value.

Address type used by GPSD server.

Parameters

in	*service*	Instance of MIB service
out	*value*	navGpsdServerIpAddressType value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_navGpsdServerPort (mib_service_t * *service*, uint16_t * *value*) Get navGpsdServerPort value.

Port number used by GPSD server.

Parameters

in	*service*	Instance of MIB service
out	*value*	navGpsdServerPort value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_navGpsSatelliteCnt (mib_service_t * *service*, uint32_t * *value*) Get navGpsSatelliteCnt value.

Number of GPS satellites in view.

Currently this value includes both GPS and SBAS satellites (PRNs 1-64).

Parameters

in	*service*	Instance of MIB service
out	*value*	navGpsSatelliteCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_navSysTimeAccuracy (mib_service_t * *service,* mib_navSysTimeAccuracy_t * *value*) Get navSysTime-Accuracy value.

System time accuracy.

– 41: The time is accurate to within 1 ms – 48: The time is accurate to within 1 s – 48: The time is accurate to within 10 s – 49: The time is accurate to within $>$ 10 s – 254: Default indicating unknown

Parameters

in	*service*	Instance of MIB service
out	*value*	navSysTimeAccuracy value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/dot4/dot4-channel-switching-example.c.

atlk_rc_t mib_get_navSysTimeLeapSeconds (mib_service_t ∗ *service,* int32_t ∗ *value*) Get navSysTimeLeapSeconds value.

Net amount of UTC leap seconds, between 2004-01-01T00:00:00Z and current time, which shall be used in conversion between TAI and UTC times in set/gettimeofday function calls.

Parameters

in	*service*	Instance of MIB service
out	*value*	navSysTimeLeapSeconds value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_navSysTimePpsCnt (mib_service_t ∗ *service,* uint32_t ∗ *value*) Get navSysTimePpsCnt value.

This counter shall be incremented after each pulse of the pulse-per-second (PPS) signal.

Parameters

in	*service*	Instance of MIB service
out	*value*	navSysTimePpsCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_navSysTimePpsSyncEnabled (mib_service_t ∗ *service,* int ∗ *value*) Get navSysTimePpsSyncEnabled value.

Whether syncing system time with external 1-PPS is enabled.

Parameters

in	*service*	Instance of MIB service
out	*value*	navSysTimePpsSyncEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_navSysTimeStatus (mib_service_t * *service*, mib_navSysTimeStatus_t * *value*) Get navSysTimeStatus value.

System time status.

Parameters

in	*service*	Instance of MIB service
out	*value*	navSysTimeStatus value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/nav/system-time-benchmark.c.

atlk_rc_t mib_get_navSysTimeUpdateEnabled (mib_service_t * *service*, int * *value*) Get navSysTimeUpdateEnabled value.

Whether updating system time from navigation fix is enabled.

Parameters

in	*service*	Instance of MIB service
out	*value*	navSysTimeUpdateEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_navConfigSaveStatus (mib_service_t * *service*, mib_ConfigSaveStatus_t *value*) Set navConfigSaveStatus value.

Navigation MIB configuration save status.

Parameters

in	*service*	Instance of MIB service
in	*value*	navConfigSaveStatus value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_navDataSource (mib_service_t * *service*, mib_navDataSource_t *value*) Set navDataSource value.

Source of navigation data fix.

Parameters

in	*service*	Instance of MIB service
in	*value*	navDataSource value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/gnss/gnss-integration-example.c.

atlk_rc_t mib_set_navFixAvailable (mib_service_t ∗ *service*, int *value*) Set navFixAvailable value.

Whether a navigation fix is available (via satellite navigation and/or another method).

Parameters

in	*service*	Instance of MIB service
in	*value*	navFixAvailable value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_navGlonassSatelliteCnt (mib_service_t ∗ *service*, uint32_t *value*) Set navGlonassSatelliteCnt value.

Number of GLONASS satellites in view.

Parameters

in	*service*	Instance of MIB service
in	*value*	navGlonassSatelliteCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_navGnssAntennaOffsetX (mib_service_t ∗ *service*, int32_t *value*) Set navGnssAntennaOffsetX value (units: cm).

Antenna offset on axis X in centimeters, relative to vehicles position reference. Axis X is positive towards vehicles front.

Parameters

in	*service*	Instance of MIB service
in	*value*	navGnssAntennaOffsetX value (units: cm)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/gnss-teseo/gnss-teseo-sou-example.c.

atlk_rc_t mib_set_navGnssAntennaOffsetY (mib_service_t * *service*, int32_t *value*) Set navGnssAntennaOffsetY value (units: cm).

Antenna offset on axis Y in centimeters, relative to vehicles position reference. Axis Y is positive towards vehicles right hand side.

Parameters

in	*service*	Instance of MIB service
in	*value*	navGnssAntennaOffsetY value (units: cm)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/gnss-teseo/gnss-teseo-sou-example.c.

atlk_rc_t mib_set_navGnssAntennaOffsetZ (mib_service_t * *service*, int32_t *value*) Set navGnssAntennaOffsetZ value (units: cm).

Antenna offset on axis Z in centimeters, relative to vehicles position reference. Axis Z is positive towards ground (i.e. down).

Parameters

in	*service*	Instance of MIB service
in	*value*	navGnssAntennaOffsetZ value (units: cm)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/gnss-teseo/gnss-teseo-sou-example.c.

atlk_rc_t mib_set_navGnssInputDeviceIndex (mib_service_t * *service*, uint16_t *value*) Set navGnssInputDeviceIndex value.

Index of currently selected GNSS input device.

Parameters

in	*service*	Instance of MIB service
in	*value*	navGnssInputDeviceIndex value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_navGpsdServerIpAddressIPv4 (mib_service_t * *service*, uint32_t *value*) Set navGpsdServerIpAddressIPv4 value.

IPv4 address used by GPSD server.

Parameters

in	*service*	Instance of MIB service
in	*value*	navGpsdServerIpAddressIPv4 value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_navGpsdServerPort (mib_service_t ∗ *service*, uint16_t *value*) Set navGpsdServerPort value.
Port number used by GPSD server.
Parameters

in	*service*	Instance of MIB service
in	*value*	navGpsdServerPort value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_navGpsSatelliteCnt (mib_service_t ∗ *service*, uint32_t *value*) Set navGpsSatelliteCnt value.
Number of GPS satellites in view.
Currently this value includes both GPS and SBAS satellites (PRNs 1-64).
Parameters

in	*service*	Instance of MIB service
in	*value*	navGpsSatelliteCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_navSysTimeAccuracy (mib_service_t ∗ *service*, mib_navSysTimeAccuracy_t *value*) Set navSysTime-Accuracy value.
System time accuracy.
– 41: The time is accurate to within 1 ms – 48: The time is accurate to within 1 s – 48: The time is accurate to within 10 s – 49: The time is accurate to within $>$ 10 s – 254: Default indicating unknown
Parameters

in	*service*	Instance of MIB service
in	*value*	navSysTimeAccuracy value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_navSysTimeLeapSeconds (mib_service_t * *service,* int32_t *value*) Set navSysTimeLeapSeconds value.

Net amount of UTC leap seconds, between 2004-01-01T00:00:00Z and current time, which shall be used in conversion between TAI and UTC times in set/gettimeofday function calls.

Parameters

in	*service*	Instance of MIB service
in	*value*	navSysTimeLeapSeconds value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_navSysTimePpsSyncEnabled (mib_service_t * *service*, int *value*) Set navSysTimePpsSyncEnabled value.

Whether syncing system time with external 1-PPS is enabled.

Parameters

in	*service*	Instance of MIB service
in	*value*	navSysTimePpsSyncEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/sntp/sntp-example.c.

atlk_rc_t mib_set_navSysTimeStatus (mib_service_t * *service*, mib_navSysTimeStatus_t *value*) Set navSysTimeStatus value.

System time status.

Parameters

in	*service*	Instance of MIB service
in	*value*	navSysTimeStatus value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_navSysTimeUpdateEnabled (mib_service_t * *service*, int *value*) Set navSysTimeUpdateEnabled value.

Whether updating system time from navigation fix is enabled.

Parameters

in	*service*	Instance of MIB service
in	*value*	navSysTimeUpdateEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.28 atlk/mibs/profiling-mibstat.h File Reference

Profiling Status API.

```
#include <atlk/mib_service.h>
```

Data Structures

- struct mibstat_profilingThreadEntry_t

  Profiling thread status.
- struct mibstat_profilingMib_t

  Profiling status.

Macros

- #define MIB_profilingThreadName_SIZE_MAX 32

  Profiling thread name max length.
- #define MIB_profilingThreadsTable_SIZE 32

  Profiling threads table size.

Functions

- atlk_rc_t mibstat_get_profilingMib (mib_service_t ∗service, mibstat_profilingMib_t ∗value)

  Get profiling status.
- atlk_rc_t mibstat_reset_profilingMib (mib_service_t ∗service)

  Reset profiling status.

7.28.1 Detailed Description

Profiling Status API.

7.28.2 Function Documentation

atlk_rc_t mibstat_get_profilingMib (mib_service_t ∗ *service*, mibstat_profilingMib_t ∗ *value*) Get profiling status.

Parameters

in	*service*	Instance of MIB service
out	*value*	Profiling status value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mibstat_reset_profilingMib (mib_service_t ∗ *service*) Reset profiling status.

Parameters

in	*service*	Instance of MIB service

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.29 atlk/mibs/rsvc-mib.h File Reference

AUTOTALKS-RSVC-MIB access API.

```
   #include <atlk/mib_service.h>
#include <atlk/mibs/snmpv2-tc.h>
```

Enumerations

- enum mib_rsvcWlanFwdFrameType_t
  Enumeration of rsvcWlanFwdFrameType values.

Functions

- atlk_rc_t mib_get_rsvcDefaultDestIpAddressIPv4 (mib_service_t ∗service, uint32_t ∗value)
  Get rsvcDefaultDestIpAddressIPv4 value.
- atlk_rc_t mib_set_rsvcDefaultDestIpAddressIPv4 (mib_service_t ∗service, uint32_t value)
  Set rsvcDefaultDestIpAddressIPv4 value.
- atlk_rc_t mib_get_rsvcWlanFwdRowStatus (mib_service_t ∗service, int32_t rsvcWlanFwdIndex, mib_RowStatus_t ∗value)
  Get rsvcWlanFwdRowStatus value.
- atlk_rc_t mib_set_rsvcWlanFwdRowStatus (mib_service_t ∗service, int32_t rsvcWlanFwdIndex, mib_RowStatus_t value)
  Set rsvcWlanFwdRowStatus value.
- atlk_rc_t mib_get_rsvcWlanFwdDestPortNumber (mib_service_t ∗service, int32_t rsvcWlanFwdIndex, uint16_t ∗value)
  Get rsvcWlanFwdDestPortNumber value.
- atlk_rc_t mib_set_rsvcWlanFwdDestPortNumber (mib_service_t ∗service, int32_t rsvcWlanFwdIndex, uint16_t value)
  Set rsvcWlanFwdDestPortNumber value.
- atlk_rc_t mib_get_rsvcWlanFwdIfIndex (mib_service_t ∗service, int32_t rsvcWlanFwdIndex, int32_t ∗value)
  Get rsvcWlanFwdIfIndex value.
- atlk_rc_t mib_set_rsvcWlanFwdIfIndex (mib_service_t ∗service, int32_t rsvcWlanFwdIndex, int32_t value)
  Set rsvcWlanFwdIfIndex value.
- atlk_rc_t mib_get_rsvcWlanFwdFrameType (mib_service_t ∗service, int32_t rsvcWlanFwdIndex, mib_rsvcWlanFwd-FrameType_t ∗value)
  Get rsvcWlanFwdFrameType value.
- atlk_rc_t mib_set_rsvcWlanFwdFrameType (mib_service_t ∗service, int32_t rsvcWlanFwdIndex, mib_rsvcWlanFwd-FrameType_t value)
  Set rsvcWlanFwdFrameType value.
- atlk_rc_t mib_get_rsvcWlanFwdProtocolId (mib_service_t ∗service, int32_t rsvcWlanFwdIndex, uint64_t ∗value)
  Get rsvcWlanFwdProtocolId value.
- atlk_rc_t mib_set_rsvcWlanFwdProtocolId (mib_service_t ∗service, int32_t rsvcWlanFwdIndex, uint64_t value)
  Set rsvcWlanFwdProtocolId value.
- atlk_rc_t mib_get_rsvcWlanBridgeEnabled (mib_service_t ∗service, int ∗value)
  Get rsvcWlanBridgeEnabled value.
- atlk_rc_t mib_set_rsvcWlanBridgeEnabled (mib_service_t ∗service, int value)
  Set rsvcWlanBridgeEnabled value.
- atlk_rc_t mib_get_rsvcWlanBridgeIfIndex (mib_service_t ∗service, int32_t ∗value)
  Get rsvcWlanBridgeIfIndex value.
- atlk_rc_t mib_set_rsvcWlanBridgeIfIndex (mib_service_t ∗service, int32_t value)
  Set rsvcWlanBridgeIfIndex value.
- atlk_rc_t mib_get_rsvcWlanBridgeVlanId (mib_service_t ∗service, int32_t ∗value)
  Get rsvcWlanBridgeVlanId value.
- atlk_rc_t mib_set_rsvcWlanBridgeVlanId (mib_service_t ∗service, int32_t value)
  Set rsvcWlanBridgeVlanId value.

7.29.1 Detailed Description

AUTOTALKS-RSVC-MIB access API. Autotalks Remote Services MIB.

7.29.2 Function Documentation

atlk_rc_t mib_get_rsvcDefaultDestIpAddressIPv4 (mib_service_t ∗ *service*, uint32_t ∗ *value*) Get rsvcDefaultDestIpAddressIPv4 value.

Default destination IPv4 address for Remote Services messages.

Parameters

in	*service*	Instance of MIB service
out	*value*	rsvcDefaultDestIpAddressIPv4 value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_rsvcWlanBridgeEnabled (mib_service_t ∗ *service*, int ∗ *value*) Get rsvcWlanBridgeEnabled value.

Whether bridging of WLAN to local VLAN is enabled.

Parameters

in	*service*	Instance of MIB service
out	*value*	rsvcWlanBridgeEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_rsvcWlanBridgeIfIndex (mib_service_t ∗ *service*, int32_t ∗ *value*) Get rsvcWlanBridgeIfIndex value.

WLAN interface index that's bridged to a local VLAN ID.

Parameters

in	*service*	Instance of MIB service
out	*value*	rsvcWlanBridgeIfIndex value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_rsvcWlanBridgeVlanId (mib_service_t ∗ *service*, int32_t ∗ *value*) Get rsvcWlanBridgeVlanId value.

VLAN ID of the VLAN that's bridged to WLAN interface selected by rsvcWlanBridgeIfIndex.

Parameters

in	*service*	Instance of MIB service

out	*value*	rsvcWlanBridgeVlanId value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_rsvcWlanFwdDestPortNumber (mib_service_t * *service*, int32_t *rsvcWlanFwdIndex*, uint16_t * *value*) Get rsvcWlanFwdDestPortNumber value.

Destination port for forwarded WLAN packets.

Parameters

in	*service*	Instance of MIB service
in	*rsvcWlanFwd-Index*	rsvcWlanFwdIndex value
out	*value*	rsvcWlanFwdDestPortNumber value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_rsvcWlanFwdFrameType (mib_service_t * *service*, int32_t *rsvcWlanFwdIndex*, mib_rsvcWlanFwdFrame-Type_t * *value*) Get rsvcWlanFwdFrameType value.

WLAN frame type.

Value 'vsa' is not supported.

Parameters

in	*service*	Instance of MIB service
in	*rsvcWlanFwd-Index*	rsvcWlanFwdIndex value
out	*value*	rsvcWlanFwdFrameType value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_rsvcWlanFwdIfIndex (mib_service_t * *service*, int32_t *rsvcWlanFwdIndex*, int32_t * *value*) Get rsvcWlan-FwdIfIndex value.

WLAN MAC interface index.

Parameters

in	*service*	Instance of MIB service
in	*rsvcWlanFwd-Index*	rsvcWlanFwdIndex value

out	*value*	rsvcWlanFwdIfIndex value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_rsvcWlanFwdProtocolId (mib_service_t ∗ *service,* int32_t *rsvcWlanFwdIndex,* uint64_t ∗ *value*) Get rsvcWlanFwdProtocolId value.

WLAN packet protocol ID.

Parameters

in	*service*	Instance of MIB service
in	*rsvcWlanFwd-Index*	rsvcWlanFwdIndex value
out	*value*	rsvcWlanFwdProtocolId value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_rsvcWlanFwdRowStatus (mib_service_t ∗ *service,* int32_t *rsvcWlanFwdIndex,* mib_RowStatus_t ∗ *value*) Get rsvcWlanFwdRowStatus value.

WLAN packet forwarding conceptual row status.

Parameters

in	*service*	Instance of MIB service
in	*rsvcWlanFwd-Index*	rsvcWlanFwdIndex value
out	*value*	rsvcWlanFwdRowStatus value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_rsvcDefaultDestIpAddressIPv4 (mib_service_t ∗ *service,* uint32_t *value*) Set rsvcDefaultDestIpAddressIPv4 value.

Default destination IPv4 address for Remote Services messages.

Parameters

in	*service*	Instance of MIB service
in	*value*	rsvcDefaultDestIpAddressIPv4 value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_rsvcWlanBridgeEnabled (mib_service_t ∗ *service,* int *value*) Set rsvcWlanBridgeEnabled value.

Whether bridging of WLAN to local VLAN is enabled.

Parameters

in	*service*	Instance of MIB service
in	*value*	rsvcWlanBridgeEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_rsvcWlanBridgeIfIndex (mib_service_t ∗ *service,* int32_t *value*) Set rsvcWlanBridgeIfIndex value.

WLAN interface index that's bridged to a local VLAN ID.

Parameters

in	*service*	Instance of MIB service
in	*value*	rsvcWlanBridgeIfIndex value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_rsvcWlanBridgeVlanId (mib_service_t ∗ *service,* int32_t *value*) Set rsvcWlanBridgeVlanId value.

VLAN ID of the VLAN that's bridged to WLAN interface selected by rsvcWlanBridgeIfIndex.

Parameters

in	*service*	Instance of MIB service
in	*value*	rsvcWlanBridgeVlanId value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_rsvcWlanFwdDestPortNumber (mib_service_t ∗ *service,* int32_t *rsvcWlanFwdIndex,* uint16_t *value*) Set rsvcWlanFwdDestPortNumber value.

Destination port for forwarded WLAN packets.

Parameters

in	*service*	Instance of MIB service
in	*rsvcWlanFwd-Index*	rsvcWlanFwdIndex value
in	*value*	rsvcWlanFwdDestPortNumber value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_rsvcWlanFwdFrameType (mib_service_t ∗ *service*, int32_t *rsvcWlanFwdIndex*, mib_rsvcWlanFwdFrame-Type_t *value*) Set rsvcWlanFwdFrameType value.

WLAN frame type.

Value 'vsa' is not supported.

Parameters

in	*service*	Instance of MIB service
in	*rsvcWlanFwd-Index*	rsvcWlanFwdIndex value
in	*value*	rsvcWlanFwdFrameType value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_rsvcWlanFwdIfIndex (mib_service_t ∗ *service*, int32_t *rsvcWlanFwdIndex*, int32_t *value*) Set rsvcWlan-FwdIfIndex value.

WLAN MAC interface index.

Parameters

in	*service*	Instance of MIB service
in	*rsvcWlanFwd-Index*	rsvcWlanFwdIndex value
in	*value*	rsvcWlanFwdIfIndex value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_rsvcWlanFwdProtocolId (mib_service_t ∗ *service*, int32_t *rsvcWlanFwdIndex*, uint64_t *value*) Set rsvc-WlanFwdProtocolId value.

WLAN packet protocol ID.

Parameters

in	*service*	Instance of MIB service

in	*rsvcWlanFwd-Index*	rsvcWlanFwdIndex value
in	*value*	rsvcWlanFwdProtocolId value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_rsvcWlanFwdRowStatus (mib_service_t ∗ *service*, int32_t *rsvcWlanFwdIndex*, mib_RowStatus_t *value*)
Set rsvcWlanFwdRowStatus value.
WLAN packet forwarding conceptual row status.
Parameters

in	*service*	Instance of MIB service
in	*rsvcWlanFwd-Index*	rsvcWlanFwdIndex value
in	*value*	rsvcWlanFwdRowStatus value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.30 atlk/mibs/slx97-mibstat.h File Reference

SLx97 Status API.

```
#include <atlk/mib_service.h>
```

Data Structures

- struct mibstat_slx97Mib_t

  SLx97 status.

Functions

- atlk_rc_t mibstat_get_slx97Mib (mib_service_t ∗service, mibstat_slx97Mib_t ∗value)

  Get SLx97 status.
- atlk_rc_t mibstat_reset_slx97Mib (mib_service_t ∗service)

  Reset SLx97 status.

7.30.1 Detailed Description

SLx97 Status API.

7.30.2 Function Documentation

atlk_rc_t mibstat_get_slx97Mib (mib_service_t ∗ *service*, mibstat_slx97Mib_t ∗ *value*) Get SLx97 status.
Parameters

in	*service*	Instance of MIB service
out	*value*	SLx97 status value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mibstat_reset_slx97Mib (mib_service_t * *service*) Reset SLx97 status.

Parameters

in	*service*	Instance of MIB service

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.31 atlk/mibs/snmpv2-mib.h File Reference

SNMPv2 MIB API.

```
#include <atlk/mib_service.h>
```

Functions

- atlk_rc_t mib_get_sysDescr (mib_service_t *service, char *value, size_t *size)
  Get system description.

7.31.1 Detailed Description

SNMPv2 MIB API.

7.31.2 Function Documentation

atlk_rc_t mib_get_sysDescr (mib_service_t * *service*, char * *value*, size_t * *size*) Get system description.

On success, `size` will be set to actual length of description string. The system description string is guaranteed to be null-terminated.

Parameters

in	*service*	Instance of MIB service
in	*value*	System description
in,out	*size*	Description string length in octets

Return values

ATLK_OK	if succeeded
ATLK_E_BUFFER_TOO_SMALL	if `size` is too small

Returns

Error code if failed

Examples:

craton-threadx/mibs/mibs-example.c, and remote-posix/mibs/mibs-example.c.

7.32 atlk/mibs/snmpv2-tc.h File Reference

Mapping of types in SNMPv2-TC.

Enumerations

- enum mib_RowStatus_t {
  MIB_RowStatus_active = 1, MIB_RowStatus_notInService = 2, MIB_RowStatus_notReady = 3, MIB_RowStatus_-createAndGo = 4,
  MIB_RowStatus_createAndWait = 5, MIB_RowStatus_destroy = 6 }

  Status of a MIB table conceptual row.

7.32.1 Detailed Description

Mapping of types in SNMPv2-TC.

7.32.2 Enumeration Type Documentation

enum mib_RowStatus_t Status of a MIB table conceptual row.

Enumerator

MIB_RowStatus_active Indicates that the conceptual row with all columns is available for use by the managed device.

MIB_RowStatus_notInService Indicates that the conceptual row exists in the agent, but is unavailable for use by the managed device.

MIB_RowStatus_notReady Indicates that the conceptual row exists in the agent, one or more required columns in the row are not instantiated.

MIB_RowStatus_createAndGo Supplied by a manager wishing to create a new instance of a conceptual row and make it available for use.

MIB_RowStatus_createAndWait Supplied by a manager wishing to create a new instance of a conceptual row but not making it available for use.

MIB_RowStatus_destroy Supplied by a manager wishing to delete all of the instances associated with an existing conceptual row.

7.33 atlk/mibs/spi2uart-mibstat.h File Reference

SPI2UART Status API.

```
#include <atlk/mib_service.h>
```

Data Structures

- struct mibstat_spi2uartMib_t

  SPI2UART status.

Functions

- atlk_rc_t mibstat_get_spi2uartMib (mib_service_t ∗service, mibstat_spi2uartMib_t ∗value)

  Get SPI2UART status.
- atlk_rc_t mibstat_reset_spi2uartMib (mib_service_t ∗service)

  Reset SPI2UART status.

7.33.1 Detailed Description

SPI2UART Status API.

7.33.2 Function Documentation

atlk_rc_t mibstat_get_spi2uartMib (mib_service_t ∗ *service,* mibstat_spi2uartMib_t ∗ *value*) Get SPI2UART status.

Parameters

in	*service*	Instance of MIB service
out	*value*	SPI2UART status value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mibstat_reset_spi2uartMib (mib_service_t ∗ *service*) Reset SPI2UART status.

Parameters

in	*service*	Instance of MIB service

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.34 atlk/mibs/tc.h File Reference

Mapping of types in AUTOTALKS-TC.

Enumerations

- enum mib_ConfigSaveStatus_t {
  MIB_ConfigSaveStatus_upToDate = 0, MIB_ConfigSaveStatus_notUpToDate = 1, MIB_ConfigSaveStatus_save = 2,
  MIB_ConfigSaveStatus_saveInProgress = 3,
  MIB_ConfigSaveStatus_saveError = 4, MIB_ConfigSaveStatus_notSupported = 5 }

  Configuration save status.
- enum mib_AntennaStatus_t {
  MIB_AntennaStatus_notSupported = 0, MIB_AntennaStatus_connected = 1, MIB_AntennaStatus_notConnected = 2,
  MIB_AntennaStatus_shorted = 3,
  MIB_AntennaStatus_error = 4 }

  Antenna status.

7.34.1 Detailed Description

Mapping of types in AUTOTALKS-TC.

7.34.2 Enumeration Type Documentation

enum mib_AntennaStatus_t Antenna status.

Enumerator

MIB_AntennaStatus_notSupported	Antenna status sensing is not supported.
MIB_AntennaStatus_connected	Antenna is connected.
MIB_AntennaStatus_notConnected	Antenna is not connected.
MIB_AntennaStatus_shorted	Antenna has been electrically shorted.
MIB_AntennaStatus_error	An error occurred during antenna status sensing.

enum mib_ConfigSaveStatus_t Configuration save status.

Enumerator

MIB_ConfigSaveStatus_upToDate Saved configuration is up-to-date.
MIB_ConfigSaveStatus_notUpToDate Saved configuration is not up-to-date.
MIB_ConfigSaveStatus_save Configuration save operation is requested.
MIB_ConfigSaveStatus_saveInProgress Configuration save operation is in progress.
MIB_ConfigSaveStatus_saveError Latest save operation failed.
MIB_ConfigSaveStatus_notSupported Device doesn't support save operation.

7.35 atlk/mibs/vca-mib.h File Reference

AUTOTALKS-VCA-MIB access API.

```
   #include <atlk/mib_service.h>
#include <atlk/mibs/tc.h>
```

Macros

- #define MIB_vcaTxPeriod_MIN 10
  Minimum vcaTxPeriod value (units: milliseconds)
- #define MIB_vcaTxPeriod_MAX 1000
  Maximum vcaTxPeriod value (units: milliseconds)
- #define MIB_vcaFrameLen_MIN 40
  Minimum vcaFrameLen value (units: octets)
- #define MIB_vcaFrameLen_MAX 2304
  Maximum vcaFrameLen value (units: octets)

Enumerations

- enum mib_vcaLogMode_t
  Enumeration of vcaLogMode values.

Functions

- atlk_rc_t mib_get_vcaLogMode (mib_service_t ∗service, mib_vcaLogMode_t ∗value)
  Get vcaLogMode value.
- atlk_rc_t mib_set_vcaLogMode (mib_service_t ∗service, mib_vcaLogMode_t value)
  Set vcaLogMode value.
- atlk_rc_t mib_get_vcaTxPeriod (mib_service_t ∗service, int32_t ifIndex, uint32_t ∗value)
  Get vcaTxPeriod value (units: milliseconds).
- atlk_rc_t mib_set_vcaTxPeriod (mib_service_t ∗service, int32_t ifIndex, uint32_t value)
  Set vcaTxPeriod value (units: milliseconds).
- atlk_rc_t mib_get_vcaFrameLen (mib_service_t ∗service, int32_t ifIndex, uint32_t ∗value)
  Get vcaFrameLen value (units: octets).
- atlk_rc_t mib_set_vcaFrameLen (mib_service_t ∗service, int32_t ifIndex, uint32_t value)
  Set vcaFrameLen value (units: octets).
- atlk_rc_t mib_get_vcaTxEnabled (mib_service_t ∗service, int32_t ifIndex, int ∗value)
  Get vcaTxEnabled value.
- atlk_rc_t mib_set_vcaTxEnabled (mib_service_t ∗service, int32_t ifIndex, int value)
  Set vcaTxEnabled value.
- atlk_rc_t mib_get_vcaConfigSaveStatus (mib_service_t ∗service, mib_ConfigSaveStatus_t ∗value)
  Get vcaConfigSaveStatus value.
- atlk_rc_t mib_set_vcaConfigSaveStatus (mib_service_t ∗service, mib_ConfigSaveStatus_t value)
  Set vcaConfigSaveStatus value.

7.35.1 Detailed Description

AUTOTALKS-VCA-MIB access API. VCA (V2X Communication Analyzer) MIB.

7.35.2 Function Documentation

atlk_rc_t mib_get_vcaConfigSaveStatus (mib_service_t ∗ *service*, mib_ConfigSaveStatus_t ∗ *value*) Get vcaConfigSaveStatus value.

VCA MIB configuration save status.

Parameters

in	*service*	Instance of MIB service
out	*value*	vcaConfigSaveStatus value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_vcaFrameLen (mib_service_t ∗ *service*, int32_t *ifIndex*, uint32_t ∗ *value*) Get vcaFrameLen value (units: octets).

Length of IEEE 802.11 MSDU generated by VCA.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	vcaFrameLen value (units: octets)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_vcaLogMode (mib_service_t ∗ *service*, mib_vcaLogMode_t ∗ *value*) Get vcaLogMode value.

VCA logging modes:

off – No logging. ifHasNavFix – Log only when navigation fix is available. ifHasTrueTime – Log only when true (UTC) time is available.

Parameters

in	*service*	Instance of MIB service
out	*value*	vcaLogMode value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_vcaTxEnabled (mib_service_t ∗ *service*, int32_t *ifIndex*, int ∗ *value*) Get vcaTxEnabled value.

Whether VCA frame transmission from this MAC interface is enabled.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	vcaTxEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_vcaTxPeriod (mib_service_t ∗ *service,* int32_t *ifIndex,* uint32_t ∗ *value*) Get vcaTxPeriod value (units: milliseconds).

Time period between sequential VCA frame transmissions.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	vcaTxPeriod value (units: milliseconds)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_vcaConfigSaveStatus (mib_service_t ∗ *service,* mib_ConfigSaveStatus_t *value*) Set vcaConfigSave-Status value.

VCA MIB configuration save status.

Parameters

in	*service*	Instance of MIB service
in	*value*	vcaConfigSaveStatus value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_vcaFrameLen (mib_service_t ∗ *service,* int32_t *ifIndex,* uint32_t *value*) Set vcaFrameLen value (units: octets).

Length of IEEE 802.11 MSDU generated by VCA.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
in	*value*	vcaFrameLen value (units: octets)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_vcaLogMode (mib_service_t * *service*, mib_vcaLogMode_t *value*) Set vcaLogMode value.

VCA logging modes:

off – No logging. ifHasNavFix – Log only when navigation fix is available. ifHasTrueTime – Log only when true (UTC) time is available.

Parameters

`in`	*service*	Instance of MIB service
`in`	*value*	vcaLogMode value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_vcaTxEnabled (mib_service_t * *service*, int32_t *ifIndex*, int *value*) Set vcaTxEnabled value.

Whether VCA frame transmission from this MAC interface is enabled.

Parameters

`in`	*service*	Instance of MIB service
`in`	*ifIndex*	ifIndex value
`in`	*value*	vcaTxEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_vcaTxPeriod (mib_service_t * *service*, int32_t *ifIndex*, uint32_t *value*) Set vcaTxPeriod value (units: milliseconds).

Time period between sequential VCA frame transmissions.

Parameters

`in`	*service*	Instance of MIB service
`in`	*ifIndex*	ifIndex value
`in`	*value*	vcaTxPeriod value (units: milliseconds)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.36 atlk/mibs/wlan-mib.h File Reference

AUTOTALKS-WLAN-MIB access API.

```
   #include <atlk/mib_service.h>
#include <atlk/eui48.h>
#include <atlk/mibs/tc.h>
```

Macros

- #define MIB_wlanDefaultTxDataRate_MIN 6

  Minimum wlanDefaultTxDataRate value (units: 500 kbit/s)
- #define MIB_wlanDefaultTxDataRate_MAX 108

  Maximum wlanDefaultTxDataRate value (units: 500 kbit/s)
- #define MIB_wlanDefaultTxPower_MIN (-30)

  Minimum wlanDefaultTxPower value (units: dBm)
- #define MIB_wlanDefaultTxPower_MAX 33

  Maximum wlanDefaultTxPower value (units: dBm)
- #define MIB_wlanShortRetryLimit_MIN 1

  Minimum wlanShortRetryLimit value.
- #define MIB_wlanShortRetryLimit_MAX 255

  Maximum wlanShortRetryLimit value.
- #define MIB_wlanDefaultTxPowerDbm8_MIN (-240)

  Minimum wlanDefaultTxPowerDbm8 value (units: 1/8 dBm)
- #define MIB_wlanDefaultTxPowerDbm8_MAX 264

  Maximum wlanDefaultTxPowerDbm8 value (units: 1/8 dBm)
- #define MIB_wlanEdcaCWmin_MIN 0

  Minimum wlanEdcaCWmin value.
- #define MIB_wlanEdcaCWmin_MAX 255

  Maximum wlanEdcaCWmin value.
- #define MIB_wlanEdcaCWmax_MIN 0

  Minimum wlanEdcaCWmax value.
- #define MIB_wlanEdcaCWmax_MAX 65535

  Maximum wlanEdcaCWmax value.
- #define MIB_wlanCsIntervalA_MIN 0

  Minimum wlanCsIntervalA value (units: millisecond)
- #define MIB_wlanCsIntervalA_MAX 255

  Maximum wlanCsIntervalA value (units: millisecond)
- #define MIB_wlanCsIntervalA_DEFVAL 50

  Default wlanCsIntervalA value (units: millisecond)
- #define MIB_wlanCsIntervalB_MIN 0

  Minimum wlanCsIntervalB value (units: millisecond)
- #define MIB_wlanCsIntervalB_MAX 255

  Maximum wlanCsIntervalB value (units: millisecond)
- #define MIB_wlanCsIntervalB_DEFVAL 50

  Default wlanCsIntervalB value (units: millisecond)
- #define MIB_wlanCsSyncTolerance_MIN 0

  Minimum wlanCsSyncTolerance value (units: milliseconds)
- #define MIB_wlanCsSyncTolerance_MAX 255

  Maximum wlanCsSyncTolerance value (units: milliseconds)
- #define MIB_wlanCsSyncTolerance_DEFVAL 2

  Default wlanCsSyncTolerance value (units: milliseconds)
- #define MIB_wlanTxCsd_MIN 0

  Minimum wlanTxCsd value (units: samples)
- #define MIB_wlanTxCsd_MAX 4

  Maximum wlanTxCsd value (units: samples)
- #define MIB_wlanChannelProbingInterval_MIN 1

  Minimum wlanChannelProbingInterval value (units: milliseconds)
- #define MIB_wlanChannelProbingInterval_MAX 1000

Maximum wlanChannelProbingInterval value (units: milliseconds)

- #define MIB_wlanChannelLoadThreshold_MIN (-95)

  Minimum wlanChannelLoadThreshold value (units: dBm)
- #define MIB_wlanChannelLoadThreshold_MAX (-35)

  Maximum wlanChannelLoadThreshold value (units: dBm)
- #define MIB_wlanChannelBusyRatio_MIN 0

  Minimum wlanChannelBusyRatio value (units: %)
- #define MIB_wlanChannelBusyRatio_MAX 100

  Maximum wlanChannelBusyRatio value (units: %)
- #define MIB_wlanRficTemperature_MIN (-200)

  Minimum wlanRficTemperature value (units: degrees Celsius)
- #define MIB_wlanRficTemperature_MAX 200

  Maximum wlanRficTemperature value (units: degrees Celsius)
- #define MIB_wlanRcpiLatestFrame_MIN 0

  Minimum wlanRcpiLatestFrame value (units: dBm)
- #define MIB_wlanRcpiLatestFrame_MAX 255

  Maximum wlanRcpiLatestFrame value (units: dBm)
- #define MIB_wlanFrequency_MIN 740

  Minimum wlanFrequency value (units: MHz)
- #define MIB_wlanFrequency_MAX 5920

  Maximum wlanFrequency value (units: MHz)
- #define MIB_wlanRfFrontEndOffset_MIN 0

  Minimum wlanRfFrontEndOffset value (units: dBm)
- #define MIB_wlanRfFrontEndOffset_MAX 30

  Maximum wlanRfFrontEndOffset value (units: dBm)
- #define MIB_wlanPresetFrequency0_MIN 5180

  Minimum wlanPresetFrequency0 value (units: MHz)
- #define MIB_wlanPresetFrequency0_MAX 5930

  Maximum wlanPresetFrequency0 value (units: MHz)
- #define MIB_wlanPresetFrequency1_MIN 5180

  Minimum wlanPresetFrequency1 value (units: MHz)
- #define MIB_wlanPresetFrequency1_MAX 5930

  Maximum wlanPresetFrequency1 value (units: MHz)
- #define MIB_wlanTxIqImbalanceAmplitude_MIN (-60)

  Minimum wlanTxIqImbalanceAmplitude value (units: 0.1 dB)
- #define MIB_wlanTxIqImbalanceAmplitude_MAX 60

  Maximum wlanTxIqImbalanceAmplitude value (units: 0.1 dB)
- #define MIB_wlanTxIqImbalancePhase_MIN (-100)

  Minimum wlanTxIqImbalancePhase value (units: 0.1 degree)
- #define MIB_wlanTxIqImbalancePhase_MAX 100

  Maximum wlanTxIqImbalancePhase value (units: 0.1 degree)
- #define MIB_wlanPantLutIndex_MIN 0

  Minimum wlanPantLutIndex value.
- #define MIB_wlanPantLutIndex_MAX 4

  Maximum wlanPantLutIndex value.
- #define MIB_wlanRxSampleGainLow_MIN (-1280)

  Minimum wlanRxSampleGainLow value (units: 0.1 dB)
- #define MIB_wlanRxSampleGainLow_MAX 1270

  Maximum wlanRxSampleGainLow value (units: 0.1 dB)
- #define MIB_wlanRxSampleGainMid_MIN (-1280)

  Minimum wlanRxSampleGainMid value (units: 0.1 dB)

- #define MIB_wlanRxSampleGainMid_MAX 1270

  Maximum wlanRxSampleGainMid value (units: 0.1 dB)
- #define MIB_wlanRxSampleGainHigh_MIN (-1280)

  Minimum wlanRxSampleGainHigh value (units: 0.1 dB)
- #define MIB_wlanRxSampleGainHigh_MAX 1270

  Maximum wlanRxSampleGainHigh value (units: 0.1 dB)
- #define MIB_wlanGrfiSignalDelayResolution_MIN 0

  Minimum wlanGrfiSignalDelayResolution value (units: 0.1 usec)
- #define MIB_wlanGrfiSignalDelayResolution_MAX 256

  Maximum wlanGrfiSignalDelayResolution value (units: 0.1 usec)
- #define MIB_wlanRxIqImbalanceAmplitude_MIN (-60)

  Minimum wlanRxIqImbalanceAmplitude value (units: 0.1 dB)
- #define MIB_wlanRxIqImbalanceAmplitude_MAX 60

  Maximum wlanRxIqImbalanceAmplitude value (units: 0.1 dB)
- #define MIB_wlanRxIqImbalancePhase_MIN (-100)

  Minimum wlanRxIqImbalancePhase value (units: 0.1 degree)
- #define MIB_wlanRxIqImbalancePhase_MAX 100

  Maximum wlanRxIqImbalancePhase value (units: 0.1 degree)
- #define MIB_wlanLoLeakage_MIN 0

  Minimum wlanLoLeakage value.
- #define MIB_wlanLoLeakage_MAX 65535

  Maximum wlanLoLeakage value.
- #define MIB_wlanPantLutDbm8_MIN (-240)

  Minimum wlanPantLutDbm8 value.
- #define MIB_wlanPantLutDbm8_MAX 320

  Maximum wlanPantLutDbm8 value.

Enumerations

- enum mib_wlanDcocStatus_t

  Enumeration of wlanDcocStatus values.
- enum mib_wlanPhyOFDMChannelWidth_t

  Enumeration of wlanPhyOFDMChannelWidth values.
- enum mib_wlanRfTestMode_t

  Enumeration of wlanRfTestMode values.

Functions

- atlk_rc_t mib_get_wlanConfigSaveStatus (mib_service_t ∗service, mib_ConfigSaveStatus_t ∗value)

  Get wlanConfigSaveStatus value.
- atlk_rc_t mib_set_wlanConfigSaveStatus (mib_service_t ∗service, mib_ConfigSaveStatus_t value)

  Set wlanConfigSaveStatus value.
- atlk_rc_t mib_get_wlanDefaultTxDataRate (mib_service_t ∗service, int32_t ifIndex, int32_t ∗value)

  Get wlanDefaultTxDataRate value (units: 500 kbit/s).
- atlk_rc_t mib_set_wlanDefaultTxDataRate (mib_service_t ∗service, int32_t ifIndex, int32_t value)

  Set wlanDefaultTxDataRate value (units: 500 kbit/s).
- atlk_rc_t mib_get_wlanBssid (mib_service_t ∗service, int32_t ifIndex, eui48_t ∗value)

  Get wlanBssid value.
- atlk_rc_t mib_set_wlanBssid (mib_service_t ∗service, int32_t ifIndex, eui48_t value)

  Set wlanBssid value.
- atlk_rc_t mib_get_wlanDefaultTxPower (mib_service_t ∗service, int32_t ifIndex, int32_t ∗value)

  Get wlanDefaultTxPower value (units: dBm).

- atlk_rc_t mib_set_wlanDefaultTxPower (mib_service_t ∗service, int32_t ifIndex, int32_t value)
  Set wlanDefaultTxPower value (units: dBm).
- atlk_rc_t mib_get_wlanRandomBackoffEnabled (mib_service_t ∗service, int32_t ifIndex, int ∗value)
  Get wlanRandomBackoffEnabled value.
- atlk_rc_t mib_set_wlanRandomBackoffEnabled (mib_service_t ∗service, int32_t ifIndex, int value)
  Set wlanRandomBackoffEnabled value.
- atlk_rc_t mib_get_wlanMacAddress (mib_service_t ∗service, int32_t ifIndex, eui48_t ∗value)
  Get wlanMacAddress value.
- atlk_rc_t mib_set_wlanMacAddress (mib_service_t ∗service, int32_t ifIndex, eui48_t value)
  Set wlanMacAddress value.
- atlk_rc_t mib_get_wlanTxSaOverrideEnabled (mib_service_t ∗service, int32_t ifIndex, int ∗value)
  Get wlanTxSaOverrideEnabled value.
- atlk_rc_t mib_set_wlanTxSaOverrideEnabled (mib_service_t ∗service, int32_t ifIndex, int value)
  Set wlanTxSaOverrideEnabled value.
- atlk_rc_t mib_get_wlanRxUcastDaFilterEnabled (mib_service_t ∗service, int32_t ifIndex, int ∗value)
  Get wlanRxUcastDaFilterEnabled value.
- atlk_rc_t mib_set_wlanRxUcastDaFilterEnabled (mib_service_t ∗service, int32_t ifIndex, int value)
  Set wlanRxUcastDaFilterEnabled value.
- atlk_rc_t mib_get_wlanShortRetryLimit (mib_service_t ∗service, int32_t ifIndex, int32_t ∗value)
  Get wlanShortRetryLimit value.
- atlk_rc_t mib_set_wlanShortRetryLimit (mib_service_t ∗service, int32_t ifIndex, int32_t value)
  Set wlanShortRetryLimit value.
- atlk_rc_t mib_get_wlanDefaultTxPowerDbm8 (mib_service_t ∗service, int32_t ifIndex, int32_t ∗value)
  Get wlanDefaultTxPowerDbm8 value (units: 1/8 dBm).
- atlk_rc_t mib_set_wlanDefaultTxPowerDbm8 (mib_service_t ∗service, int32_t ifIndex, int32_t value)
  Set wlanDefaultTxPowerDbm8 value (units: 1/8 dBm).
- atlk_rc_t mib_get_wlanQosDataEnabled (mib_service_t ∗service, int32_t ifIndex, int ∗value)
  Get wlanQosDataEnabled value.
- atlk_rc_t mib_set_wlanQosDataEnabled (mib_service_t ∗service, int32_t ifIndex, int value)
  Set wlanQosDataEnabled value.
- atlk_rc_t mib_get_wlanFrameTxCnt (mib_service_t ∗service, int32_t ifIndex, uint32_t ∗value)
  Get wlanFrameTxCnt value.
- atlk_rc_t mib_get_wlanFrameRxCnt (mib_service_t ∗service, int32_t ifIndex, uint32_t ∗value)
  Get wlanFrameRxCnt value.
- atlk_rc_t mib_get_wlanTxFailCnt (mib_service_t ∗service, int32_t ifIndex, uint32_t ∗value)
  Get wlanTxFailCnt value.
- atlk_rc_t mib_get_wlanTxAllocFailCnt (mib_service_t ∗service, int32_t ifIndex, uint32_t ∗value)
  Get wlanTxAllocFailCnt value.
- atlk_rc_t mib_get_wlanTxQueueFailCnt (mib_service_t ∗service, int32_t ifIndex, uint32_t ∗value)
  Get wlanTxQueueFailCnt value.
- atlk_rc_t mib_get_wlanRxFailCnt (mib_service_t ∗service, int32_t ifIndex, uint32_t ∗value)
  Get wlanRxFailCnt value.
- atlk_rc_t mib_get_wlanRxAllocFailCnt (mib_service_t ∗service, int32_t ifIndex, uint32_t ∗value)
  Get wlanRxAllocFailCnt value.
- atlk_rc_t mib_get_wlanRxQueueFailCnt (mib_service_t ∗service, int32_t ifIndex, uint32_t ∗value)
  Get wlanRxQueueFailCnt value.
- atlk_rc_t mib_get_wlanRxCrcFailCnt (mib_service_t ∗service, int32_t ifIndex, uint32_t ∗value)
  Get wlanRxCrcFailCnt value.
- atlk_rc_t mib_get_wlanRxDuplicateFrameFilteringEnabled (mib_service_t ∗service, int ∗value)
  Get wlanRxDuplicateFrameFilteringEnabled value.
- atlk_rc_t mib_set_wlanRxDuplicateFrameFilteringEnabled (mib_service_t ∗service, int value)

Set wlanRxDuplicateFrameFilteringEnabled value.

- atlk_rc_t mib_get_wlanEdcaCWmin (mib_service_t ∗service, int32_t wlanEdcaIndex, uint32_t ∗value)

  Get wlanEdcaCWmin value.
- atlk_rc_t mib_set_wlanEdcaCWmin (mib_service_t ∗service, int32_t wlanEdcaIndex, uint32_t value)

  Set wlanEdcaCWmin value.
- atlk_rc_t mib_get_wlanEdcaCWmax (mib_service_t ∗service, int32_t wlanEdcaIndex, uint32_t ∗value)

  Get wlanEdcaCWmax value.
- atlk_rc_t mib_set_wlanEdcaCWmax (mib_service_t ∗service, int32_t wlanEdcaIndex, uint32_t value)

  Set wlanEdcaCWmax value.
- atlk_rc_t mib_get_wlanCsIntervalA (mib_service_t ∗service, int32_t ifIndex, int32_t ∗value)

  Get wlanCsIntervalA value (units: millisecond).
- atlk_rc_t mib_set_wlanCsIntervalA (mib_service_t ∗service, int32_t ifIndex, int32_t value)

  Set wlanCsIntervalA value (units: millisecond).
- atlk_rc_t mib_get_wlanCsIntervalB (mib_service_t ∗service, int32_t ifIndex, int32_t ∗value)

  Get wlanCsIntervalB value (units: millisecond).
- atlk_rc_t mib_set_wlanCsIntervalB (mib_service_t ∗service, int32_t ifIndex, int32_t value)

  Set wlanCsIntervalB value (units: millisecond).
- atlk_rc_t mib_get_wlanCsSyncTolerance (mib_service_t ∗service, int32_t ifIndex, int32_t ∗value)

  Get wlanCsSyncTolerance value (units: milliseconds).
- atlk_rc_t mib_set_wlanCsSyncTolerance (mib_service_t ∗service, int32_t ifIndex, int32_t value)

  Set wlanCsSyncTolerance value (units: milliseconds).
- atlk_rc_t mib_get_wlanEpdEnabled (mib_service_t ∗service, int ∗value)

  Get wlanEpdEnabled value.
- atlk_rc_t mib_set_wlanEpdEnabled (mib_service_t ∗service, int value)

  Set wlanEpdEnabled value.
- atlk_rc_t mib_get_wlanTxDiversityEnabled (mib_service_t ∗service, int ∗value)

  Get wlanTxDiversityEnabled value.
- atlk_rc_t mib_set_wlanTxDiversityEnabled (mib_service_t ∗service, int value)

  Set wlanTxDiversityEnabled value.
- atlk_rc_t mib_get_wlanTxCsd (mib_service_t ∗service, int32_t ∗value)

  Get wlanTxCsd value (units: samples).
- atlk_rc_t mib_set_wlanTxCsd (mib_service_t ∗service, int32_t value)

  Set wlanTxCsd value (units: samples).
- atlk_rc_t mib_get_wlanRxDiversityEnabled (mib_service_t ∗service, int ∗value)

  Get wlanRxDiversityEnabled value.
- atlk_rc_t mib_set_wlanRxDiversityEnabled (mib_service_t ∗service, int value)

  Set wlanRxDiversityEnabled value.
- atlk_rc_t mib_get_wlanRxDiversityCnt (mib_service_t ∗service, uint32_t ∗value)

  Get wlanRxDiversityCnt value.
- atlk_rc_t mib_get_wlanChannelProbingInterval (mib_service_t ∗service, int32_t ifIndex, int32_t ∗value)

  Get wlanChannelProbingInterval value (units: milliseconds).
- atlk_rc_t mib_set_wlanChannelProbingInterval (mib_service_t ∗service, int32_t ifIndex, int32_t value)

  Set wlanChannelProbingInterval value (units: milliseconds).
- atlk_rc_t mib_get_wlanChannelLoadThreshold (mib_service_t ∗service, int32_t ifIndex, int32_t ∗value)

  Get wlanChannelLoadThreshold value (units: dBm).
- atlk_rc_t mib_set_wlanChannelLoadThreshold (mib_service_t ∗service, int32_t ifIndex, int32_t value)

  Set wlanChannelLoadThreshold value (units: dBm).
- atlk_rc_t mib_get_wlanChannelBusyRatio (mib_service_t ∗service, int32_t ifIndex, int32_t ∗value)

  Get wlanChannelBusyRatio value (units: %).
- atlk_rc_t mib_get_wlanPhyHeaderErrCnt (mib_service_t ∗service, int32_t ifIndex, uint32_t ∗value)

  Get wlanPhyHeaderErrCnt value.

- atlk_rc_t mib_get_wlanDcocEnabled (mib_service_t ∗service, int32_t wlanRfIndex, int ∗value)
  Get wlanDcocEnabled value.
- atlk_rc_t mib_set_wlanDcocEnabled (mib_service_t ∗service, int32_t wlanRfIndex, int value)
  Set wlanDcocEnabled value.
- atlk_rc_t mib_get_wlanRssiLatestFrame (mib_service_t ∗service, int32_t wlanRfIndex, int32_t ∗value)
  Get wlanRssiLatestFrame value (units: dBm).
- atlk_rc_t mib_get_wlanRficTemperature (mib_service_t ∗service, int32_t wlanRfIndex, int32_t ∗value)
  Get wlanRficTemperature value (units: degrees Celsius).
- atlk_rc_t mib_get_wlanRcpiLatestFrame (mib_service_t ∗service, int32_t wlanRfIndex, int32_t ∗value)
  Get wlanRcpiLatestFrame value (units: dBm).
- atlk_rc_t mib_get_wlanAntennaStatus (mib_service_t ∗service, int32_t wlanRfIndex, mib_AntennaStatus_t ∗value)
  Get wlanAntennaStatus value.
- atlk_rc_t mib_get_wlanFrequency (mib_service_t ∗service, int32_t wlanRfIndex, int32_t ∗value)
  Get wlanFrequency value (units: MHz).
- atlk_rc_t mib_set_wlanFrequency (mib_service_t ∗service, int32_t wlanRfIndex, int32_t value)
  Set wlanFrequency value (units: MHz).
- atlk_rc_t mib_get_wlanDcocStatus (mib_service_t ∗service, int32_t wlanRfIndex, mib_wlanDcocStatus_t ∗value)
  Get wlanDcocStatus value.
- atlk_rc_t mib_get_wlanRfFrontEndConnected (mib_service_t ∗service, int32_t wlanRfIndex, int ∗value)
  Get wlanRfFrontEndConnected value.
- atlk_rc_t mib_set_wlanRfFrontEndConnected (mib_service_t ∗service, int32_t wlanRfIndex, int value)
  Set wlanRfFrontEndConnected value.
- atlk_rc_t mib_get_wlanRfEnabled (mib_service_t ∗service, int32_t wlanRfIndex, int ∗value)
  Get wlanRfEnabled value.
- atlk_rc_t mib_set_wlanRfEnabled (mib_service_t ∗service, int32_t wlanRfIndex, int value)
  Set wlanRfEnabled value.
- atlk_rc_t mib_get_wlanRfFrontEndOffset (mib_service_t ∗service, int32_t wlanRfIndex, int32_t ∗value)
  Get wlanRfFrontEndOffset value (units: dBm).
- atlk_rc_t mib_set_wlanRfFrontEndOffset (mib_service_t ∗service, int32_t wlanRfIndex, int32_t value)
  Set wlanRfFrontEndOffset value (units: dBm).
- atlk_rc_t mib_get_wlanPhyOFDMChannelWidth (mib_service_t ∗service, int32_t wlanRfIndex, mib_wlanPhyOFDM-ChannelWidth_t ∗value)
  Get wlanPhyOFDMChannelWidth value.
- atlk_rc_t mib_set_wlanPhyOFDMChannelWidth (mib_service_t ∗service, int32_t wlanRfIndex, mib_wlanPhyOFDM-ChannelWidth_t value)
  Set wlanPhyOFDMChannelWidth value.
- atlk_rc_t mib_get_wlanPresetFrequency0 (mib_service_t ∗service, int32_t wlanRfIndex, int32_t ∗value)
- atlk_rc_t mib_set_wlanPresetFrequency0 (mib_service_t ∗service, int32_t wlanRfIndex, int32_t value)
- atlk_rc_t mib_get_wlanPresetFrequency1 (mib_service_t ∗service, int32_t wlanRfIndex, int32_t ∗value)
- atlk_rc_t mib_set_wlanPresetFrequency1 (mib_service_t ∗service, int32_t wlanRfIndex, int32_t value)
- atlk_rc_t mib_get_wlanRfTestMode (mib_service_t ∗service, int32_t wlanRfIndex, mib_wlanRfTestMode_t ∗value)
  Get wlanRfTestMode value.
- atlk_rc_t mib_set_wlanRfTestMode (mib_service_t ∗service, int32_t wlanRfIndex, mib_wlanRfTestMode_t value)
  Set wlanRfTestMode value.
- atlk_rc_t mib_get_wlanTxIqImbalanceAmplitude (mib_service_t ∗service, int32_t wlanRfIndex, int32_t ∗value)
  Get wlanTxIqImbalanceAmplitude value (units: 0.1 dB).
- atlk_rc_t mib_set_wlanTxIqImbalanceAmplitude (mib_service_t ∗service, int32_t wlanRfIndex, int32_t value)
  Set wlanTxIqImbalanceAmplitude value (units: 0.1 dB).
- atlk_rc_t mib_get_wlanTxIqImbalancePhase (mib_service_t ∗service, int32_t wlanRfIndex, int32_t ∗value)
  Get wlanTxIqImbalancePhase value (units: 0.1 degree).
- atlk_rc_t mib_set_wlanTxIqImbalancePhase (mib_service_t ∗service, int32_t wlanRfIndex, int32_t value)

Set wlanTxIqImbalancePhase value (units: 0.1 degree).
- atlk_rc_t mib_get_wlanPantLutIndex (mib_service_t ∗service, int32_t wlanRfIndex, int32_t ∗value)
  Get wlanPantLutIndex value.
- atlk_rc_t mib_set_wlanPantLutIndex (mib_service_t ∗service, int32_t wlanRfIndex, int32_t value)
  Set wlanPantLutIndex value.
- atlk_rc_t mib_get_wlanTssiDetectorReading (mib_service_t ∗service, int32_t wlanRfIndex, int32_t ∗value)
  Get wlanTssiDetectorReading value.
- atlk_rc_t mib_get_wlanRfCalibrationRequired (mib_service_t ∗service, int32_t wlanRfIndex, int ∗value)
  Get wlanRfCalibrationRequired value.
- atlk_rc_t mib_set_wlanRfCalibrationRequired (mib_service_t ∗service, int32_t wlanRfIndex, int value)
  Set wlanRfCalibrationRequired value.
- atlk_rc_t mib_get_wlanTssiInterval (mib_service_t ∗service, int32_t wlanRfIndex, int32_t ∗value)
  Get wlanTssiInterval value (units: sec).
- atlk_rc_t mib_set_wlanTssiInterval (mib_service_t ∗service, int32_t wlanRfIndex, int32_t value)
  Set wlanTssiInterval value (units: sec).
- atlk_rc_t mib_get_wlanRxSampleGainLow (mib_service_t ∗service, int32_t wlanRfIndex, int32_t ∗value)
  Get wlanRxSampleGainLow value (units: 0.1 dB).
- atlk_rc_t mib_set_wlanRxSampleGainLow (mib_service_t ∗service, int32_t wlanRfIndex, int32_t value)
  Set wlanRxSampleGainLow value (units: 0.1 dB).
- atlk_rc_t mib_get_wlanRxSampleGainMid (mib_service_t ∗service, int32_t wlanRfIndex, int32_t ∗value)
  Get wlanRxSampleGainMid value (units: 0.1 dB).
- atlk_rc_t mib_set_wlanRxSampleGainMid (mib_service_t ∗service, int32_t wlanRfIndex, int32_t value)
  Set wlanRxSampleGainMid value (units: 0.1 dB).
- atlk_rc_t mib_get_wlanRxSampleGainHigh (mib_service_t ∗service, int32_t wlanRfIndex, int32_t ∗value)
  Get wlanRxSampleGainHigh value (units: 0.1 dB).
- atlk_rc_t mib_set_wlanRxSampleGainHigh (mib_service_t ∗service, int32_t wlanRfIndex, int32_t value)
  Set wlanRxSampleGainHigh value (units: 0.1 dB).
- atlk_rc_t mib_get_wlanGrfiSignalDelayResolution (mib_service_t ∗service, int32_t wlanRfIndex, int32_t ∗value)
  Get wlanGrfiSignalDelayResolution value (units: 0.1 usec).
- atlk_rc_t mib_set_wlanGrfiSignalDelayResolution (mib_service_t ∗service, int32_t wlanRfIndex, int32_t value)
  Set wlanGrfiSignalDelayResolution value (units: 0.1 usec).
- atlk_rc_t mib_get_wlanRxIqImbalanceAmplitude (mib_service_t ∗service, int32_t wlanRfIndex, int32_t ∗value)
  Get wlanRxIqImbalanceAmplitude value (units: 0.1 dB).
- atlk_rc_t mib_set_wlanRxIqImbalanceAmplitude (mib_service_t ∗service, int32_t wlanRfIndex, int32_t value)
  Set wlanRxIqImbalanceAmplitude value (units: 0.1 dB).
- atlk_rc_t mib_get_wlanRxIqImbalancePhase (mib_service_t ∗service, int32_t wlanRfIndex, int32_t ∗value)
  Get wlanRxIqImbalancePhase value (units: 0.1 degree).
- atlk_rc_t mib_set_wlanRxIqImbalancePhase (mib_service_t ∗service, int32_t wlanRfIndex, int32_t value)
  Set wlanRxIqImbalancePhase value (units: 0.1 degree).
- atlk_rc_t mib_get_wlanPantLut (mib_service_t ∗service, int32_t wlanRfIndex, char ∗value, size_t ∗size)
- atlk_rc_t mib_set_wlanPantLut (mib_service_t ∗service, int32_t wlanRfIndex, const char ∗value, size_t size)
- atlk_rc_t mib_get_wlanLoLeakage (mib_service_t ∗service, int32_t wlanLoLeakageIndex, int32_t ∗value)
  Get wlanLoLeakage value.
- atlk_rc_t mib_set_wlanLoLeakage (mib_service_t ∗service, int32_t wlanLoLeakageIndex, int32_t value)
  Set wlanLoLeakage value.
- atlk_rc_t mib_get_wlanPantLutDbm8 (mib_service_t ∗service, int32_t wlanPantLutDbm8Index, int32_t ∗value)
  Get wlanPantLutDbm8 value.
- atlk_rc_t mib_set_wlanPantLutDbm8 (mib_service_t ∗service, int32_t wlanPantLutDbm8Index, int32_t value)
  Set wlanPantLutDbm8 value.

7.36.1 Detailed Description

AUTOTALKS-WLAN-MIB access API. CRATON WLAN MIB definition.

7.36.2 Function Documentation

atlk_rc_t mib_get_wlanAntennaStatus (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, mib_AntennaStatus_t ∗ *value*) Get wlanAntennaStatus value.

Current status of WLAN (DSRC) antenna.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanAntennaStatus value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanBssid (mib_service_t ∗ *service*, int32_t *ifIndex*, eui48_t ∗ *value*) Get wlanBssid value.

802.11 BSSID address.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanBssid value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanChannelBusyRatio (mib_service_t ∗ *service*, int32_t *ifIndex*, int32_t ∗ *value*) Get wlanChannelBusyRatio value (units: %).

The percentage of time during which the channel was busy in the last probing interval.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanChannelBusyRatio value (units: %)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanChannelLoadThreshold (mib_service_t ∗ *service*, int32_t *ifIndex*, int32_t ∗ *value*) Get wlanChannelLoadThreshold value (units: dBm).

Threshold of received signal strength above which the channel will be considered busy.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanChannelLoadThreshold value (units: dBm)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanChannelProbingInterval (mib_service_t * *service*, int32_t *ifIndex*, int32_t * *value*) Get wlanChannelProbingInterval value (units: milliseconds).

Channel load probing interval.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanChannelProbingInterval value (units: milliseconds)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanConfigSaveStatus (mib_service_t * *service*, mib_ConfigSaveStatus_t * *value*) Get wlanConfigSaveStatus value.

WLAN configuration save status.

Parameters

in	*service*	Instance of MIB service
out	*value*	wlanConfigSaveStatus value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanCsIntervalA (mib_service_t * *service*, int32_t *ifIndex*, int32_t * *value*) Get wlanCsIntervalA value (units: millisecond).

Channel A interval.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanCsIntervalA value (units: millisecond)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanCsIntervalB (mib_service_t ∗ *service*, int32_t *ifIndex*, int32_t ∗ *value*) Get wlanCsIntervalB value (units: millisecond).

Channel B interval.

Parameters

`in`	*service*	Instance of MIB service
`in`	*ifIndex*	ifIndex value
`out`	*value*	wlanCsIntervalB value (units: millisecond)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanCsSyncTolerance (mib_service_t ∗ *service*, int32_t *ifIndex*, int32_t ∗ *value*) Get wlanCsSyncTolerance value (units: milliseconds).

This attribute is equivalent to SyncTolerance as defined in IEEE 1609.4-2010 clause 6.2.5.

Parameters

`in`	*service*	Instance of MIB service
`in`	*ifIndex*	ifIndex value
`out`	*value*	wlanCsSyncTolerance value (units: milliseconds)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanDcocEnabled (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int ∗ *value*) Get wlanDcocEnabled value.

Whether periodic DCOC is enabled.

Parameters

`in`	*service*	Instance of MIB service
`in`	*wlanRfIndex*	wlanRfIndex value
`out`	*value*	wlanDcocEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanDcocStatus (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, mib_wlanDcocStatus_t ∗ *value*) Get wlanDcocStatus value.

Indicates status of DCOC (DC Offset Cancellation) process.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanDcocStatus value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanDefaultTxDataRate (mib_service_t ∗ *service*, int32_t *ifIndex*, int32_t ∗ *value*) Get wlanDefaultTxDataRate value (units: 500 kbit/s).

Default transmission data rate.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanDefaultTxDataRate value (units: 500 kbit/s)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanDefaultTxPower (mib_service_t ∗ *service*, int32_t *ifIndex*, int32_t ∗ *value*) Get wlanDefaultTxPower value (units: dBm).

Default transmission power.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanDefaultTxPower value (units: dBm)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanDefaultTxPowerDbm8 (mib_service_t ∗ *service*, int32_t *ifIndex*, int32_t ∗ *value*) Get wlanDefaultTxPowerDbm8 value (units: 1/8 dBm).

Default transmission power in 1/8 dBm.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanDefaultTxPowerDbm8 value (units: 1/8 dBm)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanEdcaCWmax (mib_service_t ∗ *service*, int32_t *wlanEdcaIndex*, uint32_t ∗ *value*) Get wlanEdcaCWmax value.

The maximum size of the window that is used for generating a random number for the backoff.

The value of this attribute is such that it could always be expressed in the form of 2∗∗X - 1, where X is an integer.

Parameters

in	*service*	Instance of MIB service
in	*wlanEdcaIndex*	wlanEdcaIndex value
out	*value*	wlanEdcaCWmax value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/mibs/mibs-edca-example.c.

atlk_rc_t mib_get_wlanEdcaCWmin (mib_service_t ∗ *service*, int32_t *wlanEdcaIndex*, uint32_t ∗ *value*) Get wlanEdcaCWmin value.

The minimum size of the window that is used for generating a random number for the backoff.

The value of this attribute is such that it could always be expressed in the form of 2∗∗X - 1, where X is an integer.

Parameters

in	*service*	Instance of MIB service
in	*wlanEdcaIndex*	wlanEdcaIndex value
out	*value*	wlanEdcaCWmin value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanEpdEnabled (mib_service_t ∗ *service*, int ∗ *value*) Get wlanEpdEnabled value.

Whether IEEE Std 802-2014 EtherType Protocol Discrimination (EPD) is used in the LLC sublayer.

Parameters

in	*service*	Instance of MIB service
out	*value*	wlanEpdEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanFrameRxCnt (mib_service_t ∗ *service,* int32_t *ifIndex,* uint32_t ∗ *value*) Get wlanFrameRxCnt value.

This counter shall be incremented for each correctly received frame.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanFrameRxCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanFrameTxCnt (mib_service_t ∗ *service,* int32_t *ifIndex,* uint32_t ∗ *value*) Get wlanFrameTxCnt value.

This counter shall be incremented for each transmitted frame.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanFrameTxCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanFrequency (mib_service_t ∗ *service,* int32_t *wlanRfIndex,* int32_t ∗ *value*) Get wlanFrequency value (units: MHz).

Current frequency.

Should not be used when channel switching is enabled.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanFrequency value (units: MHz)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanGrfiSignalDelayResolution (mib_service_t ∗ *service,* int32_t *wlanRfIndex,* int32_t ∗ *value*) Get wlanGrfiSignalDelayResolution value (units: 0.1 usec).

Timing resolution of the delay between transmitted packets to PA and T/R RF switch.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanGrfiSignalDelayResolution value (units: 0.1 usec)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanLoLeakage (mib_service_t * *service*, int32_t *wlanLoLeakageIndex*, int32_t * *value*) Get wlanLoLeakage value.

LO leakage cancellation per gain.

Parameters

in	*wlanLoLeakage-Index*	wlanLoLeakageIndex value
out	*value*	wlanLoLeakage value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanMacAddress (mib_service_t * *service*, int32_t *ifIndex*, eui48_t * *value*) Get wlanMacAddress value.

802.11 MAC Address.

Please note that MAC frame queues are not flushed when the address is changed, meaning that the previous address value may appear as outgoing frame source address or incoming destination address (in the case of unicast frames) some time after the change.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanMacAddress value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanPantLut (mib_service_t * *service*, int32_t *wlanRfIndex*, char * *value*, size_t * *size*) Lookup table (LUT) for conversion of power detector output into 1 dBm.

Parameters

in	*service*	Instance of MIB service

in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanPantLut array
in,out	*size*	Maximum (in) and actual (out) size of wlanPantLut array

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanPantLutDbm8 (mib_service_t ∗ *service*, int32_t *wlanPantLutDbm8Index*, int32_t ∗ *value*) Get wlanPantLutDbm8 value.

Conversion factor of power detector output into 1/8 dBm LUT.

Parameters

in	*service*	Instance of MIB service
in	*wlanPantLut-Dbm8Index*	wlanPantLutDbm8Index value
out	*value*	wlanPantLutDbm8 value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanPantLutIndex (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int32_t ∗ *value*) Get wlanPantLutIndex value.

Which power antenna LUT entry is used.

Index 0 denotes the defaut power antenna LUT. Index 1-4 denotes one of four power antenna LUTs defined at wlanPantLutDbm8Table.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanPantLutIndex value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanPhyHeaderErrCnt (mib_service_t ∗ *service*, int32_t *ifIndex*, uint32_t ∗ *value*) Get wlanPhyHeaderErrCnt value.

This counter shall be incremented for each error in PHY header

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value

out	*value*	wlanPhyHeaderErrCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanPhyOFDMChannelWidth (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, mib_wlanPhyOFDMChannelWidth_t ∗ *value*) Get wlanPhyOFDMChannelWidth value.

Current PHY OFDM channel width.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanPhyOFDMChannelWidth value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanPresetFrequency0 (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int32_t ∗ *value*) Preset frequency 0.

Should not be used when channel switching is enabled.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanPresetFrequency0 value (units: MHz)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanPresetFrequency1 (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int32_t ∗ *value*) Preset frequency 1.

Should not be used when channel switching is enabled.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanPresetFrequency1 value (units: MHz)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanQosDataEnabled (mib_service_t ∗ *service*, int32_t *ifIndex*, int ∗ *value*) Get wlanQosDataEnabled value.

Whether 802.11 QoS data is enabled.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanQosDataEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRandomBackoffEnabled (mib_service_t ∗ *service*, int32_t *ifIndex*, int ∗ *value*) Get wlanRandomBackoffEnabled value.

Whether MAC transmission random backoff is enabled.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanRandomBackoffEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRcpiLatestFrame (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int32_t ∗ *value*) Get wlanRcpiLatestFrame value (units: dBm).

RCPI of latest frame received at PHY.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanRcpiLatestFrame value (units: dBm)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRfCalibrationRequired (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int ∗ *value*) Get wlanRfCalibrationRequired value.

Whether calibration is required on next system boot.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanRfCalibrationRequired value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRfEnabled (mib_service_t * *service*, int32_t *wlanRfIndex*, int * *value*) Get wlanRfEnabled value.

Whether RF interface is enabled. Once an interface has been disabled, it can only be re-enabled by rebooting the unit.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanRfEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRfFrontEndConnected (mib_service_t * *service*, int32_t *wlanRfIndex*, int * *value*) Get wlanRfFrontEndConnected value.

Whether an external RF front-end is connected.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanRfFrontEndConnected value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRfFrontEndOffset (mib_service_t * *service*, int32_t *wlanRfIndex*, int32_t * *value*) Get wlanRfFrontEndOffset value (units: dBm).

Attenuation of the first transmitted frame's output power. Relevant only when a RF front-end is used.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanRfFrontEndOffset value (units: dBm)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRficTemperature (mib_service_t * *service*, int32_t *wlanRfIndex*, int32_t * *value*) Get wlanRficTemperature value (units: degrees Celsius).

RFIC temperature.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanRficTemperature value (units: degrees Celsius)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRfTestMode (mib_service_t * *service*, int32_t *wlanRfIndex*, mib_wlanRfTestMode_t * *value*) Get wlanRfTestMode value.

RF interface test mode.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanRfTestMode value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRssiLatestFrame (mib_service_t * *service*, int32_t *wlanRfIndex*, int32_t * *value*) Get wlanRssiLatestFrame value (units: dBm).

RSSI of latest frame received at PHY.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanRssiLatestFrame value (units: dBm)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRxAllocFailCnt (mib_service_t * *service*, int32_t *ifIndex*, uint32_t * *value*) Get wlanRxAllocFailCnt value.

This counter shall be incremented for each memory allocation failure during frame reception.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanRxAllocFailCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRxCrcFailCnt (mib_service_t * *service*, int32_t *ifIndex*, uint32_t * *value*) Get wlanRxCrcFailCnt value.
This counter shall be incremented for each CRC failure during frame reception.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanRxCrcFailCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRxDiversityCnt (mib_service_t * *service*, uint32_t * *value*) Get wlanRxDiversityCnt value.
RX diversity counter.

Parameters

in	*service*	Instance of MIB service
out	*value*	wlanRxDiversityCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRxDiversityEnabled (mib_service_t * *service*, int * *value*) Get wlanRxDiversityEnabled value.
Whether RX diversity is enabled.

Parameters

in	*service*	Instance of MIB service
out	*value*	wlanRxDiversityEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRxDuplicateFrameFilteringEnabled (mib_service_t * *service*, int * *value*) Get wlanRxDuplicateFrame-FilteringEnabled value.
Whether MAC duplicate frame filtering is enabled in RX diversity mode.

Parameters

in	*service*	Instance of MIB service
out	*value*	wlanRxDuplicateFrameFilteringEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRxFailCnt (mib_service_t ∗ *service*, int32_t *ifIndex*, uint32_t ∗ *value*) Get wlanRxFailCnt value.

This counter shall be incremented for each failure during frame reception (including allocation, queuing, CRC and other failures).

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanRxFailCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRxIqImbalanceAmplitude (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int32_t ∗ *value*) Get wlanRxIqImbalanceAmplitude value (units: 0.1 dB).

Received signal I/Q imbalance amplitude correction factor.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanRxIqImbalanceAmplitude value (units: 0.1 dB)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRxIqImbalancePhase (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int32_t ∗ *value*) Get wlanRxIqImbalancePhase value (units: 0.1 degree).

Received signal I/Q imbalance phase correction factor.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanRxIqImbalancePhase value (units: 0.1 degree)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRxQueueFailCnt (mib_service_t ∗ *service,* int32_t *ifIndex,* uint32_t ∗ *value*) Get wlanRxQueueFailCnt value.

This counter shall be incremented for each queuing failure during frame reception.

Parameters

`in`	*service*	Instance of MIB service
`in`	*ifIndex*	ifIndex value
`out`	*value*	wlanRxQueueFailCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRxSampleGainHigh (mib_service_t ∗ *service,* int32_t *wlanRfIndex,* int32_t ∗ *value*) Get wlanRxSampleGainHigh value (units: 0.1 dB).

High-range input power gain correction factor.

Parameters

`in`	*service*	Instance of MIB service
`in`	*wlanRfIndex*	wlanRfIndex value
`out`	*value*	wlanRxSampleGainHigh value (units: 0.1 dB)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRxSampleGainLow (mib_service_t ∗ *service,* int32_t *wlanRfIndex,* int32_t ∗ *value*) Get wlanRxSampleGainLow value (units: 0.1 dB).

Low-range input power gain correction factor.

Parameters

`in`	*service*	Instance of MIB service
`in`	*wlanRfIndex*	wlanRfIndex value
`out`	*value*	wlanRxSampleGainLow value (units: 0.1 dB)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRxSampleGainMid (mib_service_t ∗ *service,* int32_t *wlanRfIndex,* int32_t ∗ *value*) Get wlanRxSampleGainMid value (units: 0.1 dB).

Mid-range input power gain correction factor.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanRxSampleGainMid value (units: 0.1 dB)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanRxUcastDaFilterEnabled (mib_service_t ∗ *service*, int32_t *ifIndex*, int ∗ *value*) Get wlanRxUcastDaFilterEnabled value.

Whether unicast destination address filter is enabled.

If enabled, MAC will drop unicast frames which have destination MAC address different from the receiving station MAC address.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanRxUcastDaFilterEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanShortRetryLimit (mib_service_t ∗ *service*, int32_t *ifIndex*, int32_t ∗ *value*) Get wlanShortRetryLimit value.

This attribute indicates the maximum number of transmission attempts of a frame, the length of which is less than or equal to RTSThreshold, that is made before a failure condition is indicated.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanShortRetryLimit value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanTssiDetectorReading (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int32_t ∗ *value*) Get wlanTssiDetectorReading value.

ADC TSSI feedback detector reading.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanTssiDetectorReading value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanTssiInterval (mib_service_t * *service*, int32_t *wlanRfIndex*, int32_t * *value*) Get wlanTssiInterval value (units: sec).

TSSI sampling interval.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanTssiInterval value (units: sec)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanTxAllocFailCnt (mib_service_t * *service*, int32_t *ifIndex*, uint32_t * *value*) Get wlanTxAllocFailCnt value.

This counter shall be incremented for each memory allocation failure during frame transmission.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanTxAllocFailCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanTxCsd (mib_service_t * *service*, int32_t * *value*) Get wlanTxCsd value (units: samples).

Cyclic shift delay to the transmitted OFDM symbol.

Parameters

in	*service*	Instance of MIB service
out	*value*	wlanTxCsd value (units: samples)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanTxDiversityEnabled (mib_service_t * *service*, int * *value*) Get wlanTxDiversityEnabled value.

Whether TX diversity is enabled.

Parameters

in	*service*	Instance of MIB service
out	*value*	wlanTxDiversityEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanTxFailCnt (mib_service_t ∗ *service*, int32_t *ifIndex*, uint32_t ∗ *value*) Get wlanTxFailCnt value.

This counter shall be incremented for each failure during frame transmission (including allocation, queuing and other failures).

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanTxFailCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanTxIqImbalanceAmplitude (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int32_t ∗ *value*) Get wlanTxIqImbalanceAmplitude value (units: 0.1 dB).

Transmitted signal I/Q imbalance amplitude correction factor.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanTxIqImbalanceAmplitude value (units: 0.1 dB)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanTxIqImbalancePhase (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int32_t ∗ *value*) Get wlanTxIqImbalancePhase value (units: 0.1 degree).

Transmitted signal I/Q imbalance phase correction factor.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
out	*value*	wlanTxIqImbalancePhase value (units: 0.1 degree)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanTxQueueFailCnt (mib_service_t ∗ *service*, int32_t *ifIndex*, uint32_t ∗ *value*) Get wlanTxQueueFailCnt value.

This counter shall be incremented for each queuing failure during frame transmission.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanTxQueueFailCnt value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_get_wlanTxSaOverrideEnabled (mib_service_t ∗ *service*, int32_t *ifIndex*, int ∗ *value*) Get wlanTxSaOverride-Enabled value.

Whether source address override is enabled.

If enabled, source MAC address (SA) can be set arbitrarily per frame transmission by upper layer, without changing wlanMacAddress.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
out	*value*	wlanTxSaOverrideEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanBssid (mib_service_t ∗ *service*, int32_t *ifIndex*, eui48_t *value*) Set wlanBssid value.

802.11 BSSID address.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
in	*value*	wlanBssid value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanChannelLoadThreshold (mib_service_t ∗ *service*, int32_t *ifIndex*, int32_t *value*) Set wlanChannelLoad-Threshold value (units: dBm).

Threshold of received signal strength above which the channel will be considered busy.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
in	*value*	wlanChannelLoadThreshold value (units: dBm)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanChannelProbingInterval (mib_service_t * *service*, int32_t *ifIndex*, int32_t *value*) Set wlanChannelProbingInterval value (units: milliseconds).

Channel load probing interval.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
in	*value*	wlanChannelProbingInterval value (units: milliseconds)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanConfigSaveStatus (mib_service_t * *service*, mib_ConfigSaveStatus_t *value*) Set wlanConfigSaveStatus value.

WLAN configuration save status.

Parameters

in	*service*	Instance of MIB service
in	*value*	wlanConfigSaveStatus value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanCsIntervalA (mib_service_t * *service*, int32_t *ifIndex*, int32_t *value*) Set wlanCsIntervalA value (units: millisecond).

Channel A interval.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
in	*value*	wlanCsIntervalA value (units: millisecond)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanCsIntervalB (mib_service_t ∗ *service*, int32_t *ifIndex*, int32_t *value*) Set wlanCsIntervalB value (units: millisecond).

Channel B interval.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
in	*value*	wlanCsIntervalB value (units: millisecond)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanCsSyncTolerance (mib_service_t ∗ *service*, int32_t *ifIndex*, int32_t *value*) Set wlanCsSyncTolerance value (units: milliseconds).

This attribute is equivalent to SyncTolerance as defined in IEEE 1609.4-2010 clause 6.2.5.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
in	*value*	wlanCsSyncTolerance value (units: milliseconds)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanDcocEnabled (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int *value*) Set wlanDcocEnabled value.

Whether periodic DCOC is enabled.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
in	*value*	wlanDcocEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanDefaultTxDataRate (mib_service_t ∗ *service*, int32_t *ifIndex*, int32_t *value*) Set wlanDefaultTxDataRate value (units: 500 kbit/s).

Default transmission data rate.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
in	*value*	wlanDefaultTxDataRate value (units: 500 kbit/s)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanDefaultTxPower (mib_service_t ∗ *service*, int32_t *ifIndex*, int32_t *value*) Set wlanDefaultTxPower value (units: dBm).

Default transmission power.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
in	*value*	wlanDefaultTxPower value (units: dBm)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanDefaultTxPowerDbm8 (mib_service_t ∗ *service*, int32_t *ifIndex*, int32_t *value*) Set wlanDefaultTxPowerDbm8 value (units: 1/8 dBm).

Default transmission power in 1/8 dBm.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
in	*value*	wlanDefaultTxPowerDbm8 value (units: 1/8 dBm)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanEdcaCWmax (mib_service_t ∗ *service*, int32_t *wlanEdcaIndex*, uint32_t *value*) Set wlanEdcaCWmax value.

The maximum size of the window that is used for generating a random number for the backoff.

The value of this attribute is such that it could always be expressed in the form of 2∗∗X - 1, where X is an integer.

Parameters

in	*service*	Instance of MIB service
in	*wlanEdcaIndex*	wlanEdcaIndex value

in	*value*	wlanEdcaCWmax value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanEdcaCWmin (mib_service_t ∗ *service*, int32_t *wlanEdcaIndex*, uint32_t *value*) Set wlanEdcaCWmin value.

The minimum size of the window that is used for generating a random number for the backoff.

The value of this attribute is such that it could always be expressed in the form of 2∗∗X - 1, where X is an integer.

Parameters

in	*service*	Instance of MIB service
in	*wlanEdcaIndex*	wlanEdcaIndex value
in	*value*	wlanEdcaCWmin value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/mibs/mibs-edca-example.c.

atlk_rc_t mib_set_wlanEpdEnabled (mib_service_t ∗ *service*, int *value*) Set wlanEpdEnabled value.

Whether IEEE Std 802-2014 EtherType Protocol Discrimination (EPD) is used in the LLC sublayer.

Parameters

in	*service*	Instance of MIB service
in	*value*	wlanEpdEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanFrequency (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int32_t *value*) Set wlanFrequency value (units: MHz).

Current frequency.

Should not be used when channel switching is enabled.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
in	*value*	wlanFrequency value (units: MHz)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/mibs/mibs-example.c, and remote-posix/mibs/mibs-example.c.

atlk_rc_t mib_set_wlanGrfiSignalDelayResolution (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int32_t *value*) Set wlanGrfiSignalDelayResolution value (units: 0.1 usec).

Timing resolution of the delay between transmitted packets to PA and T/R RF switch.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
in	*value*	wlanGrfiSignalDelayResolution value (units: 0.1 usec)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanLoLeakage (mib_service_t ∗ *service*, int32_t *wlanLoLeakageIndex*, int32_t *value*) Set wlanLoLeakage value.

LO leakage cancellation per gain.

Parameters

in	*service*	Instance of MIB service
in	*wlanLoLeakage-Index*	wlanLoLeakageIndex value
in	*value*	wlanLoLeakage value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanMacAddress (mib_service_t ∗ *service*, int32_t *ifIndex*, eui48_t *value*) Set wlanMacAddress value.

802.11 MAC Address.

Please note that MAC frame queues are not flushed when the address is changed, meaning that the previous address value may appear as outgoing frame source address or incoming destination address (in the case of unicast frames) some time after the change.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value

in	*value*	wlanMacAddress value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanPantLut (mib_service_t ∗ *service,* int32_t *wlanRfIndex,* const char ∗ *value,* size_t *size*) Lookup table (LUT) for conversion of power detector output into 1 dBm.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
in	*value*	wlanPantLut array
in	*size*	Size of wlanPantLut array

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanPantLutDbm8 (mib_service_t ∗ *service,* int32_t *wlanPantLutDbm8Index,* int32_t *value*) Set wlanPantLutDbm8 value.

Conversion factor of power detector output into 1/8 dBm LUT.

Parameters

in	*service*	Instance of MIB service
in	*wlanPantLut-Dbm8Index*	wlanPantLutDbm8Index value
in	*value*	wlanPantLutDbm8 value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanPantLutIndex (mib_service_t ∗ *service,* int32_t *wlanRfIndex,* int32_t *value*) Set wlanPantLutIndex value.

Which power antenna LUT entry is used.

Index 0 denotes the defaut power antenna LUT. Index 1-4 denotes one of four power antenna LUTs defined at wlanPantLutDbm8Table.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
in	*value*	wlanPantLutIndex value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanPhyOFDMChannelWidth (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, mib_wlanPhyOFDMChannelWidth_t *value*) Set wlanPhyOFDMChannelWidth value.
Current PHY OFDM channel width.
Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
in	*value*	wlanPhyOFDMChannelWidth value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanPresetFrequency0 (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int32_t *value*) Preset frequency 0.
Should not be used when channel switching is enabled.
Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
in	*value*	wlanPresetFrequency0 value (units: MHz)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanPresetFrequency1 (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int32_t *value*) Preset frequency 1.
Should not be used when channel switching is enabled.
Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
in	*value*	wlanPresetFrequency1 value (units: MHz)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanQosDataEnabled (mib_service_t ∗ *service*, int32_t *ifIndex*, int *value*) Set wlanQosDataEnabled value.
Whether 802.11 QoS data is enabled.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
in	*value*	wlanQosDataEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanRandomBackoffEnabled (mib_service_t ∗ *service,* int32_t *ifIndex,* int *value*) Set wlanRandomBackoff-Enabled value.

Whether MAC transmission random backoff is enabled.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
in	*value*	wlanRandomBackoffEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanRfCalibrationRequired (mib_service_t ∗ *service,* int32_t *wlanRfIndex,* int *value*) Set wlanRfCalibration-Required value.

Whether calibration is required on next system boot.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
in	*value*	wlanRfCalibrationRequired value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanRfEnabled (mib_service_t ∗ *service,* int32_t *wlanRfIndex,* int *value*) Set wlanRfEnabled value.

Whether RF interface is enabled. Once an interface has been disabled, it can only be re-enabled by rebooting the unit.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
in	*value*	wlanRfEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanRfFrontEndConnected (mib_service_t * *service*, int32_t *wlanRfIndex*, int *value*) Set wlanRfFrontEnd-Connected value.

Whether an external RF front-end is connected.

Parameters

`in`	*service*	Instance of MIB service
`in`	*wlanRfIndex*	wlanRfIndex value
`in`	*value*	wlanRfFrontEndConnected value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanRfFrontEndOffset (mib_service_t * *service*, int32_t *wlanRfIndex*, int32_t *value*) Set wlanRfFrontEnd-Offset value (units: dBm).

Attenuation of the first transmitted frame's output power. Relevant only when a RF front-end is used.

Parameters

`in`	*service*	Instance of MIB service
`in`	*wlanRfIndex*	wlanRfIndex value
`in`	*value*	wlanRfFrontEndOffset value (units: dBm)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanRfTestMode (mib_service_t * *service*, int32_t *wlanRfIndex*, mib_wlanRfTestMode_t *value*) Set wlanRfTestMode value.

RF interface test mode.

Parameters

`in`	*service*	Instance of MIB service
`in`	*wlanRfIndex*	wlanRfIndex value
`in`	*value*	wlanRfTestMode value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanRxDiversityEnabled (mib_service_t * *service*, int *value*) Set wlanRxDiversityEnabled value.

Whether RX diversity is enabled.

Parameters

in	*service*	Instance of MIB service
in	*value*	wlanRxDiversityEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanRxDuplicateFrameFilteringEnabled (mib_service_t * *service*, int *value*) Set wlanRxDuplicateFrameFilteringEnabled value.

Whether MAC duplicate frame filtering is enabled in RX diversity mode.

Parameters

in	*service*	Instance of MIB service
in	*value*	wlanRxDuplicateFrameFilteringEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanRxIqImbalanceAmplitude (mib_service_t * *service*, int32_t *wlanRfIndex*, int32_t *value*) Set wlanRxIqImbalanceAmplitude value (units: 0.1 dB).

Received signal I/Q imbalance amplitude correction factor.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
in	*value*	wlanRxIqImbalanceAmplitude value (units: 0.1 dB)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanRxIqImbalancePhase (mib_service_t * *service*, int32_t *wlanRfIndex*, int32_t *value*) Set wlanRxIqImbalancePhase value (units: 0.1 degree).

Received signal I/Q imbalance phase correction factor.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
in	*value*	wlanRxIqImbalancePhase value (units: 0.1 degree)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanRxSampleGainHigh (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int32_t *value*) Set wlanRxSampleGainHigh value (units: 0.1 dB).

High-range input power gain correction factor.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
in	*value*	wlanRxSampleGainHigh value (units: 0.1 dB)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanRxSampleGainLow (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int32_t *value*) Set wlanRxSampleGainLow value (units: 0.1 dB).

Low-range input power gain correction factor.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
in	*value*	wlanRxSampleGainLow value (units: 0.1 dB)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanRxSampleGainMid (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int32_t *value*) Set wlanRxSampleGainMid value (units: 0.1 dB).

Mid-range input power gain correction factor.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
in	*value*	wlanRxSampleGainMid value (units: 0.1 dB)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanRxUcastDaFilterEnabled (mib_service_t ∗ *service*, int32_t *ifIndex*, int *value*) Set wlanRxUcastDaFilterEnabled value.

Whether unicast destination address filter is enabled.

If enabled, MAC will drop unicast frames which have destination MAC address different from the receiving station MAC address.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
in	*value*	wlanRxUcastDaFilterEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanShortRetryLimit (mib_service_t ∗ *service*, int32_t *ifIndex*, int32_t *value*) Set wlanShortRetryLimit value.

This attribute indicates the maximum number of transmission attempts of a frame, the length of which is less than or equal to RTSThreshold, that is made before a failure condition is indicated.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
in	*value*	wlanShortRetryLimit value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanTssiInterval (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int32_t *value*) Set wlanTssiInterval value (units: sec).

TSSI sampling interval.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
in	*value*	wlanTssiInterval value (units: sec)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanTxCsd (mib_service_t ∗ *service*, int32_t *value*) Set wlanTxCsd value (units: samples).

Cyclic shift delay to the transmitted OFDM symbol.

Parameters

in	*service*	Instance of MIB service
in	*value*	wlanTxCsd value (units: samples)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanTxDiversityEnabled (mib_service_t ∗ *service*, int *value*) Set wlanTxDiversityEnabled value.

Whether TX diversity is enabled.

Parameters

in	*service*	Instance of MIB service
in	*value*	wlanTxDiversityEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanTxIqImbalanceAmplitude (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int32_t *value*) Set wlanTxIqImbalanceAmplitude value (units: 0.1 dB).

Transmitted signal I/Q imbalance amplitude correction factor.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
in	*value*	wlanTxIqImbalanceAmplitude value (units: 0.1 dB)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanTxIqImbalancePhase (mib_service_t ∗ *service*, int32_t *wlanRfIndex*, int32_t *value*) Set wlanTxIqImbalancePhase value (units: 0.1 degree).

Transmitted signal I/Q imbalance phase correction factor.

Parameters

in	*service*	Instance of MIB service
in	*wlanRfIndex*	wlanRfIndex value
in	*value*	wlanTxIqImbalancePhase value (units: 0.1 degree)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t mib_set_wlanTxSaOverrideEnabled (mib_service_t ∗ *service*, int32_t *ifIndex*, int *value*) Set wlanTxSaOverrideEnabled value.

Whether source address override is enabled.

If enabled, source MAC address (SA) can be set arbitrarily per frame transmission by upper layer, without changing wlanMacAddress.

Parameters

in	*service*	Instance of MIB service
in	*ifIndex*	ifIndex value
in	*value*	wlanTxSaOverrideEnabled value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.37 atlk/nav.h File Reference

Navigation API.

```
#include <stdint.h>
#include <math.h>
#include <atlk/sdk.h>
```

Data Structures

- struct nav_time_t

  Navigation timestamp.
- struct nav_fix_user_data_t

  Navigation fix user data.
- struct nav_fix_t

  Navigation fix data frame.
- struct nav_satellite_info_t

  Satellite information.
- struct nav_satellite_report_t

  Satellite report data frame.
- struct nav_data_t

  Navigation data frame.

Macros

- #define NAV_TIME_INIT

  Navigation timestamp default initializer.
- #define NAV_FIX_USER_DATA_SIZE 100

  Navigation fix user data buffer size.
- #define NAV_FIX_USES_GNSS (1U << 0)

  Navigation fix uses GNSS.
- #define NAV_FIX_USES_DGNSS (1U << 1)

  Navigation fix uses DGNSS.
- #define NAV_FIX_USES_DR (1U << 2)

  Navigation fix uses DR.
- #define NAV_FIX_INIT

  Navigation fix data frame default initializer.
- #define NAV_SATELLITE_INFO_AZIMUTH_DEG_NA UINT16_MAX

  Value indicating that satellite azimuth is N/A.
- #define NAV_SATELLITE_INFO_ELEVATION_DEG_NA UINT8_MAX

  Value indicating that satellite elevation is N/A.
- #define NAV_SATELLITE_INFO_CNR_DB_NA UINT8_MAX

Value indicating that satellite is not tracked.
- #define NAV_SATELLITE_INFO_ARRAY_SIZE_MAX 24

  Maximum number of satellites information.
- #define NAV_SATELLITE_REPORT_INIT

  Satellite report data frame default initializer.
- #define NAV_DATA_TYPE_FIX (1U << 0)

  Navigation data frame contains a navigation fix.
- #define NAV_DATA_TYPE_SATELLITE_REPORT (1U << 1)

  Navigation data frame contains satellite report.
- #define NAV_DATA_INIT

  Navigation data frame default initializer.

Typedefs

- typedef struct nav_service nav_service_t

  Navigation service instance.
- typedef atlk_rc_t(* nav_data_handler_t)(nav_service_t *service, nav_data_t *data)

  Navigation data handler.

Enumerations

- enum nav_fix_mode_t {
  NAV_FIX_MODE_NA = 0, NAV_FIX_MODE_NO_FIX = 1, NAV_FIX_MODE_TIME_ONLY = NAV_FIX_MODE_NO-_FIX, NAV_FIX_MODE_2D = 2,
  NAV_FIX_MODE_3D = 3 }

  Navigation fix mode.
- enum nav_satellites_t { NAV_SATELLITES_GPS = 0, NAV_SATELLITES_GLONASS = 1, NAV_SATELLITES_MAX = NAV_SATELLITES_GLONASS }

  GNSS satellite systems.

Functions

- double nav_time_to_posix_time (const nav_time_t *time)

  Convert navigation timestamp to POSIX time.

7.37.1 Detailed Description

Navigation API.

7.37.2 Typedef Documentation

typedef atlk_rc_t(* nav_data_handler_t)(nav_service_t *service, nav_data_t *data) Navigation data handler.
Callback is called each time a new navigation data frame is available.
Parameters

in	*service*	Navigation service instance
in	*data*	Navigation data frame to handle

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.37.3 Enumeration Type Documentation

enum nav_fix_mode_t Navigation fix mode.

Enumerator

NAV_FIX_MODE_NA Navigation fix mode is N/A.

NAV_FIX_MODE_NO_FIX No navigation fix.

NAV_FIX_MODE_TIME_ONLY No navigation fix alias.

Note

Legacy enumeration name is inaccurate; this fix mode does not necessarily mean that time is available.

NAV_FIX_MODE_2D A 2D (two dimensional) position fix is available.

NAV_FIX_MODE_3D A 3D (three dimensional) position fix is available.

enum nav_satellites_t GNSS satellite systems.

Enumerator

NAV_SATELLITES_GPS Satellites belonging to GPS.

NAV_SATELLITES_GLONASS Satellites belonging to GLONASS.

NAV_SATELLITES_MAX The last value in nav_satellites_t.

7.37.4 Function Documentation

double nav_time_to_posix_time (const nav_time_t ∗ *time*) `[inline]` Convert navigation timestamp to POSIX time.
POSIX time is defined as the number of seconds that have elapsed since 1970-01-01T00:00:00Z, not counting leap seconds.

7.38 atlk/nav_service.h File Reference

Navigation service API.

```
   #include <atlk/sdk.h>
#include <atlk/nav.h>
```

Typedefs

- typedef struct nav_subscriber nav_fix_subscriber_t

  Navigation fix subscriber.
- typedef void(∗ nav_fix_processor_t)(nav_fix_t ∗fix, void ∗context)

  Navigation fix processing callback function.
- typedef struct nav_subscriber nav_data_subscriber_t

  Navigation data subscriber.

Functions

- atlk_rc_t nav_default_service_get (nav_service_t ∗∗service_ptr)

  Get pointer to default navigation service.
- atlk_rc_t nav_service_delete (nav_service_t ∗service)

  Delete navigation service.
- atlk_rc_t nav_fix_subscriber_create (nav_service_t ∗service, nav_fix_subscriber_t ∗∗subscriber_ptr)

  Create navigation fix subscriber.
- atlk_rc_t nav_fix_subscriber_delete (nav_fix_subscriber_t ∗subscriber)

  Delete navigation fix subscriber.
- atlk_rc_t nav_fix_receive (nav_fix_subscriber_t ∗subscriber, nav_fix_t ∗fix, const atlk_wait_t ∗wait)

Receive new navigation fix via `subscriber`.

- atlk_rc_t nav_fix_publish (nav_service_t *service, nav_fix_t *fix)

  Publish navigation fix for subscribers of `service`.
- atlk_rc_t nav_fix_process_set (nav_service_t *service, nav_fix_processor_t callback, void *context)

  Register navigation fix processing callback.
- atlk_rc_t nav_data_subscriber_create (nav_service_t *service, uint32_t data_mask, nav_data_subscriber_t **subscriber_ptr)

  Create navigation data subscriber.
- atlk_rc_t nav_data_subscriber_delete (nav_data_subscriber_t *subscriber)

  Delete navigation data subscriber.
- atlk_rc_t nav_data_receive (nav_data_subscriber_t *subscriber, nav_data_t *data, const atlk_wait_t *wait)

  Receive new navigation data via `subscriber`.
- atlk_rc_t nav_data_publish (nav_service_t *service, nav_data_t *data)

  Publish navigation data for subscribers of `service`.

7.38.1 Detailed Description

Navigation service API.

Note

Service is (optionally) implemented by user on host.

7.38.2 Typedef Documentation

typedef void(* nav_fix_processor_t)(nav_fix_t *fix, void *context) Navigation fix processing callback function.

Parameters

`in,out`	*fix*	Navigation data fix to be published
`in,out`	*context*	Callback context

7.38.3 Function Documentation

atlk_rc_t nav_data_publish (nav_service_t * *service,* nav_data_t * *data*) Publish navigation data for subscribers of `service`.

Parameters

`in`	*service*	Navigation service instance
`in`	*data*	Navigation data frame to publish

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

remote-posix/gnss/gnss-example.c.

atlk_rc_t nav_data_receive (nav_data_subscriber_t * *subscriber,* nav_data_t * *data,* const atlk_wait_t * *wait*) Receive new navigation data via `subscriber`.

See Also

Using wait option.

Parameters

in	*subscriber*	Navigation data subscriber
out	*data*	Navigation data frame
in	*wait*	Wait specification (optional)

Return values

ATLK_OK	if succeeded
ATLK_E_NOT_READY	if new navigation data is not available and `wait` is NULL
ATLK_E_TIMEOUT	if new navigation data is not available and `wait` is of type ATLK_WAIT_INTERVAL.

Returns

Error code if failed

Examples:

craton-threadx/nav/nav-data-example.c.

atlk_rc_t nav_data_subscriber_create (nav_service_t * *service*, uint32_t *data_mask*, nav_data_subscriber_t ** *subscriber_ptr*) Create navigation data subscriber.

Note

`data_mask` is a bitmask of NAV_DATA_TYPE_FIX and NAV_DATA_TYPE_SATELLITE_REPORT.

Parameters

in	*service*	Navigation service
in	*data_mask*	Navigation data subscription mask
out	*subscriber_ptr*	Navigation data subscriber pointer

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/nav/nav-data-example.c.

atlk_rc_t nav_data_subscriber_delete (nav_data_subscriber_t * *subscriber*) Delete navigation data subscriber.

Parameters

in	*subscriber*	Navigation data subscriber

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/nav/nav-data-example.c.

atlk_rc_t nav_default_service_get (nav_service_t ** *service_ptr*) Get pointer to default navigation service.

Parameters

out	*service_ptr*	Pointer to navigation service

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/gnss/gnss-integration-example.c, craton-threadx/nav/nav-data-example.c, and craton-threadx/nav/nav-example.c.

atlk_rc_t nav_fix_process_set (nav_service_t ∗ *service,* nav_fix_processor_t *callback,* void ∗ *context*) Register navigation fix processing callback.

Registering the processing callback is optional. When registered, callback is called on every fix before it is published. Publishing of fix is delayed by the callbacks running time.

Parameters

in	*service*	Navigation service instance
in	*callback*	Navigation fix processing callback function
in,out	*context*	Callback context (optional)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t nav_fix_publish (nav_service_t ∗ *service,* nav_fix_t ∗ *fix*) Publish navigation fix for subscribers of `service`.

Parameters

in	*service*	Navigation service instance
in	*fix*	Navigation fix to publish

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/gnss/gnss-integration-example.c.

atlk_rc_t nav_fix_receive (nav_fix_subscriber_t ∗ *subscriber,* nav_fix_t ∗ *fix,* const atlk_wait_t ∗ *wait*) Receive new navigation fix via `subscriber`.

See Also

Using wait option.

Parameters

in	*subscriber*	Navigation fix subscriber
out	*fix*	Navigation fix
in	*wait*	Wait specification (optional)

Return values

ATLK_OK	if succeeded
ATLK_E_NOT_READY	if new navigation fix is not available and `wait` is NULL
ATLK_E_TIMEOUT	if new navigation fix is not available and `wait` is of type ATLK_WAIT_INTERVAL.

Returns

Error code if failed

Examples:

craton-threadx/nav/nav-example.c.

atlk_rc_t nav_fix_subscriber_create (nav_service_t * *service*, nav_fix_subscriber_t ** *subscriber_ptr*) Create navigation fix subscriber.
Parameters

in	*service*	Navigation service
out	*subscriber_ptr*	Navigation fix subscriber pointer

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/nav/nav-example.c.

atlk_rc_t nav_fix_subscriber_delete (nav_fix_subscriber_t * *subscriber*) Delete navigation fix subscriber.
Parameters

in	*subscriber*	Navigation fix subscriber

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/nav/nav-example.c.

atlk_rc_t nav_service_delete (nav_service_t * *service*) Delete navigation service.
Parameters

in	*service*	Navigation service

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/nav/nav-data-example.c, and craton-threadx/nav/nav-example.c.

7.39 atlk/os.h File Reference

Autotalks OS abstraction definitions.

```
#include <atlk/sdk.h>
```

Data Structures

- struct atlk_thread_sched_t

  Thread scheduling parameters.

7.39.1 Detailed Description

Autotalks OS abstraction definitions.

7.40 atlk/remote.h File Reference

Transport for remote service access.

```
#include <atlk/sdk.h>
#include <atlk/eui48.h>
```

Data Structures

- struct remote_ip_transport_config_t

  IP remote transport configuration.
- struct remote_ll_device_ops_t

  Link layer driver operations.
- struct remote_ll_transport_config_t

  Link layer remote transport configuration.

Macros

- #define REMOTE_IP_TRANSPORT_CONFIG_INIT

  IP remote transport configuration default initializer.
- #define REMOTE_LL_TRANSPORT_CONFIG_INIT

  IP remote transport configuration default initializer.

Typedefs

- typedef struct remote_transport remote_transport_t

  Remote transport instance.

Functions

- atlk_rc_t remote_ip_transport_create (const remote_ip_transport_config_t ∗config, remote_transport_t ∗∗transport_ptr)

  Create IP-based remote transport instance.
- atlk_rc_t remote_transport_delete (remote_transport_t ∗transport_ptr)

  Delete a remote transport instance.
- atlk_rc_t remote_util_local_ipv4_address_get (const char ∗interface_name, uint32_t ∗local_ipv4_address)

  Get local IPv4 address of a Network interface.
- atlk_rc_t remote_ll_transport_create (const remote_ll_transport_config_t ∗config, remote_transport_t ∗∗transport_ptr)

  Create link layer based remote transport instance.

7.40.1 Detailed Description

Transport for remote service access.

7.40.2 Function Documentation

atlk_rc_t remote_ip_transport_create (const remote_ip_transport_config_t ∗ *config*, remote_transport_t ∗∗ *transport_ptr*)
Create IP-based remote transport instance.
Parameters

in	*config*	IP remote transport configuration
out	*transport_ptr*	Remote transport pointer

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

remote-posix/crypto/ecdsa-benchmark.c, remote-posix/crypto/ecdsa-example.c, remote-posix/crypto/ecies-example.c, remote-posix/mibs/mibs-example.c, and remote-posix/v2x/v2x-example.c.

atlk_rc_t remote_ll_transport_create (const remote_ll_transport_config_t ∗ *config*, remote_transport_t ∗∗ *transport_ptr*)
Create link layer based remote transport instance.
Parameters

in	*config*	Link layer transport configuration
out	*transport_ptr*	Remote transport pointer

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t remote_transport_delete (remote_transport_t ∗ *transport_ptr*) Delete a remote transport instance.
Parameters

in	*transport_ptr*	Remote transport pointer

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

remote-posix/crypto/ecdsa-benchmark.c, remote-posix/crypto/ecdsa-example.c, remote-posix/mibs/mibs-example.c, and remote-posix/v2x/v2x-example.c.

atlk_rc_t remote_util_local_ipv4_address_get (const char ∗ *interface_name,* uint32_t ∗ *local_ipv4_address*) Get local IPv4 address of a Network interface.

Parameters

`in`	*interface_name*	Interface name (e.g. "eth0")
`out`	*local_ipv4_address*	Local IPv4 address

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

remote-posix/crypto/ecdsa-benchmark.c, remote-posix/crypto/ecdsa-example.c, remote-posix/crypto/ecies-example.c, remote-posix/mibs/mibs-example.c, and remote-posix/v2x/v2x-example.c.

7.41 atlk/rng.h File Reference

RNG API.

```
#include <atlk/sdk.h>
```

Functions

- atlk_rc_t rng_data_get (void ∗ptr, size_t size)
  Get random bytes.

7.41.1 Detailed Description

RNG API.

7.41.2 Function Documentation

atlk_rc_t rng_data_get (void ∗ *ptr,* size_t *size*) Get random bytes.

Parameters

`in`	*ptr*	Buffer to store random bytes
`in`	*size*	Buffer length in octets

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecies-example.c, and remote-posix/crypto/ecies-example.c.

7.42 atlk/sdk.h File Reference

Autotalks SDK common declarations and macros.

```
#include <stdint.h>
#include <stddef.h>
#include <atlk/verinfo.h>
#include <atlk/compiler.h>
```

Data Structures

- struct atlk_wait_t

    Wait option.
- struct atlk_fragment_t

    Data fragment.
- struct atlk_const_fragment_t

    Read-only data fragment.

Macros

- #define ATLK_OK ATLK_RC(0)

    Operation successful.
- #define ATLK_E_UNSPECIFIED ATLK_RC(1)

    Unspecified error.
- #define ATLK_E_INVALID_ARG ATLK_RC(2)

    Invalid argument.
- #define ATLK_E_UNSUPPORTED ATLK_RC(3)

    Operation not supported.
- #define ATLK_E_INVALID_STATE ATLK_RC(4)

    Object in invalid state.
- #define ATLK_E_NOT_FOUND ATLK_RC(5)

    Object not found.
- #define ATLK_E_EXISTS ATLK_RC(6)

    Object already exists.
- #define ATLK_E_NOT_READY ATLK_RC(7)

    Not ready to perform operation.
- #define ATLK_E_TIMEOUT ATLK_RC(8)

    Operation timed out.
- #define ATLK_E_OUT_OF_DOMAIN ATLK_RC(9)

    Numerical argument out of domain.
- #define ATLK_E_OUT_OF_RANGE ATLK_RC(10)

    Numerical result out of range.
- #define ATLK_E_OUT_OF_MEMORY ATLK_RC(11)

    Failed to allocate memory.
- #define ATLK_E_ADDRESS_IN_USE ATLK_RC(12)

    Address already in use.
- #define ATLK_E_CONNECTION_REFUSED ATLK_RC(13)

    Connection refused.
- #define ATLK_E_CONNECTION_LOST ATLK_RC(14)

    Connection lost.
- #define ATLK_E_PROTOCOL_ERROR ATLK_RC(15)

    Protocol error.
- #define ATLK_E_PROTOCOL_MISMATCH ATLK_RC(16)

Protocol version mismatch.

- #define ATLK_E_OUT_OF_BOUNDS ATLK_RC(17)

  Array access out of bounds.
- #define ATLK_E_BAD_ALIGNMENT ATLK_RC(18)

  Address not aligned as required.
- #define ATLK_E_BUFFER_TOO_SMALL ATLK_RC(19)

  Buffer is too small.
- #define ATLK_E_IO_ERROR ATLK_RC(20)

  Input/output error.
- #define ATLK_WAIT_INIT

  Wait option default initializer.
- #define ATLK_FRAGMENT_INIT

  Default initializer for atlk_fragment_t and atlk_const_fragment_t.

Typedefs

- typedef unsigned int atlk_rc_t

  Return code type.

Enumerations

- enum atlk_wait_type_t { ATLK_WAIT_INTERVAL = 0, ATLK_WAIT_FOREVER = 1 }

  Wait option type.

Functions

- const char ∗ atlk_rc_to_str (atlk_rc_t rc)

  Convert atlk_rc_t to human-readable error message.
- int atlk_error (atlk_rc_t rc)

  Tell whether a return code indicates an error.

Variables

- const atlk_wait_t atlk_wait_forever

  Predefined "wait forever" wait option.

7.42.1 Detailed Description

Autotalks SDK common declarations and macros.

7.42.2 Enumeration Type Documentation

enum atlk_wait_type_t Wait option type.

Enumerator

ATLK_WAIT_INTERVAL Wait a time interval.

ATLK_WAIT_FOREVER Wait indefinitely.

7.42.3 Function Documentation

int atlk_error (atlk_rc_t *rc*) `[inline]` Tell whether a return code indicates an error.

Return values

0	`rc` indicates success
1	`rc` indicates error

Examples:

craton-threadx/bridge/v2x-udp-bridge-example.c, craton-threadx/bt-spi2uart/bt-spi2uart-example.c, craton-threadx/can/can-example.c, craton-threadx/can/can-hw-filter-example.c, craton-threadx/cli/cli-example.c, craton-threadx/crypto/aes-example.c, craton-threadx/crypto/ecdsa-benchmark.c, craton-threadx/crypto/ecdsa-example.c, craton-threadx/crypto/ecies-example.c, craton-threadx/crypto/secure-storage-example.c, craton-threadx/dot4/dot4-channel-switching-example.-c, craton-threadx/firmware/fw-update-example.c, craton-threadx/gnss-teseo/gnss-teseo-fw-update-example.c, craton-threadx/gnss-teseo/gnss-teseo-sou-example.c, craton-threadx/gnss/gnss-integration-example.c, craton-threadx/i2s/i2s-example.c, craton-threadx/imq/imq-client.c, craton-threadx/imq/imq-echo-server.c, craton-threadx/mibs/mibs-edca-example.c, craton-threadx/mibs/mibs-example.c, craton-threadx/nav/nav-data-example.c, craton-threadx/nav/nav-example.c, craton-threadx/nav/system-time-benchmark.c, craton-threadx/net/http-example.c, craton-threadx/net/nx-raw-packet-receive-example.c, craton-threadx/net/udp-receive-example.c, craton-threadx/otp/otp-example.c, craton-threadx/sntp/sntp-example.c, craton-threadx/spi/spi-master-example.c, craton-threadx/spi/spi-slave-example.-c, craton-threadx/sys-alarm/sys-alarm-example.c, craton-threadx/v2x-emulator/v2x-emulator-over-udp-example.c, craton-threadx/v2x-emulator/v2x-service-user.c, craton-threadx/v2x/v2x-example.c, craton-threadx/wave-ipv6/wave-ipv6-client-example.c, craton-threadx/wave-ipv6/wave-ipv6-example.c, craton-threadx/wave-ipv6/wave-ipv6-server-example.c, craton-threadx/wlan-driver/traffic-monitor-example.c, remote-posix/crypto/aes-example.c, remote-posix/crypto/ecdsa-benchmark.c, remote-posix/crypto/ecdsa-example.c, remote-posix/crypto/ecies-example.c, remote-posix/gnss/gnss-example.c, remote-posix/mibs/mibs-example.c, and remote-posix/v2x/v2x-example.c.

const char∗ atlk_rc_to_str (atlk_rc_t *rc*) Convert atlk_rc_t to human-readable error message.

Error string must **not** be freed by the caller.

Parameters

`in`	*rc*	Return code

Returns

Error message string

Examples:

craton-threadx/bridge/v2x-udp-bridge-example.c, craton-threadx/bt-spi2uart/bt-spi2uart-example.c, craton-threadx/can/can-example.c, craton-threadx/can/can-hw-filter-example.c, craton-threadx/crypto/aes-example.c, craton-threadx/crypto/ecdsa-example.c, craton-threadx/crypto/ecies-example.c, craton-threadx/crypto/secure-storage-example.c, craton-threadx/dot4/dot4-channel-switching-example.c, craton-threadx/firmware/fw-update-example.c, craton-threadx/gnss-teseo/gnss-teseo-fw-update-example.c, craton-threadx/gnss-teseo/gnss-teseo-sou-example.c, craton-threadx/gnss/gnss-integration-example.c, craton-threadx/i2s/i2s-example.c, craton-threadx/imq/imq-client.c, craton-threadx/imq/imq-echo-server.-c, craton-threadx/mibs/mibs-edca-example.c, craton-threadx/mibs/mibs-example.c, craton-threadx/nav/nav-data-example.c, craton-threadx/nav/nav-example.c, craton-threadx/nav/system-time-benchmark.c, craton-threadx/net/http-example.c, craton-threadx/net/nx-raw-packet-receive-example.c, craton-threadx/sntp/sntp-example.c, craton-threadx/spi/spi-master-example.c, craton-threadx/spi/spi-slave-example.c, craton-threadx/sys-alarm/sys-alarm-example.c, craton-threadx/v2x-emulator/v2x-emulator-over-udp-example.c, craton-threadx/v2x-emulator/v2x-service-user.c, craton-threadx/v2x/v2x-example.c, craton-threadx/wave-ipv6/wave-ipv6-client-example.c, craton-threadx/wave-ipv6/wave-ipv6-example.c, craton-threadx/wave-ipv6/wave-ipv6-server-example.c, craton-threadx/wlan-driver/traffic-monitor-example.c, remote-posix/crypto/aes-example.c, remote-posix/crypto/ecdsa-benchmark.c, remote-posix/crypto/ecdsa-example.c, remote-posix/crypto/ecies-example.c, remote-posix/gnss/gnss-example.c, remote-posix/mibs/mibs-example.c, and remote-posix/v2x/v2x-example.c.

7.43 atlk/sensor.h File Reference

Vehicle Sensors API.

```
#include <atlk/sdk.h>
```

Data Structures

- struct sensor_value_params_t

  Sensor value parameters.
- struct sensor_wheels_speed_t

  Vehicle wheels speed.

Macros

- #define SENSOR_VALUE_NA INT32_MIN

  Value indicating that a sensor value is N/A.
- #define SENSOR_VALUE_PARAMS_INIT

  Sensor value parameters default initializer.
- #define SENSOR_WHEELS_SPEED_INIT

  Vehicle wheels speed default initializer.

Typedefs

- typedef int32_t sensor_value_t

  Sensor value.

Enumerations

- enum sensor_units_t { SENSOR_UNITS_MPS = 0, SENSOR_UNITS_RADPS = 1, SENSOR_UNITS_NA = 255 }

  Sensor physical units.

7.43.1 Detailed Description

Vehicle Sensors API.

7.43.2 Enumeration Type Documentation

enum sensor_units_t Sensor physical units.

Enumerator

SENSOR_UNITS_MPS Units of meters per second.

SENSOR_UNITS_RADPS Units of radians per second.

SENSOR_UNITS_NA Value indicating that units are N/A.

7.44 atlk/sha.h File Reference

SHA common definitions.

```
#include <atlk/sdk.h>
```

Data Structures

- struct sha_digest_t

  SHA digest.

Macros

- #define SHA_224_DIGEST_SIZE 28
  SHA-224 digest size in octets.
- #define SHA_256_DIGEST_SIZE 32
  SHA-256 digest size in octets.
- #define SHA_384_DIGEST_SIZE 48
  SHA-384 digest size in octets.
- #define SHA_DIGEST_MAX_SIZE SHA_384_DIGEST_SIZE
  Maximum SHA digest size in octets.
- #define SHA_DIGEST_INIT
  SHA digest default initializer.

Enumerations

- enum sha_algorithm_t { SHA_224 = 0, SHA_256 = 1, SHA_384 = 2 }
  SHA algorithm.

Functions

- int sha_algorithm_valid (sha_algorithm_t algorithm)
  Check if a given SHA algorithm is valid.
- size_t sha_algorithm_digest_size (sha_algorithm_t algorithm)
  Get SHA digest size for a given SHA algorithm.

7.44.1 Detailed Description

SHA common definitions.

7.44.2 Enumeration Type Documentation

enum sha_algorithm_t SHA algorithm.

Enumerator

SHA_224	SHA-224.
SHA_256	SHA-256.
SHA_384	SHA-384.

7.44.3 Function Documentation

size_t sha_algorithm_digest_size (sha_algorithm_t *algorithm*) `[inline]` Get SHA digest size for a given SHA algorithm.

Parameters

`in`	*algorithm*	SHA algorithm

Returns

SHA digest size for a valid SHA algorithm, 0 otherwise

int sha_algorithm_valid (sha_algorithm_t *algorithm*) `[inline]` Check if a given SHA algorithm is valid.

Parameters

`in`	*algorithm*	SHA algorithm

Returns

1 if SHA algorithm is valid, 0 otherwise

7.45 atlk/sha_sw.h File Reference

Autotalks SHA SW API.

```
   #include <atlk/sdk.h>
#include <atlk/sha.h>
```

Functions

- atlk_rc_t sha_sw_compute (sha_algorithm_t algorithm, const void ∗data_ptr, size_t data_size, sha_digest_t ∗digest)
  Compute SHA digest using just software.
- atlk_rc_t sha_sw_sha224_compute (const void ∗data_ptr, size_t data_size, sha_digest_t ∗digest)
  Compute SHA-224 using just software.
- atlk_rc_t sha_sw_sha256_compute (const void ∗data_ptr, size_t data_size, sha_digest_t ∗digest)
  Compute SHA-256 using just software.
- atlk_rc_t sha_sw_sha384_compute (const void ∗data_ptr, size_t data_size, sha_digest_t ∗digest)
  Compute SHA-384 using just software.

7.45.1 Detailed Description

Autotalks SHA SW API.

7.45.2 Function Documentation

atlk_rc_t sha_sw_compute (sha_algorithm_t *algorithm,* const void ∗ *data_ptr,* size_t *data_size,* sha_digest_t ∗ *digest*) Compute SHA digest using just software.

Only the first sha_digest_t::value_size octets of sha_digest_t::value are the calculated hash value.

Parameters

in	*algorithm*	SHA algorithm
in	*data_ptr*	Data over which the hash will be computed
in	*data_size*	Data length in octets
out	*digest*	Calculated SHA digest

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t sha_sw_sha224_compute (const void ∗ *data_ptr,* size_t *data_size,* sha_digest_t ∗ *digest*) Compute SHA-224 using just software.

Only the first sha_digest_t::value_size octets of sha_digest_t::value are the calculated hash value.

Parameters

in	*data_ptr*	Data over which the hash will be computed
in	*data_size*	Data length in octets
out	*digest*	Calculated SHA-224 digest

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t sha_sw_sha256_compute (const void ∗ *data_ptr,* size_t *data_size,* sha_digest_t ∗ *digest*) Compute SHA-256 using just software.

Only the first sha_digest_t::value_size octets of sha_digest_t::value are the calculated hash value.

Parameters

in	*data_ptr*	Data over which the hash will be computed
in	*data_size*	Data length in octets
out	*digest*	Calculated SHA-256 digest

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

remote-posix/crypto/ecdsa-example.c.

atlk_rc_t sha_sw_sha384_compute (const void ∗ *data_ptr*, size_t *data_size*, sha_digest_t ∗ *digest*) Compute SHA-384 using just software.

Only the first sha_digest_t::value_size octets of sha_digest_t::value are the calculated hash value.

Parameters

in	*data_ptr*	Data over which the hash will be computed
in	*data_size*	Data length in octets
out	*digest*	Calculated SHA-384 digest

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.46 atlk/slx97.h File Reference

SLx97 host API.

```
   #include <atlk/sdk.h>
#include <atlk/hsm.h>
#include <atlk/aes.h>
```

Data Structures

- struct slx97_host_sec_config_t

  SLx97 host communication security parameters.
- struct slx97_host_sec_key_t

  SLx97 host communication security key.
- struct slx97_chip_info_t

  SLx97 chip information.
- struct slx97_dsk_t

  Device specific key used for SLx97 communication security.

Macros

- #define SLX97_HOST_SEC_CONFIG_INIT

  SLx97 host communication security default initializer.
- #define SLX97_HOST_SEC_KEY_INIT

  SLx97 host communication security key default initializer.
- #define SLX97_DSK_SIZE 32

  Device specific key size in bytes used for SLx97 communication security.

Typedefs

- typedef atlk_rc_t(∗ slx97_dsk_callback_t)(slx97_dsk_t ∗dsk)

  Callback function for retrieving the value of a device specific key.

Functions

- atlk_rc_t slx97_host_hsm_service_get (hsm_service_t ∗∗service_ptr)

  Get HSM SLx97 service.
- atlk_rc_t slx97_chip_info_get (slx97_chip_info_t ∗info)

  Get HSM SLx97 chip information.
- atlk_rc_t slx97_firmware_info_str_get (const slx97_chip_info_t ∗info, char ∗firmware_info, size_t firmware_info_size)

  Get Autotalks firmware information string.

7.46.1 Detailed Description

SLx97 host API.

7.46.2 Typedef Documentation

typedef atlk_rc_t(∗ slx97_dsk_callback_t)(slx97_dsk_t ∗dsk) Callback function for retrieving the value of a device specific key.

Parameters

out	*dsk*	Device specific key

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.46.3 Function Documentation

atlk_rc_t slx97_chip_info_get (slx97_chip_info_t ∗ *info*) Get HSM SLx97 chip information.

Parameters

out	*info*	SLx97 chip information

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t slx97_firmware_info_str_get (const slx97_chip_info_t ∗ *info*, char ∗ *firmware_info*, size_t *firmware_info_size*) Get Autotalks firmware information string.

Parameters

in	*info*	SLx97 chip information
out	*firmware_info*	Firmware information string
in,out	*firmware_info_size*	Maximum (in) and actual (out) firmware information string in bytes

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t slx97_host_hsm_service_get (hsm_service_t ** *service_ptr*) Get HSM SLx97 service.

Parameters

out	*service_ptr*	HSM SLx97 service

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.47 atlk/slx97_remote.h File Reference

Remote SLx97 host API.

```
#include <atlk/sdk.h>
#include <atlk/slx97.h>
#include <atlk/remote.h>
```

Functions

- atlk_rc_t slx97_remote_hsm_service_create (remote_transport_t *transport, const slx97_host_sec_config_t *sec_config, slx97_dsk_callback_t dsk_callback)

  Create remote HSM service.

7.47.1 Detailed Description

Remote SLx97 host API.

7.47.2 Function Documentation

atlk_rc_t slx97_remote_hsm_service_create (remote_transport_t * *transport*, const slx97_host_sec_config_t * *sec_config*, slx97_dsk_callback_t *dsk_callback*) Create remote HSM service.

Remarks

Use slx97_host_hsm_service_get to get the HSM SLx97 service after creation.

Parameters

in	*sec_config*	SLx97 host communication security parameters
in	*dsk_callback*	Callback function for retrieving the value of a device specific key

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.48 atlk/snmp_agent.h File Reference

SNMP agent API.

```
#include <atlk/sdk.h>
```

Functions

- atlk_rc_t snmp_agent_start (void)

  Start SNMP agent.
- atlk_rc_t snmp_agent_stop (void)

  Stop SNMP agent.

7.48.1 Detailed Description

SNMP agent API.

7.48.2 Function Documentation

atlk_rc_t snmp_agent_start (void) Start SNMP agent.

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t snmp_agent_stop (void) Stop SNMP agent.

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.49 atlk/sntp_client.h File Reference

SNTP client API.

```
#include <atlk/sdk.h>
#include <atlk/os.h>
```

Data Structures

- struct sntp_info_t

  NTP update info.
- struct sntp_client_config_t

  SNTP client configuration parameters.

Macros

- #define SNTP_SERVER_STRATUM_MIN 1

  Server stratum minimum value.
- #define SNTP_SERVER_STRATUM_MAX 15

  Server stratum maximum value.
- #define SNTP_CLIENT_CONFIG_INIT

  SNTP client configuration parameters default initializer.

Typedefs

- typedef void(∗ sntp_client_update_handler_t)(const sntp_info_t ∗info)

  SNTP client update callback function.

Enumerations

- enum sntp_connection_type_t { SNTP_CONNECTION_TYPE_NA = 0, SNTP_CONNECTION_TYPE_BROADCAST = 1, SNTP_CONNECTION_TYPE_UNICAST = 2 }

  SNTP connection type.

Functions

- double sntp_time_to_posix_time (const sntp_info_t ∗info)

  Convert NTP timestamp to POSIX time.
- atlk_rc_t sntp_client_init (const sntp_client_config_t ∗config)

  Initialize SNTP client.

7.49.1 Detailed Description

SNTP client API.

7.49.2 Enumeration Type Documentation

enum sntp_connection_type_t SNTP connection type.

Enumerator

SNTP_CONNECTION_TYPE_NA	Value indicating that SNTP connection type is N/A.
SNTP_CONNECTION_TYPE_BROADCAST	Listen to broadcast messages.
SNTP_CONNECTION_TYPE_UNICAST	Listen to unicast messages.

7.49.3 Function Documentation

atlk_rc_t sntp_client_init (const sntp_client_config_t ∗ *config*) Initialize SNTP client.

Parameters

in	*config*	SNTP client configuration parameters

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/nav/system-time-benchmark.c, and craton-threadx/sntp/sntp-example.c.

double sntp_time_to_posix_time (const sntp_info_t ∗ *info*) `[inline]` Convert NTP timestamp to POSIX time.

POSIX time is defined as the number of seconds that have elapsed since 1970-01-01T00:00:00Z, not counting leap seconds.

Examples:

craton-threadx/nav/system-time-benchmark.c, and craton-threadx/sntp/sntp-example.c.

7.50 atlk/uart.h File Reference

UART API.

```
#include <atlk/sdk.h>
```

Macros

- #define UART_SPEED_4800_BPS 4800

  Speed of 4800 bits/s.
- #define UART_SPEED_9600_BPS 9600

  Speed of 9600 bits/s.
- #define UART_SPEED_14400_BPS 14400

  Speed of 14400 bits/s.
- #define UART_SPEED_19200_BPS 19200

  Speed of 19200 bits/s.
- #define UART_SPEED_38400_BPS 38400

  Speed of 38400 bits/s.
- #define UART_SPEED_57600_BPS 57600

  Speed of 57600 bits/s.
- #define UART_SPEED_115200_BPS 115200

  Speed of 115200 bits/s.
- #define UART_SPEED_230400_BPS 230400

  Speed of 230400 bits/s.
- #define UART_SPEED_460800_BPS 460800

  Speed of 460800 bits/s.
- #define UART_SPEED_921600_BPS 921600

  Speed of 921600 bits/s.

7.50.1 Detailed Description

UART API.

7.51 atlk/v2x.h File Reference

V2X API declarations.

```
   #include <atlk/sdk.h>
#include <atlk/eui48.h>
```

Data Structures

- struct v2x_channel_id_t

  V2X radio channel identifier.
- struct v2x_dot4_channel_start_request_t

  IEEE Std 1609.4-2016 service primitive MLMEX-CHSTART.request parameters.
- struct v2x_dot4_channel_end_request_t

  IEEE Std 1609.4-2016 service primitive MLMEX-CHEND.request parameters.
- struct v2x_dot4_channel_end_indication_t

  IEEE Std 1609.4-2016 service primitive MLMEX-CHEND.indication parameters.
- struct v2x_netif_profile_t

  Network interface V2X access profile.
- struct v2x_protocol_t

  V2X protocol descriptor.

Macros

- #define V2X_IF_INDEX_NA 0

  Value indicating that V2X physical interface index is N/A.
- #define V2X_IF_INDEX_MIN 1

  Minimum valid V2X physical interface index.
- #define V2X_IF_INDEX_MAX 2

  Maximum valid V2X physical interface index.
- #define V2X_USER_PRIORITY_MIN 0

  V2X MAC User Priority minimum value.
- #define V2X_USER_PRIORITY_MAX 7

  V2X MAC User Priority maximum value.
- #define V2X_USER_PRIORITY_NA INT8_MIN

  Value indicating that V2X MAC User Priority is N/A.
- #define V2X_POWER_DBM_NA INT8_MIN

  Value indicating that power in units of dBm is N/A.
- #define V2X_POWER_DBM8_NA INT16_MIN

  Value indicating that power in units of 1/8 dBm is N/A.
- #define V2X_POWER_DBM8_PER_DBM 8

  1/8 dBm to dBm conversion factor
- #define V2X_INDEFINITE_IMMEDIATE_ACCESS 255

  Indefinite access.
- #define V2X_CHANNEL_NUM_NA 0

  Value indicating that radio channel number is N/A.
- #define V2X_NETIF_PROFILE_INIT

  Network interface V2X access profile default initializer.
- #define V2X_EXPIRY_TIME_MS_MAX 0x7FFF

  Value indicating maximum allowed expiration time in milliseconds.
- #define V2X_EXPIRY_TIME_MS_NA 0

  Value indicating that V2X expiration time is N/A.
- #define V2X_PROTOCOL_INIT

  Default protocol descriptor initializer.
- #define V2X_PROTOCOL_ID_ANY 0ULL

  Value indicating the protocol ID used to create a V2X socket which can send/receive frames with data which includes layer 2 (i.e.

Typedefs

- typedef uint8_t v2x_if_index_t

  V2X physical interface index.
- typedef int8_t v2x_user_priority_t

  V2X MAC User Priority.
- typedef int8_t v2x_power_dbm_t

  Power in units of dBm.
- typedef int16_t v2x_power_dbm8_t

  Power in units of 1/8 dBm.
- typedef uint8_t v2x_channel_num_t

  V2X radio channel number.
- typedef uint16_t v2x_expiry_time_ms_t

  V2X expiration time in milliseconds.
- typedef struct v2x_service v2x_service_t

  V2X service instance.

Enumerations

- enum v2x_datarate_t {
  V2X_DATARATE_NA = 0, V2X_DATARATE_3MBPS = 6, V2X_DATARATE_4_5MBPS = 9, V2X_DATARATE_6-MBPS = 12,
  V2X_DATARATE_9MBPS = 18, V2X_DATARATE_12MBPS = 24, V2X_DATARATE_18MBPS = 36, V2X_DATA-RATE_24MBPS = 48,
  V2X_DATARATE_27MBPS = 54, V2X_DATARATE_36MBPS = 72, V2X_DATARATE_48MBPS = 96, V2X_DAT-ARATE_54MBPS = 108 }

  Data rates in units of 500 kbit/s.
- enum v2x_op_class_t {
  V2X_OP_CLASS_NA = 0, V2X_OP_CLASS_US_ITS_5GHZ_SPACING_10MHZ = 1, V2X_OP_CLASS_US_ITS_5GHZ-_SPACING_20MHZ = 2, V2X_OP_CLASS_EUROPE_ITS_5GHZ_SPACING_10MHZ = 3,
  V2X_OP_CLASS_EUROPE_ITS_5GHZ_SPACING_20MHZ = 4 }

  Operating class.
- enum v2x_time_slot_t { V2X_TIME_SLOT_NA = 0, V2X_TIME_SLOT_0 = 1, V2X_TIME_SLOT_1 = 2, V2X_TIME_-SLOT_EITHER = 3 }

  A set of time slot.
- enum v2x_dot4_channel_end_reason_t { V2X_DOT4_CHANNEL_END_REASON_UNSPECIFIED = 0, V2X_DOT4_C-HANNEL_END_REASON_LOSS_OF_SYNC = 1 }

  Reason for ending access to an IEEE Std 1609.4-2016 channel.
- enum v2x_frame_type_t { V2X_FRAME_TYPE_DATA = 0, V2X_FRAME_TYPE_VSA = 1 }

  V2X MAC frame type.
- enum v2x_sample_type_t { V2X_SAMPLE_TYPE_NA = 0, V2X_SAMPLE_TYPE_CBR = 1 }

  V2X sample type.

7.51.1 Detailed Description

V2X API declarations.

7.51.2 Macro Definition Documentation

#define V2X_NETIF_PROFILE_INIT **Value:**

```
{          \
  .if_index = V2X_IF_INDEX_NA,          \
  .channel_id = V2X_CHANNEL_ID_NA,       \
  .datarate = V2X_DATARATE_NA,           \
  .power_dbm8 = V2X_POWER_DBM_NA         \
}
```

Network interface V2X access profile default initializer.

#define V2X_PROTOCOL_ID_ANY 0ULL Value indicating the protocol ID used to create a V2X socket which can send/receive frames with data which includes layer 2 (i.e.
a "raw" socket above layer 1).

Note

Only one socket of this type can be created per physical interface index and frame type. Standard V2X sockets cannot be opened when this type of socket is open (and vice-versa).

7.51.3 Enumeration Type Documentation

enum v2x_datarate_t Data rates in units of 500 kbit/s.

Enumerator

V2X_DATARATE_NA	Data rate is N/A.
V2X_DATARATE_3MBPS	3 Mbit/s
V2X_DATARATE_4_5MBPS	4.5 Mbit/s

V2X_DATARATE_6MBPS 6 Mbit/s

V2X_DATARATE_9MBPS 9 Mbit/s

V2X_DATARATE_12MBPS 12 Mbit/s

V2X_DATARATE_18MBPS 18 Mbit/s

V2X_DATARATE_24MBPS 24 Mbit/s

V2X_DATARATE_27MBPS 27 Mbit/s

V2X_DATARATE_36MBPS 36 Mbit/s

V2X_DATARATE_48MBPS 48 Mbit/s

V2X_DATARATE_54MBPS 54 Mbit/s

enum v2x_dot4_channel_end_reason_t Reason for ending access to an IEEE Std 1609.4-2016 channel.

Enumerator

V2X_DOT4_CHANNEL_END_REASON_UNSPECIFIED Unspecified reason.

V2X_DOT4_CHANNEL_END_REASON_LOSS_OF_SYNC Loss of time synchronization.

enum v2x_frame_type_t V2X MAC frame type.

Enumerator

V2X_FRAME_TYPE_DATA Data frame.

V2X_FRAME_TYPE_VSA IEEE 802.11 vendor-specific action frame.

enum v2x_op_class_t Operating class.

See Also

IEEE Std 802.11-2012, Annex E.

Remarks

Operating class numbers intentionally don't follow the standard.

Enumerator

V2X_OP_CLASS_NA No operating class selected.

V2X_OP_CLASS_US_ITS_5GHZ_SPACING_10MHZ United States ITS 5 GHz, 10 MHz channel spacing.

V2X_OP_CLASS_US_ITS_5GHZ_SPACING_20MHZ United States ITS 5 GHz, 20 MHz channel spacing.

V2X_OP_CLASS_EUROPE_ITS_5GHZ_SPACING_10MHZ Europe ITS 5 GHz, 10 MHz channel spacing.

V2X_OP_CLASS_EUROPE_ITS_5GHZ_SPACING_20MHZ Europe ITS 5 GHz, 20 MHz channel spacing.

enum v2x_sample_type_t V2X sample type.

Enumerator

V2X_SAMPLE_TYPE_NA Sample type is N/A.

V2X_SAMPLE_TYPE_CBR CBR (channel busy ratio) sample. Receive samples via v2x_sample_int32_receive. Sample range and precision are the same as for mib_get_wlanChannelBusyRatio.

enum v2x_time_slot_t A set of time slot.

Enumerator

V2X_TIME_SLOT_NA No time slot selected.

V2X_TIME_SLOT_0 Time slot #0.

V2X_TIME_SLOT_1 Time slot #1.

V2X_TIME_SLOT_EITHER Either time slot #0 or #1.

7.52 atlk/v2x_emulator.h File Reference

V2X emulator API.

```
   #include <atlk/sdk.h>
#include <atlk/v2x.h>
#include <atlk/v2x_service.h>
```

Typedefs

- typedef struct v2x_emulator v2x_emulator_t

  V2X emulator instance.

Functions

- atlk_rc_t v2x_emulator_service_get (v2x_service_t **service_ptr)

  Get pointer to V2X emulator service.
- atlk_rc_t v2x_emulator_send (v2x_emulator_t *emulator, v2x_if_index_t if_index, const v2x_protocol_t *protocol, const void *data_ptr, size_t data_size, const v2x_receive_params_t *params, const atlk_wait_t *wait)

  Send V2X frame to emulated V2X service.
- atlk_rc_t v2x_emulator_receive (v2x_emulator_t *emulator, v2x_if_index_t *if_index, v2x_protocol_t *protocol, void *data_ptr, size_t *data_size_ptr, v2x_send_params_t *params, const atlk_wait_t *wait)

  Receive V2X frame from emulated V2X service.

7.52.1 Detailed Description

V2X emulator API.

7.52.2 Function Documentation

atlk_rc_t v2x_emulator_receive (v2x_emulator_t * *emulator*, v2x_if_index_t * *if_index*, v2x_protocol_t * *protocol*, void * *data_ptr*, size_t * *data_size_ptr*, v2x_send_params_t * *params*, const atlk_wait_t * *wait*) Receive V2X frame from emulated V2X service.

See Also

Using wait option.

Parameters

in	*emulator*	V2X emulator
out	*if_index*	Egress MAC interface index
out	*protocol*	V2X protocol descriptor
out	*data_ptr*	Pointer to start of data
in,out	*data_size_ptr*	Maximum (in) and actual (out) data size in bytes
out	*params*	Input parameters of V2X send operation
in	*wait*	Wait specification (optional)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/v2x-emulator/v2x-emulator-over-udp-example.c.

atlk_rc_t v2x_emulator_send (v2x_emulator_t ∗ *emulator*, v2x_if_index_t *if_index*, const v2x_protocol_t ∗ *protocol*, const void ∗ *data_ptr*, size_t *data_size*, const v2x_receive_params_t ∗ *params*, const atlk_wait_t ∗ *wait*) Send V2X frame to emulated V2X service.

See Also

Using wait option.

Parameters

in	*emulator*	V2X emulator
in	*if_index*	Ingress MAC interface index
in	*protocol*	V2X protocol descriptor
in	*data_ptr*	Pointer to start of data
in	*data_size*	Size of data in bytes
in	*params*	Output parameters of V2X receive operation
in	*wait*	Wait specification (optional)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/v2x-emulator/v2x-emulator-over-udp-example.c.

atlk_rc_t v2x_emulator_service_get (v2x_service_t ∗∗ *service_ptr*) Get pointer to V2X emulator service.

Parameters

out	*service_ptr*	V2X service

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/v2x-emulator/v2x-service-user.c.

7.53 atlk/v2x_remote.h File Reference

V2X remote service API.

```
   #include <atlk/sdk.h>
#include <atlk/remote.h>
#include <atlk/v2x.h>
```

Typedefs

- typedef struct
  v2x_remote_service_config v2x_remote_service_config_t

  V2X remote service configuration parameters.

Functions

- atlk_rc_t v2x_remote_service_create (remote_transport_t *transport, const v2x_remote_service_config_t *config, v2x_service_t **service_ptr)

  Create V2X remote service.

7.53.1 Detailed Description

V2X remote service API.

7.53.2 Function Documentation

atlk_rc_t v2x_remote_service_create (remote_transport_t * *transport*, const v2x_remote_service_config_t * *config*, v2x_service_t ** *service_ptr*) Create V2X remote service.

Parameters

in	*transport*	Remote transport instance
in	*config*	V2X remote service configuration (optional)
out	*service_ptr*	V2X service

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

remote-posix/v2x/v2x-example.c.

7.54 atlk/v2x_service.h File Reference

V2X service API.

```
   #include <atlk/sdk.h>
#include <atlk/eui48.h>
#include <atlk/v2x.h>
```

Data Structures

- struct v2x_socket_config_t

  V2X socket configuration.
- struct v2x_send_params_t

  V2X send parameters.
- struct v2x_receive_params_t

  V2X receive parameters.
- struct v2x_sample_subscriber_config_t

  V2X sample subscriber configuration.

Macros

- #define V2X_SOCKET_CONFIG_INIT

  V2X socket configuration default initializer.
- #define V2X_SEND_PARAMS_INIT

  V2X send parameters default initializer.
- #define V2X_RECEIVE_PARAMS_INIT

  V2X receive parameters default initializer.
- #define V2X_SAMPLE_SUBSCRIBER_CONFIG_INIT

  V2X sample subscriber configuration default initializer.

Typedefs

- typedef struct v2x_socket v2x_socket_t

  V2X socket.
- typedef struct
  v2x_sample_subscriber v2x_sample_subscriber_t

  V2X sample subscriber.

Functions

- atlk_rc_t v2x_default_service_get (v2x_service_t **service_ptr)

  Get pointer to default V2X service.
- atlk_rc_t v2x_service_delete (v2x_service_t *service)

  Delete V2X service instance.
- atlk_rc_t v2x_dot4_channel_start (v2x_service_t *service, const v2x_dot4_channel_start_request_t *request, const atlk_wait_t *wait)

  Send IEEE Std 1609.4-2016 service primitive MLMEX-CHSTART.request and receive MLMEX-CHSTART.confirm.
- atlk_rc_t v2x_dot4_channel_end (v2x_service_t *service, const v2x_dot4_channel_end_request_t *request, const atlk_wait_t *wait)

  Send IEEE Std 1609.4-2016 service primitive MLMEX-CHEND.request and receive MLMEX-CHEND.confirm.
- atlk_rc_t v2x_dot4_channel_end_receive (v2x_service_t *service, v2x_dot4_channel_end_indication_t *indication, const atlk_wait_t *wait)

  Receive IEEE Std 1609.4-2016 service primitive MLMEX-CHEND.indication.
- atlk_rc_t v2x_netif_profile_set (v2x_service_t *service, int netif_index, const v2x_netif_profile_t *profile)

  Set network interface's V2X access profile.
- atlk_rc_t v2x_socket_create (v2x_service_t *service, v2x_socket_t **socket_ptr, const v2x_socket_config_t *config)

  Create V2X socket.
- atlk_rc_t v2x_socket_delete (v2x_socket_t *socket)

  Delete V2X socket.
- atlk_rc_t v2x_send (v2x_socket_t *socket, const void *data_ptr, size_t data_size, const v2x_send_params_t *params, const atlk_wait_t *wait)

  Send V2X frame.
- atlk_rc_t v2x_receive (v2x_socket_t *socket, void *data_ptr, size_t *data_size_ptr, v2x_receive_params_t *params, const atlk_wait_t *wait)

  Receive V2X frame.
- atlk_rc_t v2x_sample_subscriber_create (v2x_service_t *service, v2x_sample_subscriber_t **subscriber_ptr, const v2x_sample_subscriber_config_t *config)

  Create V2X sample subscriber.
- atlk_rc_t v2x_sample_subscriber_delete (v2x_sample_subscriber_t *subscriber)

  Delete V2X sample subscriber.
- atlk_rc_t v2x_sample_int32_receive (v2x_sample_subscriber_t *subscriber, int32_t *value, const atlk_wait_t *wait)

  Receive a V2X sample.

7.54.1 Detailed Description

V2X service API.

7.54.2 Function Documentation

atlk_rc_t v2x_default_service_get (v2x_service_t ∗∗ *service_ptr*) Get pointer to default V2X service.

Parameters

out	*service_ptr*	Pointer to V2X service

Note

New implementation of this getter will override default getter (declared as a weak symbol).
Not supported by remote service library.

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/bridge/v2x-udp-bridge-example.c, craton-threadx/dot4/dot4-channel-switching-example.c, craton-threadx/v2x-emulator/v2x-service-user.c, craton-threadx/v2x/v2x-example.c, craton-threadx/wave-ipv6/wave-ipv6-client-example.c, craton-threadx/wave-ipv6/wave-ipv6-example.c, craton-threadx/wave-ipv6/wave-ipv6-server-example.c, and craton-threadx/wlan-driver/traffic-monitor-example.c.

atlk_rc_t v2x_netif_profile_set (v2x_service_t ∗ *service,* int *netif_index,* const v2x_netif_profile_t ∗ *profile*) Set network interface's V2X access profile.

Parameters

in	*service*	V2X service instance
in	*netif_index*	Network interface index
in	*profile*	Network interface V2X access profile

Remarks

The method to obtain `netif_index` that refers to a V2X-enabled network interface is specific to the type of host operating system. For example, on some systems if_nametoindex() could be used.

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/wave-ipv6/wave-ipv6-client-example.c, craton-threadx/wave-ipv6/wave-ipv6-example.c, and craton-threadx/wave-ipv6/wave-ipv6-server-example.c.

atlk_rc_t v2x_receive (v2x_socket_t ∗ *socket,* void ∗ *data_ptr,* size_t ∗ *data_size_ptr,* v2x_receive_params_t ∗ *params,* const atlk_wait_t ∗ *wait*) Receive V2X frame.

See Also

Using wait option.

Parameters

in	*socket*	V2X socket
out	*data_ptr*	Pointer to start of data
in,out	*data_size_ptr*	Maximum (in) and actual (out) data size in bytes
out	*params*	Output parameters of receive operation
in	*wait*	Wait specification (optional)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/bridge/v2x-udp-bridge-example.c, craton-threadx/v2x-emulator/v2x-service-user.c, craton-threadx/v2x/v2x-example.c, craton-threadx/wlan-driver/traffic-monitor-example.c, and remote-posix/v2x/v2x-example.c.

atlk_rc_t v2x_sample_int32_receive (v2x_sample_subscriber_t ∗ *subscriber*, int32_t ∗ *value*, const atlk_wait_t ∗ *wait*) Receive a V2X sample.

Parameters

in	*subscriber*	V2X sample subscriber to delete
out	*value*	Value of the requested sample
in	*wait*	Wait specification (optional)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t v2x_sample_subscriber_create (v2x_service_t ∗ *service*, v2x_sample_subscriber_t ∗∗ *subscriber_ptr*, const v2x_sample_subscriber_config_t ∗ *config*) Create V2X sample subscriber.

Parameters

in	*service*	V2X service instance
out	*subscriber_ptr*	V2X sample subscriber pointer
in	*config*	V2X sample subscriber configuration

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t v2x_sample_subscriber_delete (v2x_sample_subscriber_t ∗ *subscriber*) Delete V2X sample subscriber.

Parameters

in	*subscriber*	V2X sample subscriber to delete

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t v2x_send (v2x_socket_t ∗ *socket*, const void ∗ *data_ptr*, size_t *data_size*, const v2x_send_params_t ∗ *params*, const atlk_wait_t ∗ *wait*) Send V2X frame.

See Also

Using wait option.

Parameters

in	*socket*	V2X socket
in	*data_ptr*	Pointer to start of data
in	*data_size*	Size of data in bytes
in	*params*	Input parameters of send operation
in	*wait*	Wait specification (optional)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/bridge/v2x-udp-bridge-example.c, craton-threadx/dot4/dot4-channel-switching-example.c, craton-threadx/v2x-emulator/v2x-service-user.c, craton-threadx/v2x/v2x-example.c, craton-threadx/wlan-driver/traffic-monitor-example.c, and remote-posix/v2x/v2x-example.c.

atlk_rc_t v2x_service_delete (v2x_service_t ∗ *service*) Delete V2X service instance.

Parameters

in	*service*	V2X service instance

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/bridge/v2x-udp-bridge-example.c, craton-threadx/dot4/dot4-channel-switching-example.c, craton-threadx/v2x-emulator/v2x-service-user.c, craton-threadx/v2x/v2x-example.c, craton-threadx/wlan-driver/traffic-monitor-example.c, and remote-posix/v2x/v2x-example.c.

atlk_rc_t v2x_socket_create (v2x_service_t ∗ *service*, v2x_socket_t ∗∗ *socket_ptr*, const v2x_socket_config_t ∗ *config*) Create V2X socket.

Parameters

in	*service*	V2X service instance
out	*socket_ptr*	V2X socket pointer
in	*config*	V2X socket configuration

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/bridge/v2x-udp-bridge-example.c, craton-threadx/dot4/dot4-channel-switching-example.c, craton-threadx/v2x-emulator/v2x-service-user.c, craton-threadx/v2x/v2x-example.c, craton-threadx/wlan-driver/traffic-monitor-example.c, and remote-posix/v2x/v2x-example.c.

atlk_rc_t v2x_socket_delete (v2x_socket_t ∗ *socket*) Delete V2X socket.

Parameters

in	*socket*	V2X socket to delete

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/bridge/v2x-udp-bridge-example.c, craton-threadx/dot4/dot4-channel-switching-example.c, craton-threadx/v2x-emulator/v2x-service-user.c, craton-threadx/v2x/v2x-example.c, craton-threadx/wlan-driver/traffic-monitor-example.c, and remote-posix/v2x/v2x-example.c.

7.55 atlk/verinfo.h File Reference

Autotalks SDK version information.

7.55.1 Detailed Description

Autotalks SDK version information.

7.56 craton/bootparam.h File Reference

CRATON boot parameter API.

```
   #include <atlk/sdk.h>
#include <craton/check.h>
```

Functions

- atlk_rc_t bootparam_get (const char ∗name, char ∗value, size_t ∗size)

  Get boot parameter value.
- atlk_rc_t bootparam_set (const char ∗name, const char ∗value)

  Set boot parameter value.

7.56.1 Detailed Description

CRATON boot parameter API.

7.56.2 Function Documentation

atlk_rc_t bootparam_get (const char ∗ *name*, char ∗ *value*, size_t ∗ *size*) Get boot parameter value.

Parameters

in	*name*	Boot parameter name
out	*value*	Boot parameter value
in,out	*size*	Size of value

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t bootparam_set (const char ∗ *name*, const char ∗ *value*) Set boot parameter value.

Note

This function is not supported when OS is loaded from U-Boot.

Parameters

in	*name*	Boot parameter name
in	*value*	Boot parameter value

Return values

ATLK_OK	if succeeded
ATLK_E_UNSUPPORTED	if system was loaded from U-Boot

Returns

Error code if failed

7.57 craton/cache.h File Reference

CRATON cache-related definitions.

```
   #include <atlk/sdk.h>
#include <craton/check.h>
```

7.57.1 Detailed Description

CRATON cache-related definitions.

7.58 craton/can_driver.h File Reference

CAN driver API.

```
   #include <atlk/sdk.h>
#include <atlk/can.h>
#include <craton/check.h>
```

Data Structures

- struct can_hw_buffer_t

  CAN HW buffer (direction + ID)
- struct can_hw_buffer_config_t

  CAN HW buffer configuration.

Macros

- #define CAN_HW_NUM_DEVICES 2

  Number of CAN HW devices.
- #define CAN_HW_BUFFER_INIT

  CAN HW buffer default initializer.
- #define CAN_HW_NUM_BUFFERS 15

  Number of CAN HW buffers.
- #define CAN_HW_BUFFER_CONFIG_INIT

  CAN HW buffer configuration default initializer.
- #define CAN_HW_ID(id)

  Convert CAN ID to CAN HW ID.
- #define CAN_HW_MASK(type, mask)

  Convert CAN ID mask to CAN HW ID mask.
- #define CAN_HW_MASK_SELECT_ALL 0x0

  CAN HW ID mask used to select all.

Typedefs

- typedef uint32_t can_hw_id_t

  CAN HW ID.
- typedef int(∗ can_filter_callback_t)(can_device_id_t device_id, can_id_t id)

  CAN SW filter callback.

Enumerations

- enum can_hw_buffer_direction_t { CAN_HW_BUFFER_DIRECTION_NA = 0, CAN_HW_BUFFER_DIRECTION_RX = 1, CAN_HW_BUFFER_DIRECTION_TX = 2 }

  CAN HW buffer direction.
- enum can_speed_t {
  CAN_SPEED_33_KBPS = 0, CAN_SPEED_250_KBPS = 1, CAN_SPEED_500_KBPS = 2, CAN_SPEED_1_MBPS = 3,
  CAN_SPEED_125_KBPS = 4 }

  CAN driver speed.

Functions

- atlk_rc_t can_hw_buffer_config_set (can_device_id_t device_id, const can_hw_buffer_config_t ∗config)

  Set CAN HW buffer configuration.
- atlk_rc_t can_hw_buffer_config_get (can_device_id_t device_id, can_hw_buffer_config_t ∗config)

  Get CAN HW buffer configuration.
- atlk_rc_t can_isr_filter_callback_set (can_device_id_t device_id, can_filter_callback_t callback)

  Set CAN SW filter callback.
- atlk_rc_t can_hw_speed_set (can_device_id_t device_id, can_speed_t speed)

  Set CAN device speed.
- atlk_rc_t can_hw_speed_get (can_device_id_t device_id, can_speed_t ∗speed)

  Get CAN device speed.

7.58.1 Detailed Description

CAN driver API. Reference:

- IPextreme Controller Area Network (CAN) Controller User Guide v1.1.0

7.58.2 Macro Definition Documentation

#define CAN_HW_ID(*id*) **Value:**

```
(((id) & (1 << CAN_ID_EXTENDED_BIT)) ?                              \
  ((((id) & 0x1ffc0000) << (21 - 18)) | (1 << 20) | (1 << 19) |        \
  (((id) & 0x3ffff) << 1) |                                            \
  (((id) & (1 << CAN_ID_RTR_BIT) ? 1 << 20 : 0))) :                   \
  ((((id) & 0x7ff) << 21) | ((id) & (1 << CAN_ID_RTR_BIT) ? 1 << 20 : 0)))
```

Convert CAN ID to CAN HW ID.

Parameters

in	*id*	CAN ID in can_id_t format

Returns

CAN HW ID in can_hw_id_t format

#define CAN_HW_MASK(*type, mask*) **Value:**

```
((type) ? ((((mask) & 0x1ffc0000) << (21 - 18)) |                    \
  (((mask) & 0x3ffff) << 1) |                                          \
  (((mask) & (1 << CAN_ID_RTR_BIT) ? 1 : 0)) |                        \
  (((mask) & (1 << CAN_ID_EXTENDED_BIT) ? 1 << 19: 0))) :             \
  ((((mask) & 0x7ff) << 21) |                                          \
  (((mask) & (1 << CAN_ID_RTR_BIT) ? 1 << 20 : 0)) |                  \
  (((mask) & (1 << CAN_ID_EXTENDED_BIT) ? 1 << 19: 0))))
```

Convert CAN ID mask to CAN HW ID mask.

Parameters

in	*type*	Base format (0) or extended format (1)
in	*mask*	CAN ID mask in can_id_t format

Returns

CAN HW ID mask in can_hw_id_t format

#define CAN_HW_MASK_SELECT_ALL 0x0 CAN HW ID mask used to select all.

Use this mask to specify that all bits in CAN HW ID should be considered when filtering.

7.58.3 Typedef Documentation

typedef int(∗ can_filter_callback_t)(can_device_id_t device_id, can_id_t id) CAN SW filter callback.

Parameters

in	*device_id*	CAN device ID
in	*id*	CAN ID of received frame

Return values

0	indicates frame should be discarded
1	indicates frame should be accepted

typedef uint32_t can_hw_id_t CAN HW ID.

Memory registers `ID1` (MSB) and `ID0` (LSB), see Appendix A, Table 25.

7.58.4 Enumeration Type Documentation

enum can_hw_buffer_direction_t CAN HW buffer direction.

Enumerator

CAN_HW_BUFFER_DIRECTION_NA	Value indicating that CAN HW buffer direction is N/A.
CAN_HW_BUFFER_DIRECTION_RX	Ingress CAN HW buffer.
CAN_HW_BUFFER_DIRECTION_TX	Egress CAN HW buffer.

enum can_speed_t CAN driver speed.

Enumerator

CAN_SPEED_33_KBPS	Speed of 33 Kbps.
CAN_SPEED_250_KBPS	Speed of 250 Kbps.
CAN_SPEED_500_KBPS	Speed of 500 Kbps.
CAN_SPEED_1_MBPS	Speed of 1 Mbps.
CAN_SPEED_125_KBPS	Speed of 125 Kbps.

7.58.5 Function Documentation

atlk_rc_t can_hw_buffer_config_get (can_device_id_t *device_id*, can_hw_buffer_config_t ∗ *config*) Get CAN HW buffer configuration.

Note

In masks, 0s denote 'care' and 1s denote 'dont care'.

Parameters

in	*device_id*	CAN device ID
out	*config*	CAN HW configuration

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t can_hw_buffer_config_set (can_device_id_t *device_id*, const can_hw_buffer_config_t ∗ *config*) Set CAN HW buffer configuration.

Note

In masks, 0s denote 'care' and 1s denote 'dont care'.

Parameters

in	*device_id*	CAN device ID
in	*config*	CAN HW configuration

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/can/can-hw-filter-example.c.

atlk_rc_t can_hw_speed_get (can_device_id_t *device_id*, can_speed_t ∗ *speed*) Get CAN device speed.

Parameters

in	*device_id*	CAN device ID
out	*speed*	CAN speed

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t can_hw_speed_set (can_device_id_t *device_id*, can_speed_t *speed*) Set CAN device speed.

Parameters

`in`	*device_id*	CAN device ID
`in`	*speed*	CAN speed

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t can_isr_filter_callback_set (can_device_id_t *device_id*, can_filter_callback_t *callback*) Set CAN SW filter callback.

Setting callback to NULL means accept all.

Warning

User callback is called at ISR context and should be handled as such.

Parameters

`in`	*device_id*	CAN device ID
`in`	*callback*	CAN SW filter callback

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.59 craton/cc3100_driver.h File Reference

TI CC3100 Wi-Fi driver configuration.

```
#include <atlk/sdk.h>
#include <atlk/os.h>
#include <craton/spi_driver.h>
#include <craton/hdmac_driver.h>
#include <craton/gpio.h>
#include <craton/check.h>
```

Data Structures

- struct cc3100_config_t

  CC3100 configuration parameters.

Macros

- #define CC3100_CONFIG_INIT

  CC3100 configuration parameters default init.

Functions

- atlk_rc_t cc3100_init (const cc3100_config_t *config)

  Initialize CC3100.

7.59.1 Detailed Description

TI CC3100 Wi-Fi driver configuration.

7.59.2 Function Documentation

atlk_rc_t cc3100_init (const cc3100_config_t * *config*) Initialize CC3100.
Parameters

in	*config*	CC3100 configuration parameters

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.60 craton/check.h File Reference

CRATON build environment check.

7.60.1 Detailed Description

CRATON build environment check.

7.61 craton/cli.h File Reference

CRATON CLI instances API.
```
   #include <atlk/sdk.h>
#include <craton/check.h>
```

Typedefs

- typedef struct cli_def cli_instance_t

  CLI instance.

Enumerations

- enum cli_instance_type_t {
  CLI_INSTANCE_TYPE_UART = 0, CLI_INSTANCE_TYPE_TELNET = 1, CLI_INSTANCE_TYPE_TELNET2 = 2,
  CLI_INSTANCE_TYPE_MIN = CLI_INSTANCE_TYPE_UART,
  CLI_INSTANCE_TYPE_MAX = CLI_INSTANCE_TYPE_TELNET2 }

  CLI instance type.

Functions

- atlk_rc_t cli_instance_get (cli_instance_t **cli_ptr, cli_instance_type_t type)

  Get pointer to CLI instance.
- atlk_rc_t cli_suspend (cli_instance_type_t type)

  Suspend CLI instance.
- atlk_rc_t cli_resume (cli_instance_type_t type)

  Resume CLI instance.

7.61.1 Detailed Description

CRATON CLI instances API.

7.61.2 Enumeration Type Documentation

enum cli_instance_type_t CLI instance type.

Enumerator

CLI_INSTANCE_TYPE_UART	CLI is accessible over UART.
CLI_INSTANCE_TYPE_TELNET	CLI is accessible over Telnet, using well-known port 23.
CLI_INSTANCE_TYPE_TELNET2	CLI is accessible over Telnet, using port 1123.
CLI_INSTANCE_TYPE_MIN	Minimum CLI instance.
CLI_INSTANCE_TYPE_MAX	Maximum CLI instance.

7.61.3 Function Documentation

atlk_rc_t cli_instance_get (cli_instance_t ** *cli_ptr*, cli_instance_type_t *type*) Get pointer to CLI instance.

Parameters

out	*cli_ptr*	CLI instance
in	*type*	CLI instance type

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/bt-spi2uart/bt-spi2uart-example.c, craton-threadx/cli/cli-example.c, and craton-threadx/v2x-emulator/v2x-emulator-over-udp-example.c.

atlk_rc_t cli_resume (cli_instance_type_t *type*) Resume CLI instance.

Parameters

in	*type*	CLI instance type

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t cli_suspend (cli_instance_type_t *type*) Suspend CLI instance.

Parameters

in	*type*	CLI instance type

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.62 craton/debug.h File Reference

CRATON debug API.

```
#include <craton/exception.h>
#include <craton/check.h>
```

Functions

- void debug_printf (const char *format,...)

  Print formatted message to debug console.
- void debug_thread_state_print (void)

  Print debug message describing thread state of calling function thread.
- void debug_exception_info_print (const exception_info_t *info)

  Dump exception info to debug console.

7.62.1 Detailed Description

CRATON debug API.

7.62.2 Function Documentation

void debug_exception_info_print (const exception_info_t * *info*) Dump exception info to debug console.

The dumped data is a pre-formatted text message containing information extracted from `info` structure.

Parameters

in	*info*	Exception information

void debug_printf (const char * *format*, ...) Print formatted message to debug console.

Parameters

in	*format*	printf() compatible format string

void debug_thread_state_print (void) Print debug message describing thread state of calling function thread.

Prints stack dump of calling function thread, along with other registers values.

Debug message will be written to console.

7.63 craton/duid.h File Reference

Device Unique ID API.

```
#include <atlk/sdk.h>
#include <craton/check.h>
```

Macros

- #define DUID_MAX_LEN 32

  Maximum length of Device Unique ID in octets.

Functions

- atlk_rc_t duid_get (void *buf, size_t buf_len)

  Get Device Unique ID (DUID).

7.63.1 Detailed Description

Device Unique ID API.

7.63.2 Function Documentation

atlk_rc_t duid_get (void * *buf*, size_t *buf_len*) Get Device Unique ID (DUID).

`buf_len` must be less or equal to DUID_MAX_LEN.

Note

If an engineering sample of CRATON is used, this function invokes SHA256-based KDF2 on boot parameter `duid_str` and the result is returned as DUID.

Parameters

out	*buf*	Buffer to store the DUID
in	*buf_len*	Length of desired DUID

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.64 craton/exception.h File Reference

CRATON CPU exception API.

```
#include <atlk/sdk.h>
#include <craton/exception_arc.h>
#include <craton/exception_arm.h>
#include <craton/check.h>
```

7.64.1 Detailed Description

CRATON CPU exception API.

7.65 craton/exception_arc.h File Reference

CRATON ARC CPU exception API.

```
#include <stdint.h>
#include <craton/check.h>
```

Data Structures

- struct exception_arc_regs_t
  ARC exception registers.
- struct exception_arc_info_t
  ARC exception structure containing all necessary information upon exception.

Macros

- #define EXC_ARC_THREAD_STACK_MAX_SIZE 512
  Maximum size of thread stack upon exception.
- #define EXC_ARC_THREAD_NAME_MAX_SIZE 128
  Maximum size of thread name.

Enumerations

- enum exception_arc_type_t { EXC_ARC_TYPE_MEMORY = 0, EXC_ARC_TYPE_INSTRUCTION = 1, EXC_ARC_TYPE_WD_TIMER = 2, EXC_ARC_TYPE_ABNORMAL_EXIT = 3 }
  ARC exception type.

7.65.1 Detailed Description

CRATON ARC CPU exception API.

7.65.2 Enumeration Type Documentation

enum exception_arc_type_t ARC exception type.

Enumerator

EXC_ARC_TYPE_MEMORY Memory exception.

EXC_ARC_TYPE_INSTRUCTION Instruction exception.

EXC_ARC_TYPE_WD_TIMER Watchdog expiration exception.

EXC_ARC_TYPE_ABNORMAL_EXIT Abnormal exit - bad return code from ARC.

7.66 craton/exception_arm.h File Reference

CRATON ARM CPU exception API.

```
   #include <stdint.h>
#include <craton/check.h>
```

Data Structures

- struct exception_arm_regs_t

  ARM exception registers.
- struct exception_arm_info_t

  ARM exception structure containing all necessary information upon exception.

Macros

- #define EXC_ARM_THREAD_STACK_MAX_SIZE 512

  Maximum size of thread stack upon exception.
- #define EXC_ARM_THREAD_NAME_MAX_SIZE 128

  Maximum size of thread name.

Enumerations

- enum exception_arm_fault_operation_t {
  EXC_ARM_FAULT_OP_READ = 0, EXC_ARM_FAULT_OP_WRITE = 1, EXC_ARM_FAULT_OP_FETCH = 2, EX-C_ARM_FAULT_OP_EXECUTE = 3,
  EXC_ARM_FAULT_OP_NA = 4 }

  ARM exception fault operation.
- enum exception_arm_type_t {
  EXC_ARM_TYPE_DATA = 0, EXC_ARM_TYPE_PREFETCH = 1, EXC_ARM_TYPE_INSTRUCTION = 2, EXC_A-RM_TYPE_WD_TIMER = 3,
  EXC_ARM_TYPE_NA = 4 }

  ARM exception type.
- enum exception_arm_reason_t {
  EXC_ARM_REASON_BACKGROUND = 0, EXC_ARM_REASON_ALIGNMENT = 1, EXC_ARM_REASON_DEBUG-_EVENT = 2, EXC_ARM_REASON_SYNC_EXTERNAL = 3,
  EXC_ARM_REASON_ASYNC_EXTERNAL = 4, EXC_ARM_REASON_PERMISSION = 5, EXC_ARM_REASON_AS-YNC_ECC = 6, EXC_ARM_REASON_SYNC_ECC = 7,
  EXC_ARM_REASON_ABNORMAL_EXIT = 8, EXC_ARM_REASON_NA = 9 }

  ARM exception fault reason.

7.66.1 Detailed Description

CRATON ARM CPU exception API.

7.66.2 Enumeration Type Documentation

enum exception_arm_fault_operation_t ARM exception fault operation.

Enumerator

EXC_ARM_FAULT_OP_READ Fault upon read operation.

EXC_ARM_FAULT_OP_WRITE Fault upon write operation.

EXC_ARM_FAULT_OP_FETCH Fault upon pre-fetch operation.

EXC_ARM_FAULT_OP_EXECUTE Fault upon execution operation.

EXC_ARM_FAULT_OP_NA Unavailable operation upon fault.

enum exception_arm_reason_t ARM exception fault reason.

Enumerator

EXC_ARM_REASON_BACKGROUND Background MPU exception reason: access to undefined memory area in MPU.

EXC_ARM_REASON_ALIGNMENT Unaligned memory access exception reason.

EXC_ARM_REASON_DEBUG_EVENT Debug exception reason when in debug mode.

EXC_ARM_REASON_SYNC_EXTERNAL Synchronous external abort exception reason.

EXC_ARM_REASON_ASYNC_EXTERNAL Asynchronous external abort exception reason.

EXC_ARM_REASON_PERMISSION Permission exception reason.

EXC_ARM_REASON_ASYNC_ECC Asynchronous Parity/Error Correction Code (ECC) exception reason.

EXC_ARM_REASON_SYNC_ECC Synchronous Parity/Error Correction Code (ECC) exception reason.

EXC_ARM_REASON_ABNORMAL_EXIT Abnormal exit - Caused by assert/abort/BUG...

EXC_ARM_REASON_NA Unavailable exception reason.

enum exception_arm_type_t ARM exception type.

Enumerator

EXC_ARM_TYPE_DATA Data exception.

EXC_ARM_TYPE_PREFETCH Prefetch exception.

EXC_ARM_TYPE_INSTRUCTION Instruction exception.

EXC_ARM_TYPE_WD_TIMER Watchdog expiration exception.

EXC_ARM_TYPE_NA Unavailable exception type.

7.67 craton/fs.h File Reference

File system.

```
   #include <atlk/sdk.h>
#include <craton/check.h>
```

Data Structures

- struct fs_dirstat

  Directory statistics structure.

Macros

- #define FS_OPEN_FILES_MAX 127
  Maximum number of open files.
- #define FS_DIRSTAT_DEPTH_MAX 8
  Maximum directory traversal depth.
- #define FS_WRITE_BUFFER_SIZE 1024
  Write buffer size.
- #define FS_READ_WRITE_SIZE_MAX 0x7FFFFFFF
  Maximum data size for read/write operations.

Functions

- int fs_dirstat (const char ∗dirpath, struct fs_dirstat ∗dirstat, unsigned int depth_limit)
  Obtain directory statistics.
- void fs_sync (void)
  Sync file system.
- int fs_thread_enable (void)
  Enable thread to work with file system.

7.67.1 Detailed Description

File system.

7.67.2 Function Documentation

int fs_dirstat (const char ∗ *dirpath*, struct fs_dirstat ∗ *dirstat*, unsigned int *depth_limit*) Obtain directory statistics.

Parameters

`in`	*dirpath*	Directory path
`in`	*dirstat*	Directory statistics structure
`in`	*depth_limit*	Maximun depth of traversal, must not exceed FS_DIRSTAT_DEPTH_MAX

Return values

0	if succeeded

Returns

-1 if failed and set errno appropriately

Remarks

: Supported on ARM core only.

Examples:

craton-threadx/fs/fs-example.c.

void fs_sync (void) Sync file system.

Flushes all open files in file system to media (Flash or microSD).

Remarks

Supported on ARM core only.

int fs_thread_enable (void) Enable thread to work with file system.

This function should be called in every thread created via tx_thread_create which requires file system access. There is no need to call it for threads created via pthread_create.

Calling this function more than once is safe.

Return values

0	if succeeded

Returns

-1 if failed and set errno appropriately

Remarks

: Supported on ARM core only.

Examples:

craton-threadx/firmware/fw-update-example.c, and craton-threadx/gnss-teseo/gnss-teseo-fw-update-example.c.

7.68 craton/fw_rbi.h File Reference

Firmware verification API.

```
#include <craton/check.h>
```

Enumerations

- enum fw_rbi_verify_result_t { FW_RBI_VERIFY_OK = 0, FW_RBI_VERIFY_E_PUBLIC_KEY_SIGNATURE = 1, FW_-RBI_VERIFY_E_IMAGE_SIGNATURE = 2, FW_RBI_VERIFY_E_INVALID_ARG = 3 }

  Firmware verification result.

Functions

- fw_rbi_verify_result_t fw_rbis_verify (const void ∗image, size_t image_size)

  Verify signed image.

7.68.1 Detailed Description

Firmware verification API.

7.68.2 Enumeration Type Documentation

enum fw_rbi_verify_result_t Firmware verification result.

Enumerator

FW_RBI_VERIFY_OK	Firmware verified.
FW_RBI_VERIFY_E_PUBLIC_KEY_SIGNATURE	Invalid OEM public key signature.
FW_RBI_VERIFY_E_IMAGE_SIGNATURE	Invalid image signature.
FW_RBI_VERIFY_E_INVALID_ARG	Invalid argument.

7.68.3 Function Documentation

fw_rbi_verify_result_t fw_rbis_verify (const void ∗ *image*, size_t *image_size*) Verify signed image.

Parameters

in	*image*	Signed image to be verified
in	*image_size*	Image size in bytes

Returns

FW_RBI_VERIFY_OK if image verification succeeded, otherwise return ::FW_RBI_VERIFY_E_∗

7.69 craton/fw_uimage.h File Reference

CRATON firmware validation.

```
   #include <atlk/sdk.h>
#include <craton/check.h>
```

Functions

- int fw_uimage_valid (const void ∗uimage, size_t uimage_size)

  Check whether CRATON firmware uimage is valid.

7.69.1 Detailed Description

CRATON firmware validation.

7.69.2 Function Documentation

int fw_uimage_valid (const void ∗ *uimage*, size_t *uimage_size*) Check whether CRATON firmware uimage is valid.

Note

Doesn't verify cryptographic signature!

Return values

:	1 if firmware uimage is valid, 0 otherwise

Examples:

craton-threadx/firmware/fw-update-example.c.

7.70 craton/gpio.h File Reference

CRATON GPIO definitions.

```
   #include <atlk/sdk.h>
#include <craton/check.h>
```

Macros

- #define GPIO_NUM_MIN 0

  Minimum GPIO number.
- #define GPIO_NUM_MAX 31

  Maximum GPIO number.
- #define GPIO_NUM_NA 0xff

  Value indicating that GPIO number is N/A.

Typedefs

- typedef uint8_t gpio_num_t

  GPIO number.

7.70.1 Detailed Description

CRATON GPIO definitions.

7.71 craton/gpio_driver.h File Reference

CRATON GPIO driver API.

```
   #include <atlk/sdk.h>
#include <craton/gpio.h>
#include <craton/check.h>
```

Typedefs

- typedef void(∗ gpio_irq_handler_t)(gpio_num_t gpio_num, void ∗context)

  GPIO IRQ handler.

Enumerations

- enum gpio_dir_t { GPIO_INPUT = 0, GPIO_OUTPUT = 1 }

  GPIO direction.
- enum gpio_sense_t { GPIO_SENSE_EDGE = 0, GPIO_SENSE_LEVEL = 1 }

  GPIO IRQ sense mode.
- enum gpio_edge_t { GPIO_EDGE_ONE = 0, GPIO_EDGE_BOTH = 1 }

  GPIO edge mode.
- enum gpio_event_t { GPIO_EVENT_FALLING_OR_LOW = 0, GPIO_EVENT_RISING_OR_HIGH = 1 }

  GPIO IRQ event mode.
- enum gpio_mask_t { GPIO_MASKED = 0, GPIO_UNMASKED = 1 }

  GPIO IRQ mask status.

Functions

- atlk_rc_t gpio_dir_set (gpio_num_t gpio_num, gpio_dir_t direction)

  Set GPIO direction.
- atlk_rc_t gpio_dir_get (gpio_num_t gpio_num, gpio_dir_t ∗direction)

  Get GPIO direction.
- atlk_rc_t gpio_irq_sense_set (gpio_num_t gpio_num, gpio_sense_t sense)

  Set interrupt sense.
- atlk_rc_t gpio_irq_edge_set (gpio_num_t gpio_num, gpio_edge_t edge)

  Set interrupt both-edges register.
- atlk_rc_t gpio_irq_event_set (gpio_num_t gpio_num, gpio_event_t event)

  Set event register.
- atlk_rc_t gpio_irq_mask_set (gpio_num_t gpio_num, gpio_mask_t mask)

  Set interrupt mask register.
- atlk_rc_t gpio_irq_clear (gpio_num_t gpio_num)

  Clear IRQ register.
- atlk_rc_t gpio_data_set (gpio_num_t gpio_num, int data)

  Set GPIO level.
- atlk_rc_t gpio_data_get (gpio_num_t gpio_num, int ∗data)

  Get GPIO level.
- atlk_rc_t gpio_irq_status_get (gpio_num_t gpio_num, int ∗status)

  Get IRQ status.
- atlk_rc_t gpio_irq_handler_set (gpio_num_t gpio_num, gpio_irq_handler_t handler, void ∗context)

  Set GPIO IRQ handler.

7.71.1 Detailed Description

CRATON GPIO driver API.

7.71.2 Enumeration Type Documentation

enum gpio_dir_t GPIO direction.

Enumerator

GPIO_INPUT Treat pin as input.

GPIO_OUTPUT Drive pin as output.

enum gpio_edge_t GPIO edge mode.

Enumerator

GPIO_EDGE_ONE Detect one edge.

GPIO_EDGE_BOTH Detect both edges.

enum gpio_event_t GPIO IRQ event mode.

Enumerator

GPIO_EVENT_FALLING_OR_LOW Detect falling edge or low level.

GPIO_EVENT_RISING_OR_HIGH Detect rising edge or high level.

enum gpio_mask_t GPIO IRQ mask status.

Enumerator

GPIO_MASKED Masked.

GPIO_UNMASKED Unmasked.

enum gpio_sense_t GPIO IRQ sense mode.

Enumerator

GPIO_SENSE_EDGE Detect edge.

GPIO_SENSE_LEVEL Detect level.

7.71.3 Function Documentation

atlk_rc_t gpio_data_get (gpio_num_t *gpio_num*, int ∗ *data*) Get GPIO level.

Note

This function should only be used if GPIO is configured as input.

Parameters

`in`	*gpio_num*	GPIO number
`out`	*data*	0 or 1

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t gpio_data_set (gpio_num_t *gpio_num*, int *data*) Set GPIO level.

Note

This function should only be used if GPIO is configured as output.

Parameters

in	*gpio_num*	GPIO number
in	*data*	0 or 1

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t gpio_dir_get (gpio_num_t *gpio_num*, gpio_dir_t * *direction*) Get GPIO direction.

Parameters

in	*gpio_num*	GPIO number
out	*direction*	GPIO direction

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t gpio_dir_set (gpio_num_t *gpio_num*, gpio_dir_t *direction*) Set GPIO direction.

Parameters

in	*gpio_num*	GPIO number
in	*direction*	GPIO direction

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t gpio_irq_clear (gpio_num_t *gpio_num*) Clear IRQ register.

Parameters

in	*gpio_num*	GPIO number

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t gpio_irq_edge_set (gpio_num_t *gpio_num*, gpio_edge_t *edge*) Set interrupt both-edges register.

Note

If `edge` is set to GPIO_EDGE_ONE then gpio_irq_event_set should be used to configure the desired edge.

Parameters

in	*gpio_num*	gpio number
in	*edge*	GPIO edge mode

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t gpio_irq_event_set (gpio_num_t *gpio_num*, gpio_event_t *event*) Set event register.

Parameters

in	*gpio_num*	GPIO number
in	*event*	IRQ event mode

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t gpio_irq_handler_set (gpio_num_t *gpio_num*, gpio_irq_handler_t *handler*, void ∗ *context*) Set GPIO IRQ handler.

Parameters

in	*gpio_num*	GPIO number
in	*handler*	GPIO IRQ handler
in	*context*	GPIO handler context

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t gpio_irq_mask_set (gpio_num_t *gpio_num*, gpio_mask_t *mask*) Set interrupt mask register.

Parameters

in	*gpio_num*	GPIO number
in	*mask*	IRQ mask status

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t gpio_irq_sense_set (gpio_num_t *gpio_num*, gpio_sense_t *sense*) Set interrupt sense.

Note

If `sense` is set to GPIO_SENSE_LEVEL then gpio_irq_event_set should be used to configure the desired level.

Parameters

in	*gpio_num*	GPIO number
in	*sense*	IRQ sense mode

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t gpio_irq_status_get (gpio_num_t *gpio_num*, int ∗ *status*) Get IRQ status.

Parameters

in	*gpio_num*	GPIO number
out	*status*	IRQ status (0 or 1)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.72 craton/hdmac.h File Reference

CRATON HDMAC (DMA controller) definitions.

```
   #include <atlk/sdk.h>
#include <craton/check.h>
```

Macros

- #define HDMAC_CHANNEL_ID_MIN 0

  HDMAC minimum valid channel ID.
- #define HDMAC_CHANNEL_ID_MAX 7

  HDMAC maximum valid channel ID.
- #define HDMAC_CHANNEL_ID_NA 0xff

  Value indicating that HDMAC channel ID is N/A.

Typedefs

- typedef uint8_t hdmac_channel_id_t

  HDMAC channel ID.

7.72.1 Detailed Description

CRATON HDMAC (DMA controller) definitions.

7.73 craton/i2c_driver.h File Reference

CRATON I2C driver API.

```
   #include <atlk/sdk.h>
#include <craton/check.h>
```

Data Structures

- struct i2c_driver_config_t

  I2C initialization configuration descriptor.

Macros

- #define I2C_CLOCK_DIVISOR_NA 0

  Value indicating that I2C clock divisor is N/A.
- #define I2C_CLOCK_DIVISOR_MIN 32

  I2C clock divisor minimum value.
- #define I2C_CLOCK_DIVISOR_MAX 2044

  I2C clock divisor maximum value.
- #define I2C_CLOCK_DIVISOR_STEP 4

  I2C clock divisor step.
- #define I2C_DRIVER_CONFIG_INIT

  I2C driver configuration default initializer.
- #define I2C_F_NO_STOP (1 << 0)

  I2C flag: no stop condition.

Typedefs

- typedef uint16_t i2c_slave_address_t

  I2C slave address.

Functions

- atlk_rc_t i2c_driver_init (const i2c_driver_config_t *config)

  Initialize I2C driver.
- atlk_rc_t i2c_pio_send (i2c_slave_address_t address, const void *data_ptr, size_t data_size, unsigned int flags)

  Send data to a slave device.
- atlk_rc_t i2c_pio_receive (i2c_slave_address_t address, void *data_ptr, size_t data_size, unsigned int flags)

  Receive data from a slave device.

7.73.1 Detailed Description

CRATON I2C driver API.

7.73.2 Function Documentation

atlk_rc_t i2c_driver_init (const i2c_driver_config_t * *config*) Initialize I2C driver.

Remarks

This is the first function to be called prior to using I2C.

Parameters

in	*config*	I2C driver configuration

Return values

ATLK_OK	for successful operation

Returns

Error code if failed

atlk_rc_t i2c_pio_receive (i2c_slave_address_t *address*, void ∗ *data_ptr*, size_t *data_size*, unsigned int *flags*) Receive data from a slave device.

Remarks

This function doesn't utilize DMA ("PIO" stands for processor I/O).
This function returns only after all data has been transferred.

Parameters

in	*address*	Slave device address
out	*data_ptr*	Buffer for data to be received
in	*data_size*	Size in bytes of receive buffer
in	*flags*	I2C flags

Return values

ATLK_OK	for successful operation

Returns

Error code if failed

atlk_rc_t i2c_pio_send (i2c_slave_address_t *address*, const void ∗ *data_ptr*, size_t *data_size*, unsigned int *flags*) Send data to a slave device.

Remarks

This function doesn't utilize DMA ("PIO" stands for processor I/O).
This function returns only after all data has been transferred.

Parameters

in	*address*	Slave device address
in	*data_ptr*	Data to be sent
in	*data_size*	Size in bytes of data to be sent
in	*flags*	I2C flags

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.74 craton/i2s_driver.h File Reference

CRATON I2S driver API.

```
  #include <atlk/sdk.h>
#include <craton/hdmac.h>
#include <craton/gpio.h>
#include <craton/check.h>
```

Data Structures

- struct i2s_driver_config_t

  I2S Driver configuration.
- struct i2s_dma_playback_t

  I2S DMA playback descriptor.

Macros

- #define I2S_MAX_SAMPLE_BUFFER_SIZE 262080
  Maximum size of audio sample buffer in bytes.
- #define I2S_DRIVER_CONFIG_INIT
  I2S Driver configuration default initializer.
- #define I2S_DMA_PLAYBACK_INIT
  I2S DMA playback descriptor default initializer.

Enumerations

- enum i2s_playback_status_t {
  I2S_PLAYBACK_NOT_STARTED = 0, I2S_PLAYBACK_IN_PROGRESS = 1, I2S_PLAYBACK_COMPLETE = 2,
  I2S_PLAYBACK_INTERRUPTED = 3,
  I2S_PLAYBACK_FAILED = 4 }
  Playback status codes.

Functions

- atlk_rc_t i2s_driver_init (const i2s_driver_config_t ∗config)
  Initialize the I2S driver.
- atlk_rc_t i2s_dma_playback_start (i2s_dma_playback_t ∗playback)
  Start DMA playback of audio sample buffers.

7.74.1 Detailed Description

CRATON I2S driver API.

7.74.2 Enumeration Type Documentation

enum i2s_playback_status_t Playback status codes.

Enumerator

I2S_PLAYBACK_NOT_STARTED	Playback not started.
I2S_PLAYBACK_IN_PROGRESS	Playback in progress.
I2S_PLAYBACK_COMPLETE	Playback complete.
I2S_PLAYBACK_INTERRUPTED	Playback interrupted by user.
I2S_PLAYBACK_FAILED	Playback failed.

7.74.3 Function Documentation

atlk_rc_t i2s_dma_playback_start (i2s_dma_playback_t ∗ *playback*) Start DMA playback of audio sample buffers.

If playback is already in progress, it will be stopped.

The audio samples should be two's complement signed 8-bit PCM with 20 kHz sample rate.

Parameters

`in,out`	*DMA*	playback descriptor

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/i2s/i2s-example.c.

atlk_rc_t i2s_driver_init (const i2s_driver_config_t ∗ *config*) Initialize the I2S driver.

Parameters

in	*config*	Driver configuration parameters

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.75 craton/imq.h File Reference

IMQ API.

```
#include <atlk/sdk.h>
#include <craton/check.h>
```

Data Structures

- struct imq_socket_t
  IMQ socket.
- struct imq_queue_config_t
  IMQ queue configuration.
- struct imq_service_config_t
  IMQ service configuration.

Macros

- #define IMQ_ADDRESS_NA UINT16_MAX
  Value indicating that IMQ address is N/A.
- #define IMQ_ADDRESS_MAX 31
  IMQ address maximum value.
- #define IMQ_SOCKET_INIT
  IMQ socket default initializer.
- #define IMQ_QUEUE_CONFIG_INIT
  IMQ queue configuration default initializer.
- #define IMQ_SERVICE_NAME_LENGTH_MAX 16
  Maximum length for IMQ socket name.
- #define IMQ_SERVICE_CONFIG_INIT
  IMQ service configuration default initializer.

Typedefs

- typedef struct imq_config imq_config_t
  IMQ configuration parameters.
- typedef uint16_t imq_address_t
  IMQ address.

Functions

- atlk_rc_t imq_init (const imq_config_t *config)
  Initialize IMQ on a specific CPU.
- atlk_rc_t imq_bind (imq_socket_t *socket, imq_address_t address)
  Bind IMQ socket.
- atlk_rc_t imq_listen (imq_socket_t *socket, const imq_service_config_t *config)

Listen for incoming IMQ connections.

- atlk_rc_t imq_accept (imq_socket_t ∗socket, imq_socket_t ∗connected_socket, const atlk_wait_t ∗wait)

  Accept IMQ connection.
- atlk_rc_t imq_connect (imq_socket_t ∗socket, imq_address_t address, const atlk_wait_t ∗wait)

  Connect to IMQ server.
- atlk_rc_t imq_send (imq_socket_t ∗socket, const void ∗data_ptr, size_t data_size, const atlk_wait_t ∗wait)

  Send IMQ message.
- atlk_rc_t imq_fragmented_send (imq_socket_t ∗socket, const atlk_const_fragment_t ∗fragment_array_ptr, size_t fragment_array_size, const atlk_wait_t ∗wait)

  Send IMQ scatter-gather message.
- atlk_rc_t imq_receive (imq_socket_t ∗socket, void ∗data_ptr, size_t ∗data_size_ptr, const atlk_wait_t ∗wait)

  Receive IMQ message.
- atlk_rc_t imq_fragmented_receive (imq_socket_t ∗socket, const atlk_fragment_t ∗fragment_array_ptr, size_t fragment_array_size, size_t ∗data_size_ptr, const atlk_wait_t ∗wait)

  Receive IMQ message into a scatter-gather buffer.
- atlk_rc_t imq_close (imq_socket_t ∗socket)

  Close IMQ socket.

7.75.1 Detailed Description

IMQ API.

7.75.2 Macro Definition Documentation

#define IMQ_SERVICE_NAME_LENGTH_MAX 16 Maximum length for IMQ socket name.

7.75.3 Typedef Documentation

typedef struct imq_config imq_config_t IMQ configuration parameters.

7.75.4 Function Documentation

atlk_rc_t imq_accept (imq_socket_t ∗ *socket,* imq_socket_t ∗ *connected_socket,* const atlk_wait_t ∗ *wait*) Accept IMQ connection.

See Also

Using wait option.

Parameters

in	*socket*	IMQ socket
out	*connected_socket*	Connected IMQ socket
in	*wait*	Wait specification (optional)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/imq/imq-echo-server.c.

atlk_rc_t imq_bind (imq_socket_t ∗ *socket,* imq_address_t *address*) Bind IMQ socket.

Parameters

in	*socket*	IMQ socket
in	*address*	IMQ address

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/imq/imq-echo-server.c.

atlk_rc_t imq_close (imq_socket_t ∗ *socket*) Close IMQ socket.

Parameters

in	*socket*	IMQ socket

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/imq/imq-client.c, and craton-threadx/imq/imq-echo-server.c.

atlk_rc_t imq_connect (imq_socket_t ∗ *socket*, imq_address_t *address*, const atlk_wait_t ∗ *wait*) Connect to IMQ server.

See Also

Using wait option.

Parameters

in	*socket*	IMQ socket
in	*address*	IMQ server address
in	*wait*	Wait specification (optional)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/imq/imq-client.c.

atlk_rc_t imq_fragmented_receive (imq_socket_t ∗ *socket*, const atlk_fragment_t ∗ *fragment_array_ptr*, size_t *fragment_array-_size*, size_t ∗ *data_size_ptr*, const atlk_wait_t ∗ *wait*) Receive IMQ message into a scatter-gather buffer.

See Also

Using wait option.

Parameters

in	*socket*	IMQ socket
in	*fragment_array_ptr*	Pointer to array of data fragments
in	*fragment_array_size*	Number of data fragments
out	*data_size_ptr*	Data size in bytes
in	*wait*	Wait specification (optional)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t imq_fragmented_send (imq_socket_t ∗ *socket*, const atlk_const_fragment_t ∗ *fragment_array_ptr*, size_t *fragment_array_size*, const atlk_wait_t ∗ *wait*) Send IMQ scatter-gather message.

See Also

Using wait option.

Parameters

in	*socket*	IMQ socket
in	*fragment_array_ptr*	Pointer to array of data fragments
in	*fragment_array_size*	Number of data fragments to send
in	*wait*	Wait specification (optional)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t imq_init (const imq_config_t ∗ *config*) Initialize IMQ on a specific CPU.
Parameters

in	*config*	IMQ configuration parameters (optional)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t imq_listen (imq_socket_t ∗ *socket*, const imq_service_config_t ∗ *config*) Listen for incoming IMQ connections.
Parameters

in	*socket*	IMQ socket
in	*config*	Service configuration

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/imq/imq-echo-server.c.

atlk_rc_t imq_receive (imq_socket_t ∗ *socket,* void ∗ *data_ptr,* size_t ∗ *data_size_ptr,* const atlk_wait_t ∗ *wait*) Receive IMQ message.

See Also

Using wait option.

Parameters

in	*socket*	IMQ socket
in	*data_ptr*	Pointer to start of data
in,out	*data_size_ptr*	Maximum (in) and actual (out) data size in bytes
in	*wait*	Wait specification (optional)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/imq/imq-client.c, and craton-threadx/imq/imq-echo-server.c.

atlk_rc_t imq_send (imq_socket_t ∗ *socket,* const void ∗ *data_ptr,* size_t *data_size,* const atlk_wait_t ∗ *wait*) Send IMQ message.

See Also

Using wait option.

Parameters

in	*socket*	IMQ socket
in	*data_ptr*	Pointer to start of data
in	*data_size*	Size of data in bytes
in	*wait*	Wait specification (optional)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/imq/imq-client.c, and craton-threadx/imq/imq-echo-server.c.

7.76 craton/imq_user.h File Reference

IMQ user API.

```
#include <craton/check.h>
```

Macros

- #define IMQ_USER_ADDRESS_MIN 0

  IMQ user address minimum value.
- #define IMQ_USER_ADDRESS_MAX 11

  IMQ user address maximum value.

7.76.1 Detailed Description

IMQ user API.

7.77 craton/io.h File Reference

Memory-mapped I/O API.

```
   #include <atlk/sdk.h>
#include <craton/check.h>
```

Functions

- uint8_t ioread8 (const void ∗ptr)

  Read a memory-mapped byte.
- uint16_t ioread16 (const void ∗ptr)

  Read a memory-mapped word.
- uint32_t ioread32 (const void ∗ptr)

  Read a memory-mapped double word.
- void iowrite8 (uint8_t value, void ∗ptr)

  Write a memory-mapped byte.
- void iowrite16 (uint16_t value, void ∗ptr)

  Write a memory-mapped word.
- void iowrite32 (uint32_t value, void ∗ptr)

  Write a memory-mapped double word.

7.77.1 Detailed Description

Memory-mapped I/O API.

7.77.2 Function Documentation

uint16_t ioread16 (const void ∗ *ptr*) `[inline]` Read a memory-mapped word.

Parameters

in	*ptr*	Pointer to read from

Returns

Read word

uint32_t ioread32 (const void ∗ *ptr*) `[inline]` Read a memory-mapped double word.

Parameters

in	*ptr*	Pointer to read from

Returns

Read double word

uint8_t ioread8 (const void * *ptr*) `[inline]` Read a memory-mapped byte.

Parameters

in	*ptr*	Pointer to read from

Returns

Read byte

void iowrite16 (uint16_t *value*, void * *ptr*) `[inline]` Write a memory-mapped word.

Parameters

in	*value*	Value to write
in	*ptr*	Pointer to write to

void iowrite32 (uint32_t *value*, void * *ptr*) `[inline]` Write a memory-mapped double word.

Parameters

in	*value*	Value to write
in	*ptr*	Pointer to write to

void iowrite8 (uint8_t *value*, void * *ptr*) `[inline]` Write a memory-mapped byte.

Parameters

in	*value*	Value to write
in	*ptr*	Pointer to write to

7.78 craton/iomux.h File Reference

CRATON IOMUX control API.

```
   #include <atlk/sdk.h>
#include <craton/check.h>
```

Enumerations

- enum iomux_entry_t

  IOMUX configuration table entry.

Functions

- atlk_rc_t iomux_write (iomux_entry_t entry, uint32_t value)

  Write one entry of CRATON's IOMUX configuration table.

7.78.1 Detailed Description

CRATON IOMUX control API. Reference: "Craton top level register file", revision 10.

7.78.2 Function Documentation

atlk_rc_t iomux_write (iomux_entry_t *entry*, uint32_t *value*) Write one entry of CRATON's IOMUX configuration table.

Remarks

For example, to set IOMUX_SOR_3 to 0b1 call iomux_write(IOMUX_SOR_3, 1).

Parameters

in	*entry*	IOMUX configuration table entry
in	*value*	Value to be set for that entry

Return values

ATLK_OK	if succeeded
ATLK_E_INVALID_ARG	Invalid entry or value

7.79 craton/memc.h File Reference

MEMC API.

```
   #include <atlk/sdk.h>
#include <craton/io.h>
#include <craton/check.h>
```

Data Structures

- struct memc_chip_select_info_t

  MEMC chip-select information.

Enumerations

- enum memc_chip_select_t {
  MEMC_CHIP_SELECT_0 = 0, MEMC_CHIP_SELECT_1 = 1, MEMC_CHIP_SELECT_2 = 2, MEMC_CHIP_SELECT-_3 = 3,
  MEMC_CHIP_SELECT_4 = 4, MEMC_CHIP_SELECT_5 = 5, MEMC_CHIP_SELECT_6 = 6, MEMC_CHIP_SELECT-_7 = 7,
  MEMC_CHIP_SELECT_MAX = MEMC_CHIP_SELECT_7 }

  MEMC chip select number.

Functions

- atlk_rc_t memc_chip_select_info_get (memc_chip_select_t chip_select, memc_chip_select_info_t ∗chip_select_info_ptr)

  Get chip select memory region information.

7.79.1 Detailed Description

MEMC API.

7.79.2 Enumeration Type Documentation

enum memc_chip_select_t MEMC chip select number.

Enumerator

MEMC_CHIP_SELECT_0	CS number 0.
MEMC_CHIP_SELECT_1	CS number 1.
MEMC_CHIP_SELECT_2	CS number 2.
MEMC_CHIP_SELECT_3	CS number 3.
MEMC_CHIP_SELECT_4	CS number 4.

MEMC_CHIP_SELECT_5	CS number 5.
MEMC_CHIP_SELECT_6	CS number 6.
MEMC_CHIP_SELECT_7	CS number 7.
MEMC_CHIP_SELECT_MAX	CS maximum value.

7.79.3 Function Documentation

atlk_rc_t memc_chip_select_info_get (memc_chip_select_t *chip_select*, memc_chip_select_info_t ∗ *chip_select_info_ptr*)
Get chip select memory region information.
Parameters

in	*chip_select*	Chip select number
out	*chip_select_info_-ptr*	Pointer to chip select information

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.80 craton/net.h File Reference

CRATON TCP/IP network stack API.

```
#include <nx_api.h>
#include <atlk/sdk.h>
#include <craton/check.h>
```

Functions

- atlk_rc_t net_ip_trusted_instance_get (NX_IP ∗∗ip_ptr)

  Get pointer to NetX-Duo trusted IP instance.
- atlk_rc_t net_ip_untrusted_instance_get (NX_IP ∗∗ip_ptr)

  Get pointer to NetX-Duo untrusted IP instance.

7.80.1 Detailed Description

CRATON TCP/IP network stack API.

7.80.2 Function Documentation

atlk_rc_t net_ip_trusted_instance_get (NX_IP ∗∗ *ip_ptr*) Get pointer to NetX-Duo trusted IP instance.
Parameters

out	*ip_ptr*	Pointer to IP instance

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/bridge/v2x-udp-bridge-example.c, craton-threadx/net/nx-raw-packet-receive-example.c, and craton-threadx/net/udp-receive-example.c.

atlk_rc_t net_ip_untrusted_instance_get (NX_IP ∗∗ *ip_ptr*) Get pointer to NetX-Duo untrusted IP instance.

Parameters

out	*ip_ptr*	Pointer to IP instance

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/wave-ipv6/wave-ipv6-client-example.c, craton-threadx/wave-ipv6/wave-ipv6-example.c, and craton-threadx/wave-ipv6/wave-ipv6-server-example.c.

7.81 craton/nor_flash.h File Reference

NOR Flash API.

```
   #include <atlk/sdk.h>
#include <craton/check.h>
```

Data Structures

- struct norfl_part_info_t

  Partition table entry.
- struct norfl_part_table_t

  Partition table.

Macros

- #define NORFL_PART_FIRMWARE_MAIN 0

  Main firmware image partition number.
- #define NORFL_PART_FIRMWARE_RESCUE 1

  Rescue firmware image partition number.
- #define NORFL_NUM_PARTS 16

  Total number of partitions.
- #define NORFL_PART_TYPE_UNUSED 0

  Partition type "unused".
- #define NORFL_PART_TYPE_FIRMWARE 1

  Partition type "firmware".
- #define NORFL_PART_TYPE_SAFEFAT 2

  Partition type "SafeFAT file system".
- #define NORFL_PART_TYPE_USER_MIN 0x80

  Smallest user-defined partition type.
- #define NORFL_PART_TYPE_USER_MAX 0xff

  Largest user-defined partition type.
- #define NORFL_PART_TYPE_F_LOCKABLE 0x100

  Partition can be locked against modification.
- #define NORFL_PART_TYPE_F_LOCKED 0x200

  Partition is locked against modification.
- #define NORFL_PART_INFO_INIT

  Partition table entry default initializer.
- #define NORFL_PART_TABLE_INIT

  Partition table default initializer.

Typedefs

- typedef uint8_t norfl_part_num_t

  Partition number.

Enumerations

- enum norfl_next_boot_t { NORFL_NEXT_BOOT_MAIN = 0, NORFL_NEXT_BOOT_RESCUE = 1, NORFL_NEXT_BOOT_FLASHER = 2 }

  Source of next warm boot.

Functions

- atlk_rc_t norfl_init (void)

  Initialize NOR Flash Interface.
- atlk_rc_t norfl_part_table_read (norfl_part_table_t ∗table)

  Read partition table from flash.
- atlk_rc_t norfl_part_read (norfl_part_num_t part_num, uint32_t offset, void ∗data_ptr, size_t data_size)

  Read bytes from partition.
- atlk_rc_t norfl_part_rewrite (norfl_part_num_t part_num, const void ∗data_ptr, size_t data_size)

  Erase and program entire partition.
- atlk_rc_t norfl_part_lock (norfl_part_num_t part_num)

  Lock partition against further modification.
- atlk_rc_t norfl_next_boot_set (norfl_next_boot_t next_boot)

  Set source of next warm boot.
- atlk_rc_t norfl_next_boot_get (norfl_next_boot_t ∗next_boot)

  Get source of next warm boot.

7.81.1 Detailed Description

NOR Flash API.

7.81.2 Enumeration Type Documentation

enum norfl_next_boot_t Source of next warm boot.

Enumerator

NORFL_NEXT_BOOT_MAIN	Try to boot from main firmware image.
NORFL_NEXT_BOOT_RESCUE	Try to boot from rescue firmware image.
NORFL_NEXT_BOOT_FLASHER	Boot into ROM-based flashing agent.

7.81.3 Function Documentation

atlk_rc_t norfl_init (void) Initialize NOR Flash Interface.

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t norfl_next_boot_get (norfl_next_boot_t ∗ *next_boot*) Get source of next warm boot.

Parameters

out	*next_boot*	Source of next boot

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t norfl_next_boot_set (norfl_next_boot_t *next_boot*) Set source of next warm boot.

Parameters

in	*next_boot*	Source of next boot

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t norfl_part_lock (norfl_part_num_t *part_num*) Lock partition against further modification.

Parameters

in	*part_num*	Partition number

Precondition

A partition must have the NORFL_PART_TYPE_F_LOCKABLE flag set in the norfl_part_info_t::part_type field.

Postcondition

A partition will have the NORFL_PART_TYPE_F_LOCKED flag set in the norfl_part_info_t::part_type field, and will have the NORFL_PART_TYPE_F_LOCKABLE flag unset.

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t norfl_part_read (norfl_part_num_t *part_num*, uint32_t *offset*, void ∗ *data_ptr*, size_t *data_size*) Read bytes from partition.

Parameters

in	*part_num*	Partition number
in	*offset*	Offset from partition start
out	*data_ptr*	Pointer to output buffer
in	*data_size*	Size of output buffer in bytes

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/otp/otp-example.c.

atlk_rc_t norfl_part_rewrite (norfl_part_num_t *part_num*, const void * *data_ptr*, size_t *data_size*) Erase and program entire partition.

Warning

Doesn't check validity of written data or that it has been written correctly.

Parameters

in	*part_num*	Partition number
in	*data_ptr*	Pointer to input buffer
in	*data_size*	Size of input buffer in bytes

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/firmware/fw-update-example.c, and craton-threadx/otp/otp-example.c.

atlk_rc_t norfl_part_table_read (norfl_part_table_t * *table*) Read partition table from flash.

Parameters

in	*table*	Partition table pointer

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/firmware/fw-update-example.c, and craton-threadx/otp/otp-example.c.

7.82 craton/reboot.h File Reference

Reboot API.

```
   #include <atlk/sdk.h>
#include <craton/check.h>
```

Enumerations

- enum system_reboot_t { SYSTEM_REBOOT_SOC = 0, SYSTEM_REBOOT_PCB = 1 }

  Reboot method specifier.

Functions

- atlk_rc_t system_reboot (system_reboot_t reboot_type)

  Reboot the system.

7.82.1 Detailed Description

Reboot API.

7.82.2 Enumeration Type Documentation

enum system_reboot_t Reboot method specifier.

Enumerator

SYSTEM_REBOOT_SOC	System-On-Chip reboot - reboot CPU only.
SYSTEM_REBOOT_PCB	PCB reboot - reboot entire system (not supported on all boards)

7.82.3 Function Documentation

atlk_rc_t system_reboot (system_reboot_t *reboot_type*) Reboot the system.

Parameters

in	*reboot_type*	Reboot method specifier

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.83 craton/rng_hw.h File Reference

CRATON RNG HW API.

```
#include <atlk/sdk.h>
#include <craton/check.h>
```

Functions

- atlk_rc_t rng_hw_init (void)

  Init RNG module.
- atlk_rc_t rng_hw_get (uint32_t *vector, size_t size)

  Get a vector of uniformly distributed random 32-bit words.

7.83.1 Detailed Description

CRATON RNG HW API.

7.83.2 Function Documentation

atlk_rc_t rng_hw_get (uint32_t * *vector,* size_t *size*) Get a vector of uniformly distributed random 32-bit words.

Parameters

in	*vector*	Buffer to store the random 32-bit words

in	*size*	Buffer size in 32-bit words

Return values

ATLK_OK	if succeeded
ATLK_E_INVALID_STATE	if RNG is not initialized
ATLK_E_INVALID_ARG	if mandatory function argument is not specified

Returns

Error code if failed

atlk_rc_t rng_hw_init (void) Init RNG module.

Warning

Should be called once only.

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.84 craton/sha_hw.h File Reference

CRATON SHA HW API.

```
   #include <atlk/sdk.h>
#include <atlk/sha.h>
#include <craton/check.h>
```

Functions

- atlk_rc_t sha_hw_init (void)

  Initialize SHA hardware driver.
- atlk_rc_t sha_hw_sha224_compute (const void ∗data_ptr, size_t data_size, sha_digest_t ∗digest)

  Compute SHA-224 using dedicated hardware.
- atlk_rc_t sha_hw_sha256_compute (const void ∗data_ptr, size_t data_size, sha_digest_t ∗digest)

  Compute SHA-256 using dedicated hardware.

7.84.1 Detailed Description

CRATON SHA HW API.

7.84.2 Function Documentation

atlk_rc_t sha_hw_init (void) Initialize SHA hardware driver.

Warning

Should be called once only.

Return values

ATLK_OK	if succeeded
ATLK_E_INVALID_STATE	if SHA is already initialized

atlk_rc_t sha_hw_sha224_compute (const void * *data_ptr,* size_t *data_size,* sha_digest_t * *digest*) Compute SHA-224 using dedicated hardware.

Only the first sha_digest_t::value_size octets of sha_digest_t::value are the calculated hash value.

Parameters

in	*data_ptr*	Data over which the hash will be computed
in	*data_size*	Data length in octets
out	*digest*	Calculated SHA-224 digest

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t sha_hw_sha256_compute (const void * *data_ptr,* size_t *data_size,* sha_digest_t * *digest*) Compute SHA-256 using dedicated hardware.

Only the first sha_digest_t::value_size octets of sha_digest_t::value are the calculated hash value.

Parameters

in	*data_ptr*	Data over which the hash will be computed
in	*data_size*	Data length in octets
out	*digest*	Calculated SHA-256 digest

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/crypto/ecdsa-example.c.

7.85 craton/slx97_host.h File Reference

CRATON SLx97 host API.

```
#include <atlk/sdk.h>
#include <atlk/slx97.h>
#include <craton/spi_driver.h>
#include <craton/gpio.h>
#include <craton/check.h>
```

Data Structures

- struct slx97_host_io_config_t

  SLx97 host I/O configuration parameters.

Macros

- #define SLX97_HOST_IO_CONFIG_INIT

  SLx97 host I/O configuration default initializer.

Functions

- atlk_rc_t slx97_host_init (const slx97_host_io_config_t *io_config, const slx97_host_sec_config_t *sec_config)

    Create SLx97 host.
- atlk_rc_t slx97_host_sec_init (const slx97_host_sec_key_t *master_key)

    Initialize SLx97 communication security.
- atlk_rc_t slx97_server_init (const slx97_host_io_config_t *io_config)

    Initialize SLx97 server.

7.85.1 Detailed Description

CRATON SLx97 host API.

7.85.2 Function Documentation

atlk_rc_t slx97_host_init (const slx97_host_io_config_t * *io_config*, const slx97_host_sec_config_t * *sec_config*) Create SLx97 host.

Warning

If this function failed, calling it again may result in undefined behavior.

Parameters

in	*io_config*	SLx97 host I/O configuration parameters
in	*sec_config*	SLx97 host communication security parameters

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t slx97_host_sec_init (const slx97_host_sec_key_t * *master_key*) Initialize SLx97 communication security.

`master_key` is required if and only if slx97_host_sec_config_t::sec_master_key_external was nonzero when provided as argument to slx97_host_init.

Parameters

in	*master_key*	Master key (optional)

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t slx97_server_init (const slx97_host_io_config_t * *io_config*) Initialize SLx97 server.

Warning

If this function failed, calling it again may result in undefined behavior.

Parameters

in	*io_config*	SLx97 host I/O configuration parameters

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.86 craton/spi_driver.h File Reference

CRATON SPI driver API.

```
  #include <atlk/sdk.h>
#include <craton/hdmac.h>
#include <craton/check.h>
```

Data Structures

- struct spi_config_t

  SPI device configuration.
- struct spi_dma_transfer_t

  SPI DMA transfer descriptor.

Macros

- #define SPI_DEVICE_ID_MIN 0

  First SPI device ID.
- #define SPI_DEVICE_ID_MAX 4

  Last SPI device ID.
- #define SPI_DEVICE_ID_NA 0xff

  Invalid SPI device ID.
- #define SPI_DATA_BITS_MIN 4

  Minimum SPI data size in bits.
- #define SPI_DATA_BITS_MAX 16

  Maximum SPI data size in bits.
- #define SPI_CONFIG_INIT

  SPI device configuration default initializer.
- #define SPI_DMA_TRANSFER_INIT

  SPI DMA transfer descriptor default initializer.

Typedefs

- typedef uint8_t spi_device_id_t

  SPI device ID.
- typedef struct spi_device spi_device_t

  SPI device object.

Enumerations

- enum spi_mode_t { SPI_MODE_MASTER = 0, SPI_MODE_SLAVE = 1, SPI_MODE_NA = 0xff }

  SPI device mode.
- enum spi_clock_polarity_t { SPI_CLOCK_POLARITY_IDLE_LOW = 0, SPI_CLOCK_POLARITY_IDLE_HIGH = 1 }

  SPI clock polarity.
- enum spi_clock_phase_t { SPI_CLOCK_PHASE_1ST_EDGE = 0, SPI_CLOCK_PHASE_2ND_EDGE = 1 }

  SPI clock phase.

Functions

- atlk_rc_t spi_sspclk_get (spi_device_id_t device_id, uint32_t *sspclk)
  Get base clock rate (in Hz) of SPI device.
- atlk_rc_t spi_driver_init (void)
  Initialize SPI driver.
- atlk_rc_t spi_device_init (const spi_config_t *config, spi_device_t **device_ptr)
  Initialize SPI device according to user configuration.
- atlk_rc_t spi_dma_transfer_start (spi_dma_transfer_t *transfer)
  Start Tx and/or Rx of data on SPI using DMA.

7.86.1 Detailed Description

CRATON SPI driver API. References:

1. ARM PrimeCell(r) Synchronous Serial Port (PL022); revision r1p3.
2. ATK4100A1 – ATK4100A0 (CRATON) Datasheet; version 1.9.

7.86.2 Enumeration Type Documentation

enum spi_clock_phase_t SPI clock phase.

Enumerator

SPI_CLOCK_PHASE_1ST_EDGE	Data signal is sampled at clock first edge.
SPI_CLOCK_PHASE_2ND_EDGE	Data signal is sampled at clock second edge.

enum spi_clock_polarity_t SPI clock polarity.

Enumerator

SPI_CLOCK_POLARITY_IDLE_LOW	SPI clock signal is idle when low.
SPI_CLOCK_POLARITY_IDLE_HIGH	SPI clock signal is idle when high.

enum spi_mode_t SPI device mode.

Enumerator

SPI_MODE_MASTER	SPI device is master.
SPI_MODE_SLAVE	SPI device is slave.
SPI_MODE_NA	Invalid SPI device mode.

7.86.3 Function Documentation

atlk_rc_t spi_device_init (const spi_config_t * *config*, spi_device_t ** *device_ptr*) Initialize SPI device according to user configuration.

Parameters

in	*config*	SPI device configuration
out	*device_ptr*	SPI device object pointer

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/spi/spi-master-example.c, and craton-threadx/spi/spi-slave-example.c.

atlk_rc_t spi_dma_transfer_start (spi_dma_transfer_t * *transfer*) Start Tx and/or Rx of data on SPI using DMA.

Parameters

in	*transfer*	SPI DMA transfer descriptor

Remarks

At least one of spi_dma_transfer_t::tx_buffer_ptr and spi_dma_transfer_t::rx_buffer_ptr must be non-NULL.

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/spi/spi-master-example.c, and craton-threadx/spi/spi-slave-example.c.

atlk_rc_t spi_driver_init (void) Initialize SPI driver.

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t spi_sspclk_get (spi_device_id_t *device_id*, uint32_t ∗ *sspclk*) Get base clock rate (in Hz) of SPI device.

Parameters

in	*device_id*	SPI device ID
out	*sspclk*	SPI SSP clock in Hz

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.87 craton/syslog.h File Reference

CRATON system logger API.

```
   #include <atlk/sdk.h>
#include <craton/check.h>
```

Macros

- #define SYSLOG_SINK_F_CONSOLE (1 << 1)

  Syslog sink console flag.
- #define SYSLOG_SINK_F_TCP (1 << 2)

  Syslog sink TCP flag.
- #define SYSLOG_SINK_F_UDP (1 << 3)

  Syslog sink UDP flag.

Enumerations

- enum syslog_level_t {
  LOG_ERR = 3, LOG_WARNING = 4, LOG_NOTICE = 5, LOG_INFO = 6,
  LOG_DEBUG = 7 }

  Log trace level.

Functions

- void syslog (syslog_level_t level, const char ∗format,...) atlk_format_printf(2

  Generate a log message.
- void atlk_rc_t syslog_level_set (syslog_level_t level)

  Set Syslog trace level value.
- atlk_rc_t syslog_level_get (syslog_level_t ∗level)

  Get Syslog trace level value.
- atlk_rc_t syslog_sink_set (uint32_t sink_mask)

  Set Syslog trace sink value.
- atlk_rc_t syslog_sink_get (uint32_t ∗sink_mask)

  Get Syslog trace sink value.

7.87.1 Detailed Description

CRATON system logger API.

7.87.2 Enumeration Type Documentation

enum syslog_level_t Log trace level.

Enumerator

LOG_ERR	Error conditions.
LOG_WARNING	Warning conditions.
LOG_NOTICE	Normal but significant condition.
LOG_INFO	Informational messages.
LOG_DEBUG	Debug-level messages.

7.87.3 Function Documentation

void syslog (syslog_level_t *level*, const char ∗ *format*, ...) Generate a log message.

Parameters

in	*level*	Message level
in	*format*	Message format

Examples:

craton-threadx/gnss-teseo/poti-hil.c, craton-threadx/nav/nav-trace.h, and craton-threadx/wlan-driver/traffic-monitor-example.c.

atlk_rc_t syslog_level_get (syslog_level_t ∗ *level*) Get Syslog trace level value.

Parameters

out	*level*	Syslog trace level value.

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

void atlk_rc_t syslog_level_set (syslog_level_t *level*) Set Syslog trace level value.

Parameters

`in`	*level*	Syslog trace level value.

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t syslog_sink_get (uint32_t ∗ *sink_mask*) Get Syslog trace sink value.

Parameters

`out`	*sink_mask*	Syslog sink mask value.

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t syslog_sink_set (uint32_t *sink_mask*) Set Syslog trace sink value.

Parameters

`in`	*sink_mask*	Syslog sink mask value.

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.88 craton/uart_driver.h File Reference

UART driver API.

```
   #include <atlk/sdk.h>
#include <atlk/uart.h>
#include <craton/check.h>
```

Macros

- #define UART_MODE_F_ONLCR (1U << 0)

  UART mode flag: Map NL to CR-NL on output.

Enumerations

- enum uart_device_id_t { UART_DEVICE_ID_0 = 0, UART_DEVICE_ID_1 = 1, UART_DEVICE_ID_2 = 2 }
  UART device ID.

Functions

- atlk_rc_t uart_speed_set (uart_device_id_t device_id, uart_speed_bps_t speed_bps)
  Set UART device speed in bits/s.
- atlk_rc_t uart_speed_get (uart_device_id_t device_id, uart_speed_bps_t *speed_bps)
  Get UART device speed in bits/s.
- atlk_rc_t uart_rx_flush (uart_device_id_t device_id)
  Flush UART device receive buffer.
- atlk_rc_t uart_mode_set (uart_device_id_t device_id, unsigned int flags)
  Set UART mode.
- atlk_rc_t uart_mode_get (uart_device_id_t device_id, unsigned int *flags_ptr)
  Get UART mode.

7.88.1 Detailed Description

UART driver API.

7.88.2 Enumeration Type Documentation

enum uart_device_id_t UART device ID.

Enumerator

UART_DEVICE_ID_0	CRATON UART0 device ID.
UART_DEVICE_ID_1	CRATON UART1 device ID.
UART_DEVICE_ID_2	UART over SPI device ID.

7.88.3 Function Documentation

atlk_rc_t uart_mode_get (uart_device_id_t *device_id*, unsigned int * *flags_ptr*) Get UART mode.

Parameters

in	*device_id*	UART device ID
out	*flags_ptr*	UART device flags

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t uart_mode_set (uart_device_id_t *device_id*, unsigned int *flags*) Set UART mode.

Parameters

in	*device_id*	UART device ID
in	*flags*	UART device flags

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t uart_rx_flush (uart_device_id_t *device_id*) Flush UART device receive buffer.
Upon successful completion, any data received but not read by the device is discarded.

Parameters

`in`	*device_id*	UART device ID

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t uart_speed_get (uart_device_id_t *device_id*, uart_speed_bps_t * *speed_bps*) Get UART device speed in bits/s.

Parameters

`in`	*device_id*	UART device ID
`out`	*speed_bps*	Speed in bits/s

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t uart_speed_set (uart_device_id_t *device_id*, uart_speed_bps_t *speed_bps*) Set UART device speed in bits/s.

Remarks

UART0/1 device speed change settling time is in the order of 10ms.

Parameters

`in`	*device_id*	UART device ID
`in`	*speed_bps*	Speed in bits/s

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/bt-spi2uart/bt-spi2uart-example.c, and craton-threadx/gnss/gnss-integration-example.c.

7.89 craton/user.h File Reference

CRATON user API.

```
#include <atlk/sdk.h>
#include <craton/exception.h>
#include <craton/check.h>
#include <craton/v2x_config.h>
#include <craton/slx97_host.h>
```

Functions

- void craton_user_abort_handler (const exception_info_t *info)

  CRATON user abort handler.
- void craton_user_init (void)

  Craton user code initialization function.

Variables

- const v2x_config_t craton_user_v2x_config

  CRATON user V2X configuration.
- const slx97_host_sec_config_t craton_user_slx97_host_sec_config

  SLx97 communication security configuration.

7.89.1 Detailed Description

CRATON user API. All functions declared in this file are for the user to define.

7.89.2 Function Documentation

void craton_user_abort_handler (const exception_info_t * *info*) CRATON user abort handler.

Warning

New implementation of this handler will override default exception handling (defined as a weak symbol).

Parameters

in	*info*	Exception information

Examples:

craton-threadx/diagnostics/craton-user-abort-example.c.

void craton_user_init (void) Craton user code initialization function.

This function is the user application entry point and is defined per CRATON CPU.

Warning

Default implementation is defined as a weak symbol, but requires linking with libvca.

Examples:

craton-threadx/bridge/v2x-udp-bridge-example.c, craton-threadx/bt-spi2uart/bt-spi2uart-example.c, craton-threadx/build/main.-c, craton-threadx/can/can-example.c, craton-threadx/can/can-hw-filter-example.c, craton-threadx/cli/cli-example.-c, craton-threadx/crypto/aes-example.c, craton-threadx/crypto/ecdsa-benchmark.c, craton-threadx/crypto/ecdsa-example.c, craton-threadx/crypto/ecies-example.c, craton-threadx/crypto/secure-storage-example.c, craton-threadx/diagnostics/c user-abort-example.c, craton-threadx/dot4/dot4-channel-switching-example.c, craton-threadx/firmware/fw-update-example.c, craton-threadx/fs/fs-example.c, craton-threadx/gnss-teseo/gnss-teseo-fw-update-example.c, craton-threadx/gnss-teseo/gnss-teseo-sou-example.c, craton-threadx/gnss/gnss-integration-example.c, craton-threadx/i2s/i2s-example.c, craton-threadx/imq/imq-client.c, craton-threadx/imq/imq-echo-server.c, craton-threadx/mibs/mibs-edca-example.c, craton-threadx/mibs/mibs-example.c, craton-threadx/nav/nav-data-example.c, craton-threadx/nav/nav-example.c, craton-threadx/nav/system-time-benchmark.c, craton-threadx/net/http-example.c, craton-threadx/net/nx-bsd-udp-receive-example.c, craton-threadx/net/nx-raw-packet-receive-example.c, craton-threadx/net/udp-receive-example.c, craton-threadx/otp/otp-example.c, craton-threadx/posix/posix-example.c, craton-threadx/sntp/sntp-example.c, craton-threadx/spi/spi-master-example.c, craton-threadx/spi/spi-slave-example.c, craton-threadx/sys-alarm/sys-alarm-example.c, craton-threadx/v2x-emulator/v2x-emulator-over-udp-example.c, craton-threadx/v2x-emulator/v2x-service-user.c, craton-threadx/v2x/v2x-example.c, craton-threadx/wave-ipv6/wave-ipv6-client-example.-c, craton-threadx/wave-ipv6/wave-ipv6-example.c, craton-threadx/wave-ipv6/wave-ipv6-server-example.c, and craton-threadx/wlan-driver/traffic-monitor-example.c.

7.89.3 Variable Documentation

const slx97_host_sec_config_t craton_user_slx97_host_sec_config SLx97 communication security configuration.

Remarks

Default definition is defined as a weak symbol

const v2x_config_t craton_user_v2x_config CRATON user V2X configuration.
This structure defines V2X memory allocation sizes.

Remarks

Default definition is defined as a weak symbol.

7.90 craton/v2x_emulator_init.h File Reference

CRATON V2X emulator initialization API.

```
#include <atlk/sdk.h>
#include <atlk/os.h>
#include <atlk/v2x_emulator.h>
#include <craton/imq.h>
#include <craton/check.h>
```

Data Structures

- struct v2x_emulator_config_t

  V2X emulator configuration descriptor.

Macros

- #define V2X_EMULATOR_CONFIG_INIT

  V2X emulator internal configuration descriptor default initializer.

Functions

- atlk_rc_t v2x_emulator_master_init (v2x_emulator_t **emulator_ptr, const v2x_emulator_config_t *config)

  Initialize V2X emulator master.
- atlk_rc_t v2x_emulator_slave_init (const v2x_emulator_config_t *config)

  Initialize V2X emulator slave.

7.90.1 Detailed Description

CRATON V2X emulator initialization API.

7.90.2 Function Documentation

atlk_rc_t v2x_emulator_master_init (v2x_emulator_t ** *emulator_ptr*, const v2x_emulator_config_t * *config*) Initialize V2X emulator master.
Should be called in the CPU in which the V2X emulator is used.

Remarks

1. Init should be called before any other emulator function call
2. Function should be called only once

Parameters

in,out	*emulator_ptr*	V2X emulator
in	*config*	Emulator configuration

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/v2x-emulator/v2x-emulator-over-udp-example.c.

atlk_rc_t v2x_emulator_slave_init (const v2x_emulator_config_t ∗ *config*) Initialize V2X emulator slave.
Should be called in the CPU in which the V2X service is used.

Remarks

1. Init should be called before any other emulator function call
2. Function should be called only once

Parameters

in	*config*	Emulator configuration

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/v2x-emulator/v2x-emulator-over-udp-example.c, and craton-threadx/v2x-emulator/v2x-service-user.c.

7.91 craton/wave_ipv6.h File Reference

WAVE IPv6 API.

```
   #include <atlk/sdk.h>
#include <craton/check.h>
```

Functions

- atlk_rc_t wave_ipv6_enabled_set (int32_t if_index, int enabled)

  Set WAVE IPv6 enabled.
- atlk_rc_t wave_ipv6_enabled_get (int32_t if_index, int ∗enabled)

  Get WAVE IPv6 enabled.

7.91.1 Detailed Description

WAVE IPv6 API.

7.91.2 Function Documentation

atlk_rc_t wave_ipv6_enabled_get (int32_t *if_index*, int ∗ *enabled*) Get WAVE IPv6 enabled.

Parameters

in	*if_index*	MAC interface index
out	*enabled*	Whether WAVE IPv6 is enabled

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t wave_ipv6_enabled_set (int32_t *if_index*, int *enabled*) Set WAVE IPv6 enabled.

Parameters

in	*if_index*	MAC interface index
in	*enabled*	Whether WAVE IPv6 is enabled

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.92 craton/wd.h File Reference

Watchdog (WD) API.

```
#include <atlk/sdk.h>
#include <craton/exception_arm.h>
#include <craton/exception_arc.h>
#include <craton/check.h>
```

Data Structures

- struct wd_arc_config_t

  WD configuration for ARC.
- struct wd_config_t

  WD configuration for ARM.

Macros

- #define WD_ARC_CORES_COUNT 2

  Number of ARC cores.

Typedefs

- typedef void(∗ wd_arc_exception_handler_t)(uint8_t arc_num, const exception_arc_info_t ∗info)

  WD expiration callback for ARC.
- typedef void(∗ wd_arm_exception_handler_t)(const exception_arm_info_t ∗info)

  WD expiration callback for ARM.

Enumerations

- enum wd_mode_t { WD_MODE_RESTART = 0, WD_MODE_CB = 1 }

  WD expiration mode.

Functions

- atlk_rc_t wd_init (void)

  Initialize WD on ARC.
- atlk_rc_t wd_enabled_set (int enabled)

  Set whether WD is enabled.
- atlk_rc_t wd_enabled_get (int ∗enabled)

  Get whether WD is enabled.
- void wd_chip_reset (void) atlk_no_return

  Trigger system restart via WD.

7.92.1 Detailed Description

Watchdog (WD) API.

7.92.2 Typedef Documentation

typedef void(∗ wd_arc_exception_handler_t)(uint8_t arc_num, const exception_arc_info_t ∗info) WD expiration callback for A-RC.

Parameters

in	*arc_num*	ARC number, one of {1,2}
out	*info*	Exception info

typedef void(∗ wd_arm_exception_handler_t)(const exception_arm_info_t ∗info) WD expiration callback for ARM.

Warning

Callback is called from ISR context.

Parameters

out	*info*	Exception info

7.92.3 Enumeration Type Documentation

enum wd_mode_t WD expiration mode.

Enumerator

WD_MODE_RESTART	Restart system upon WD expiration.
WD_MODE_CB	Invoke user callback upon WD expiration.

7.92.4 Function Documentation

atlk_rc_t wd_enabled_get (int ∗ *enabled*) Get whether WD is enabled.

Parameters

out	*enabled*	WD is enabled

Return values

ATLK_OK	if succeeded

Returns

Error if failed

atlk_rc_t wd_enabled_set (int *enabled*) Set whether WD is enabled.

Parameters

in	*enabled*	WD is enabled

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

atlk_rc_t wd_init (void) Initialize WD on ARC.

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

7.93 craton/wlan_driver.h File Reference

WLAN Driver API.

```
#include <atlk/sdk.h>
#include <atlk/v2x.h>
#include <craton/check.h>
```

Data Structures

- struct wlan_frame_t

  WLAN frame.
- struct wlan_rx_frame_info_t

  WLAN RX frame info.
- struct wlan_tx_frame_info_t

  WLAN TX frame info.

Macros

- #define WLAN_FRAME_INIT

  WLAN frame default initializer.
- #define WLAN_RX_FRAME_INFO_INIT

  WLAN RX frame info default initializer.
- #define WLAN_TX_FRAME_INFO_INIT

  WLAN TX frame info default initializer.

Typedefs

- typedef void(∗ wlan_rx_traffic_monitor_t)(const wlan_frame_t ∗frame, const wlan_rx_frame_info_t ∗info)

  Traffic monitor RX callback function.
- typedef void(∗ wlan_tx_traffic_monitor_t)(const wlan_frame_t ∗frame, const wlan_tx_frame_info_t ∗info)

  Traffic monitor TX callback function.

Functions

- atlk_rc_t wlan_rx_traffic_monitor_set (uint8_t device_id, wlan_rx_traffic_monitor_t rx_callback)

  Set RX callback.
- atlk_rc_t wlan_tx_traffic_monitor_set (uint8_t device_id, wlan_tx_traffic_monitor_t tx_callback)

  Set TX callback.

7.93.1 Detailed Description

WLAN Driver API.

7.93.2 Function Documentation

atlk_rc_t wlan_rx_traffic_monitor_set (uint8_t *device_id*, wlan_rx_traffic_monitor_t *rx_callback*) Set RX callback.

RX callback is called for each MPDU received (even when there are no open V2X sockets).

Remarks

Callbacks for different interfaces might be called simultaneously. Hence, callbacks should be designed to be reentrant. When a callback is set to NULL, it clears previously set callback.

Parameters

in	*device_id*	WLAN device ID
in	*rx_callback*	RX callback function

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/wlan-driver/traffic-monitor-example.c.

atlk_rc_t wlan_tx_traffic_monitor_set (uint8_t *device_id*, wlan_tx_traffic_monitor_t *tx_callback*) Set TX callback.

TX callback is called for each MPDU transmitted.

Remarks

Callbacks for different interfaces might be called simultaneously. Hence, callbacks should be designed to be reentrant. When a callback is set to NULL, it clears previously set callback.

Parameters

in	*device_id*	WLAN device ID
in	*tx_callback*	TX callback function

Return values

ATLK_OK	if succeeded

Returns

Error code if failed

Examples:

craton-threadx/wlan-driver/traffic-monitor-example.c.

7.94 tx_posix.h File Reference

POSIX API.

```
  #include <stdarg.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <fcntl.h>
#include <sched.h>
#include <time.h>
#include <signal.h>
#include <tx_api.h>
#include <sys/time.h>
```

Data Structures

- struct signal_info
- struct pthread_attr_t
- struct POSIX_TCB
- struct pthread_mutexattr_t
- struct pthread_mutex_t
- struct mq_attr
- struct POSIX_MSG_QUEUE
- struct mqd_t
- struct sem_t
- struct pthread_cond_t
- struct pthread_condattr_t
- struct pthread_once_t

Enumerations

- enum

  Error Codes for posix_error_handler.

Functions

- INT nanosleep (const struct timespec ∗req, struct timespec ∗rem)

  See below for function limitations.
- INT mq_send (mqd_t mqdes, const char ∗msg_ptr, size_t msg_len, ULONG msg_prio)

  See below for function limitations.
- ssize_t mq_receive (mqd_t mqdes, VOID ∗pMsg, size_t msgLen, ULONG ∗pMsgPrio)

  See below for function limitations.
- mqd_t mq_open (const CHAR ∗mqName, ULONG oflags,...)

  See below for function limitations.
- INT sem_close (sem_t ∗sem)

  See below for function limitations.
- INT sem_getvalue (sem_t ∗sem, ULONG ∗sval)

  See below for function limitations.
- INT sem_post (sem_t ∗sem)

  See below for function limitations.
- INT sem_trywait (sem_t ∗sem)

  See below for function limitations.
- INT sem_unlink (const char ∗name)

  See below for function limitations.
- INT sem_wait (sem_t ∗sem)

  See below for function limitations.
- INT pthread_detach (pthread_t thread)

  See below for function limitations.
- INT pthread_attr_setdetachstate (pthread_attr_t ∗attr, INT detachstate)

  See below for function limitations.
- INT pthread_attr_setinheritsched (pthread_attr_t ∗attr, INT inheritsched)

  See below for function limitations.
- INT pthread_mutexattr_settype (pthread_mutexattr_t ∗attr, INT type)

  See below for function limitations.
- INT pthread_mutexattr_setprotocol (pthread_mutexattr_t ∗attr, INT protocol)

See below for function limitations.
- INT pthread_cancel (pthread_t thread)

  See below for function limitations.
- INT pthread_once (pthread_once_t ∗once_control, VOID(∗init_routine)(VOID))

  See below for function limitations.
- INT pthread_cond_init (pthread_cond_t ∗cond, pthread_condattr_t ∗attr)

  See below for function limitations.
- INT sched_get_priority_max (INT policy)

  Scheduler functions.

7.94.1 Detailed Description

POSIX API. Overview

The Autotalks POSIX Suite supports many of the basic POSIX calls, with some limitations, and utilizes ThreadX® primitives underneath.

Each POSIX call is documented, including information about supported/unsupported options, limitations, deviations, and suggestions on how to work-around any limitations.

1. Usage

The file tx_posix.h must be included in the application source where POSIX calls are required, replacing the file pthread.h if such was included. Since the POSIX compliancy wrapper does not cover the complete standard, not all prototypes are provided.

1. Supported features

Autotalks POSIX suite supports the following POSIX features. See limitation section for unsupported features.

- Pthread
- Pthread Mutex
- Message queue
- Semaphore
- Condition Variables
- Timer - limited support

Limitations

All POSIX calls, excluding thread_create, must be called for a POSIX context.

Due to performance and architecture issues, this POSIX suite does not support all the POSIX calls. A summary of the POSIX Compliancy limitations is as follows:

- Configuration
- Initialization
- Driver and I/O model might require porting of current drivers.
- Multi-processor extensions are not supported
- Unsupported calls (please see below)
- Calls supported with certain limitations (please see list below)
- Only the following the routines can be used as pthread cancellation points:
  - mq_send
  - mq_receive
  - nanosleep
  - sleep

  - pthread_cond_timedwait
  - pthread_cond_wait
  - pthread_join
  - sem_wait

In addition, there are also certain limitations with respect to some services. See below, function details, for limitations.

1. Limitations on POSIX instances
   - SEM_NSEMS_MAX 128 Simultaneous POSIX semaphores
   - SEM_NAME_MAX 16 Maximum length of name of semaphore
   - SEM_VALUE_MAX 128 Max value of semaphore while initialization
   - POSIX_MAX_QUEUES 32 Maximum number of simultaneous POSIX message queues supported
   - PATH_MAX 10 Maximum length of name of a message queue
   - PTHREAD_THREADS_MAX 128 Define the maximum number of simultaneous POSIX Pthreads supported.
   - POSIX_MAX_MUTEX 128 Define the maximum number of simultaneous POSIX mutexes sported.
   - POSIX_DEFAULT_STACK_SIZE 4KB Default POSIX thread stack.

List of supported POSIX API

- pthread_cancel
- pthread_create
- pthread_detach
- pthread_equal
- pthread_exit
- pthread_getschedparam
- pthread_join
- pthread_once
- pthread_self
- pthread_setcancelstate
- pthread_setcanceltype
- pthread_setschedparam
- pthread_testcancel
- pthread_yield
- pthread_attr_init
- pthread_attr_destroy
- pthread_attr_getdetachstate
- pthread_attr_getinheritsched
- pthread_attr_getschedparam
- pthread_attr_getschedpolicy
- pthread_attr_getstack
- pthread_attr_getstackaddr

- pthread_attr_getstacksize
- pthread_attr_setdetachstate
- pthread_attr_setinheritsched
- pthread_attr_setschedparam
- pthread_attr_setschedpolicy
- pthread_attr_setstack
- pthread_attr_setstackaddr
- pthread_attr_setstacksize
- sched_get_priority_max
- sched_get_priority_min
- sched_yield
- sem_close
- sem_destroy
- sem_getvalue
- sem_init
- sem_open
- sem_post
- sem_trywait
- sem_unlink
- sem_wait
- pthread_mutex_destroy
- pthread_mutex_init
- pthread_mutex_lock
- pthread_mutex_timedlock
- pthread_mutex_trylock
- pthread_mutex_unlock
- pthread_mutexattr_destroy
- pthread_mutexattr_getprotocol
- pthread_mutexattr_getpshared
- pthread_mutexattr_gettype
- pthread_mutexattr_init
- pthread_mutexattr_setprotocol
- pthread_mutexattr_setpshared
- pthread_mutexattr_settype
- mq_open
- mq_send

- mq_receive
- mq_unlink
- mq_close
- pthread_cond_broadcast
- pthread_cond_destroy
- pthread_cond_init
- pthread_cond_signal
- pthread_cond_timedwait
- pthread_cond_wait
- nanosleep

Following is a list of unsupported POSIX API

- pthread_sigmask
- pthread_kill
- clock_get
- clock_set
- clock_getres

7.94.2 Function Documentation

mqd_t mq_open (const CHAR ∗ *mqName*, ULONG *oflags*, *...*) See below for function limitations.

1. The value of mode (mode_t) has no effect in this implementation.
2. If pAttr is NULL, the message queue is created with implementation-defined default message queue attributes. The default message queue attributes selected are :
   - MQ_MAXMSG 128 [MQ_MAXMSG 1024 (POSIX value)]
   - MQ_MSGSIZE 512 [MQ_MSGSIZE 4096 (POSIX value)]
   - MQ_FLAGS 0

   This is due to limitation of size of posix_region0_byte_pool (64KB).

Examples:

craton-threadx/posix/posix-example.c.

ssize_t mq_receive (mqd_t *mqdes*, VOID ∗ *pMsg*, size_t *msgLen*, ULONG ∗ *pMsgPrio*) See below for function limitations.
If a receive (or send) message from queue with out it being opened, erratic behavior may ensue.

Examples:

craton-threadx/posix/posix-example.c.

INT mq_send (mqd_t *mqdes*, const char ∗ *msg_ptr*, size_t *msg_len*, ULONG *msg_prio*) See below for function limitations.

1. In POSIX : If more than one mq_send() is blocked on a queue and space becomes available in that queue, the message with the highest priority will be unblocked. THIS FEATURE IS NOT IMPLEMENTED.
2. If a message is sent (or received) to a queue with out opening the named queue, in such a case mqdes (message queue descriptor) pointer is invalid and may result in erratic behavior.

Examples:

craton-threadx/posix/posix-example.c.

INT nanosleep (const struct timespec * *req*, struct timespec * *rem*) See below for function limitations.
Suspend by nanosleep() calls can not be awakened by signals, once in the suspension call will complete the suspension period
Sleep time is convert clock ticks rounding up to the closes clock tick so that the thread will sleep no less than the specified time

Examples:

craton-threadx/posix/posix-example.c.

INT pthread_attr_setdetachstate (pthread_attr_t * *attr*, INT *detachstate*) See below for function limitations.
Setting detach has no effect on system.

Examples:

craton-threadx/posix/posix-example.c.

INT pthread_attr_setinheritsched (pthread_attr_t * *attr*, INT *inheritsched*) See below for function limitations.
PTHREAD_INHERIT_SCHED can be set only for threads created from within a POSIX thread.

Examples:

craton-threadx/posix/posix-example.c.

INT pthread_cancel (pthread_t *thread*) See below for function limitations.
When the pthread_cancel( ) function is called the target thread is canceled with immediate effect. (provided cancelability is enabled for the target pthread)
The cancellation processing in the target thread shall run asynchronously with respect to the ailing thread returning from pthread_cancel( ).

Examples:

craton-threadx/posix/posix-example.c.

INT pthread_cond_init (pthread_cond_t * *cond*, pthread_condattr_t * *attr*) See below for function limitations.
No attributes are supported for condition variable in this implementation.

Examples:

craton-threadx/posix/posix-example.c.

INT pthread_detach (pthread_t *thread*) See below for function limitations.
Call to function does not have any effect on system.

Examples:

craton-threadx/posix/posix-example.c.

INT pthread_mutexattr_setprotocol (pthread_mutexattr_t * *attr*, INT *protocol*) See below for function limitations.
Mutex attribute is limited to the following protocol types

1. PTHREAD_PRIO_INHERIT

Examples:

craton-threadx/posix/posix-example.c.

INT pthread_mutexattr_settype (pthread_mutexattr_t * *attr*, INT *type*) See below for function limitations.
Mutex attribute is limited to the following protocol types

1. PTHREAD_MUTEX_RECURSIVE

Examples:

craton-threadx/posix/posix-example.c.

INT pthread_once (pthread_once_t * *once_control*, VOID(*)(VOID) *init_routine*) See below for function limitations.
There is no provision if the init_routine contains a cancellation point.

Examples:

craton-threadx/posix/posix-example.c.

INT sem_close (sem_t * *sem*) See below for function limitations.

1. If operation is done before creating or opening (sem_open()) the named semaphore, erratic behavior may result.
2. This routine does not deallocate any system resources.

Examples:

craton-threadx/posix/posix-example.c.

INT sem_getvalue (sem_t * *sem*, ULONG * *sval*) See below for function limitations.
If operation is done before creating or opening (sem_open()) the named semaphore, erratic behavior may result.

Examples:

craton-threadx/posix/posix-example.c.

INT sem_post (sem_t * *sem*) See below for function limitations.
If operation is done before creating or opening (sem_open()) the named semaphore, erratic behavior may result.

Examples:

craton-threadx/posix/posix-example.c.

INT sem_trywait (sem_t * *sem*) See below for function limitations.
If operation is done before creating or opening (sem_open()) the named semaphore, erratic behavior may result.

Examples:

craton-threadx/posix/posix-example.c.

INT sem_unlink (const char * *name*) See below for function limitations.

1. If operation is done before creating or opening (sem_open()) the named semaphore, erratic behavior may result.
2. EDEADLKA :->[This is a return value when deadlock condition is detected; i.e., two separate processes are waiting for an available resource to be released via a semaphore "held" by the other process.] This is not implemented.
3. EINTR :->[This is a return value when sem_wait() was interrupted by a signal.] This is not implemented.

Examples:

craton-threadx/posix/posix-example.c.

INT sem_wait (sem_t * *sem*) See below for function limitations.

If operation is done before creating or opening (sem_open()) the named semaphore, erratic behavior may result.

Examples:

craton-threadx/posix/posix-example.c.

8 Example Documentation

8.1 craton-threadx/bridge/v2x-udp-bridge-example.c

```
/* Copyright (C) 2013-2015 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>

#include <tx_api.h>
#include <nx_api.h>

#include <atlk/v2x.h>
#include <atlk/v2x_service.h>

#include <craton/net.h>

/*
  CRATON V2X-UDP Bridge Example

  This example demonstrates using the V2X API and the NetX-Duo API in order
  to construct a V2X-UDP bridge application.

  To simplify the example, server IP address is hard-coded under the
  SERVER_ADDR define. Users are free to change this value as required.

  Two threads are created -- a TX thread and a RX thread. A V2X service is
  retrieved and a V2X socket is created, bound to Protocol ID 0x0FA1; these
  are used by both threads. Additionally, a NetX-Duo UDP socket is created,
  bound to port 2002.

  The TX thread receives UDP frames and transmits their content via the V2X
  API. The RX thread receives frames from V2X API and transmits their content
  over UDP to the server.

  @todo This example is not currently supported in multi-core SDK.
*/

/* UDP server IP address */
#define SERVER_ADDR (10 << 24 | 10 << 16 | 1 << 8 | 121 << 0)

/* UDP ports to receive and transmit on */
#define UDP_TX_PORT 2001
#define UDP_RX_PORT 2002

/* V2X Protocol ID */
#define PROTOCOL_ID 0x0FA1ULL

/* Used V2X interface index */
#define IF_INDEX 1

/* Maximum message size (excluding IP/UDP headers) */
#define MESSAGE_SIZE_MAX 1450

/* Example threads priorities */
#define TX_THREAD_PRIORITY 40
#define RX_THREAD_PRIORITY 41

#if defined __CRATON_NO_ARC || defined __CRATON_ARC1

/* Transmitting thread */
static TX_THREAD tx_thread;
static uint8_t tx_thread_stack[0x2000];
static void tx_thread_entry(ULONG input);

/* Receiving thread */
static TX_THREAD rx_thread;
static uint8_t rx_thread_stack[0x2000];
static void rx_thread_entry(ULONG input);

/* Shared V2X service */
static v2x_service_t *v2x_service = NULL;

/* Shared V2X socket */
static v2x_socket_t *v2x_socket = NULL;
```

```c
/* Shared UDP socket */
static NX_UDP_SOCKET udp_socket;

/* Trusted IP instance packet pool */
NX_PACKET_POOL *packet_pool = NULL;

/* Cleanup any allocated resources */
static void cleanup(void)
{
  v2x_socket_delete(v2x_socket);
  v2x_service_delete(v2x_service);
}

void craton_user_init(void)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* NetX return value */
  ULONG nrv = NX_SUCCESS;
  /* ThreadX return value */
  ULONG trv = TX_SUCCESS;
  /* V2X socket configuration */
  v2x_socket_config_t socket_config = V2X_SOCKET_CONFIG_INIT;
  /* NetX trusted IP instance */
  NX_IP *ip_instance = NULL;

  /* Get default V2X service instance */
  rc = v2x_default_service_get(&v2x_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "v2x_default_service_get: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Set socket configuration */
  socket_config.if_index = IF_INDEX;
  socket_config.protocol.protocol_id = PROTOCOL_ID;

  /* Create a V2X socket */
  rc = v2x_socket_create(v2x_service, &v2x_socket, &socket_config);
  if (atlk_error(rc)) {
    fprintf(stderr, "v2x_socket_create: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Get trusted NetX-Duo IP instance */
  rc = net_ip_trusted_instance_get(&ip_instance);
  if (atlk_error(rc)) {
    fprintf(stderr, "net_ip_trusted_instance_get: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Set pointer to trusted IP instance packet pool */
  packet_pool = ip_instance->nx_ip_default_packet_pool;

  /* Create a UDP socket */
  nrv = nx_udp_socket_create(ip_instance, &udp_socket, "udp_socket",
                             NX_IP_NORMAL, NX_FRAGMENT_OKAY, 0x80, 20);
  assert(nrv == NX_SUCCESS);

  /* Bind the UDP socket to the UDP receive port */
  nrv = nx_udp_socket_bind(&udp_socket, UDP_RX_PORT, NX_NO_WAIT);
  assert(nrv == NX_SUCCESS);

  /* Create TX thread */
  trv = tx_thread_create(&tx_thread, "tx_thread",
                         tx_thread_entry, 0,
                         tx_thread_stack,
                         sizeof(tx_thread_stack),
                         TX_THREAD_PRIORITY,
                         TX_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  /* Create RX thread */
  trv = tx_thread_create(&rx_thread, "rx_thread",
                         rx_thread_entry, 0,
                         rx_thread_stack,
                         sizeof(rx_thread_stack),
                         RX_THREAD_PRIORITY,
                         RX_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  return;

error:
  cleanup();
}

void tx_thread_entry(ULONG input)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* NetX return value */
  ULONG nrv = NX_SUCCESS;
  /* Send parameters */
  v2x_send_params_t send_params = V2X_SEND_PARAMS_INIT;
  /* NetX packet */
  NX_PACKET *udp_packet = NULL;
  /* Message buffer */
  uint8_t message[MESSAGE_SIZE_MAX];
  /* Message size */
  size_t size;
  /* Not using input */
  (void)input;

  while (1) {
    /* Receive a UDP packet (wait forever until it arrives) */
    nrv = nx_udp_socket_receive(&udp_socket, &udp_packet, TX_WAIT_FOREVER);
    assert(nrv == NX_SUCCESS);

    /* Copy packet data into local message buffer */
    size = sizeof(message);
        nrv = nx_packet_data_retrieve(udp_packet, message, (ULONG *)&size);
    assert(nrv == NX_SUCCESS);

        /* Release UDP packet */
    nrv = nx_packet_release(udp_packet);
    assert(nrv == NX_SUCCESS);

    /* Transmit V2X PDU */
    rc = v2x_send(v2x_socket, message, size, &send_params, NULL);
    if (atlk_error(rc)) {
      fprintf(stderr, "v2x_send: %s\n", atlk_rc_to_str(rc));
      goto error;
    }
  }

error:
  cleanup();
}

void rx_thread_entry(ULONG input)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* NetX return value */
  ULONG nrv = NX_SUCCESS;
  /* Received V2X parameters */
  v2x_receive_params_t receive_params =
      V2X_RECEIVE_PARAMS_INIT;
  /* NetX packet */
  NX_PACKET *udp_packet = NULL;
  /* Message buffer */
  uint8_t message[MESSAGE_SIZE_MAX];
  /* Message size */
  size_t size;
  /* Not using input */
  (void)input;

  while (1) {
    /* Receive frame (wait forever until it arrives) */
    size = sizeof(message);
    rc = v2x_receive(v2x_socket, message, &size, &receive_params,
                     &atlk_wait_forever);
    if (atlk_error(rc)) {
      fprintf(stderr, "v2x_receive: %s\n", atlk_rc_to_str(rc));
      goto error;
    }

    /* Allocate a packet from the packet pool */
    nrv = nx_packet_allocate(packet_pool, &udp_packet,
                             NX_UDP_PACKET, TX_WAIT_FOREVER);
    assert(nrv == NX_SUCCESS);

    /* Copy received V2X message into packet data */
    nx_packet_data_append(udp_packet, message, size,
                          packet_pool, TX_WAIT_FOREVER);

    /* Send UDP packet */
    nrv = nx_udp_socket_send(&udp_socket, udp_packet,
                             SERVER_ADDR, UDP_TX_PORT);
    assert(nrv == NX_SUCCESS);
  }

error:
  cleanup();
}

#else /* __CRATON_NO_ARC || __CRATON_ARC1 */

void craton_user_init(void)
{
}

#endif  /* __CRATON_NO_ARC || __CRATON_ARC1 */
```

8.2 craton-threadx/bt-spi2uart/bt-spi2uart-example.c

```
/* Copyright (C) 2015 Autotalks Ltd. */
#include <stdio.h>
#include <stdbool.h>
#include <inttypes.h>
#include <unistd.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>

#include <sys/time.h>

#include <tx_api.h>
#include <tx_posix.h>

#include <libcli.h>

#include <atlk/sdk.h>

#include <craton/uart_driver.h>
#include <craton/cli.h>

/* TODO: This example is a draft -- requires simplification */

/*
  CRATON Bluetooth device connected via SPI2UART Example

  This example demonstrates basic usage of the Bluetooth device over SPI2UART
  for code running on top of CRATON processor with ThreadX RTOS.

  The example demonstrates how to initialize, send and receive data
  via SPI2UART to/from the Bluetooth device.

  The flow of the example is:
  3 CLI commands are initialized at power-up:
    * bt_init - Used to initialize the Bluetooth device.
    * bt_tx   - Used to send a data to the Bluetooth device.
    * bt_rx   - Used to receive data from the Bluetooth device.

  NOTE: Host should initiate the pairing (binding) and connect to the CRATON
        over Bluetooth following CRATON power-up.
*/

/* BT device command strings */
#define BT_DEVICE_MAX_CMD_STRING_SIZE 256
/* BT device response strings */
#define MAX_BT_RESPONSE_STRLEN 256

/* SPBT2632C2A Bluetooth chip AT commands */
#define BT_ENTER_COMMAND_MODE                  "^#^$^%"
#define BT_ENTER_CMD_EXPECTED_RESPONSE        "AT-AB -CommandMode-"
#define BT_CHANGE_BAUDRATE_GENERIC_CMD        "AT+AB ChangeBaud"
#define BT_CHANGE_BAUDRATE_EXPECTED_RESPONSE  "AT-AB Baudrate Changed"
#define BT_FLOW_CONTROL_ENABLE_CMD            "AT+AB StreamingSerial Disable"
#define BT_FLOW_CONTROL_ENABLE_EXPECTED_RESP  "AT-AB StreamingSerial Disabled"

#define CLI_PRINT(fmt, ...) cli_print(cli, fmt, ## __VA_ARGS__)

/* Global */
/* File descriptor for BT device */
static int serial_fd;
static bool b_bt_device_initialized = false;

/* BT Command buffer */
char g_bt_cmd_str[BT_DEVICE_MAX_CMD_STRING_SIZE];
/* BT change baudrate command buffer */
char g_bt_cmd_change_baudrate_str[BT_DEVICE_MAX_CMD_STRING_SIZE];
/* BT response buffer */
static char g_bt_response_buf[MAX_BT_RESPONSE_STRLEN];

/* BT device name */
```

```c
static char *uart_name = "/dev/uart2";

/* Definitions */
typedef struct {
  char *request;
  char *expected_response;
} bluetooth_device_init_script_req_reply_t;

/* Global */
/*
 * SPBT2632C2A BT device Init script:
 * =================================
 * bt_tx ^#^$^%
 * bt_rx 256
 * bt_tx AT+AB ChangeBaud 921600
 * bt_rx 256
 * bt_uart_hw_baudrate_change 921600
 * */
bluetooth_device_init_script_req_reply_t g_bt_SPBT2632C2A_init_script[] = {
    { BT_ENTER_COMMAND_MODE, BT_ENTER_CMD_EXPECTED_RESPONSE },
    { BT_FLOW_CONTROL_ENABLE_CMD, BT_FLOW_CONTROL_ENABLE_EXPECTED_RESP },
    { g_bt_cmd_change_baudrate_str, BT_CHANGE_BAUDRATE_EXPECTED_RESPONSE }
};

static int cmd_bt_device_init(struct cli_def *cli,
                              const char *command,
                              char *argv[],
                              int argc);

static int cmd_bt_tx(struct cli_def *cli,
                     const char *command,
                     char *argv[],
                     int argc);

static int cmd_bt_rx(struct cli_def *cli,
                     const char *command,
                     char *argv[],
                     int argc);

static void print_hex(struct cli_def *cli, char *buf, unsigned int size);

static ssize_t bt_response_handle(struct cli_def *cli,
                                  int fd,
                                  char *rx_buf,
                                  uint32_t size);

static atlk_rc_t bt_device_init(struct cli_def *cli, unsigned int baudrate)
{
  int ret;
  int size;
  int num_commands;
  int i;
  int size_to_read;
  ssize_t len;
  atlk_rc_t rc = ATLK_OK;

  /* Prepare baudrate change BT AT command */
  ret = snprintf(g_bt_cmd_change_baudrate_str,
                 sizeof(g_bt_cmd_change_baudrate_str),
                 "%s %d",
                 BT_CHANGE_BAUDRATE_GENERIC_CMD,
                 baudrate);
  if (ret < 0) {
    CLI_PRINT("%s", "bt_device_init(): Error 1");
    return ATLK_E_UNSPECIFIED;
  }

  num_commands = sizeof(g_bt_SPBT2632C2A_init_script) /
                 sizeof(bluetooth_device_init_script_req_reply_t);

  CLI_PRINT("Starting BT initialization script, num_commands = %d\n",
            num_commands);

  /* Send initialization script and check responses from BT device */
  for (i = 0; i < num_commands; i++)
  {
    if (0 == strcmp(BT_ENTER_COMMAND_MODE,
                    g_bt_SPBT2632C2A_init_script[i].request)) {
      size = snprintf(g_bt_cmd_str,
                      sizeof(g_bt_cmd_str),
                      "%s",
                      g_bt_SPBT2632C2A_init_script[i].request);
    }
    else {
      /* Need '\n' at the end to issue the AT-command to BT device */
      size = snprintf(g_bt_cmd_str,
                      sizeof(g_bt_cmd_str),
                       "%s\r\n",
                       g_bt_SPBT2632C2A_init_script[i].request);
    }

    CLI_PRINT("[%d] Sending BT command: %s\n", i, g_bt_cmd_str);

    len = write(serial_fd, g_bt_cmd_str, size);
    if (len < 0) {
      CLI_PRINT("Error! Failed to write to BT, len = %d\n", len);
    }
    else {
      CLI_PRINT("BT command successfully written to device, len = %d",
                 len);
    }

    /* Wait for BT device to respond, it takes some time for the BT device
     * to respond and send the response to the SPI2UART device...
     * It takes about ~0.009sec == 9msec ((1/115200)*9*128) to write 128
     * bytes at 115200 (the slowest UART baudrate) ==>
     * so we wait 20msec. */
    usleep(20000);

    /* Handle response */
    size = strlen(g_bt_SPBT2632C2A_init_script[i].expected_response);

    /* The read API of the SPI2UART driver is non-blocking ==>
     * We read up to 256 bytes but if only 10 bytes are received at the
     * time of reading it will not block.
     */
    size_to_read = MAX_BT_RESPONSE_STRLEN;

    CLI_PRINT("[%d] Expecting BT response: %s , size = %d, "
              "size_in_rx_fifo=%d\n",
               i,
               g_bt_SPBT2632C2A_init_script[i].expected_response,
               size,
               size_to_read);

    /* Read response from UART */
    len = bt_response_handle(cli, serial_fd, g_bt_cmd_str, size_to_read);
    g_bt_cmd_str[size] = (char )0x00;

    CLI_PRINT("BT response: %s len=%d...    ", g_bt_cmd_str, len);
    if (0 != strcmp(g_bt_cmd_str,
                    g_bt_SPBT2632C2A_init_script[i].expected_response))
    {
      CLI_PRINT("%s", "Error! BT response is not equal to Expected_response.\n");
      CLI_PRINT("%s", "Aborting...\n");
      return ATLK_E_UNSPECIFIED;
    }
    else {
      CLI_PRINT("%s", "OK\n");
    }
  }

  /* Now it is OK to change the baudrate of the SPI2UART device */
  rc = uart_speed_set(UART_DEVICE_ID_2, baudrate);
  if (atlk_error(rc)) {
    CLI_PRINT("Error! Failed to change UART baudrate, rc = %d\n", rc);
    return rc;
  }

  CLI_PRINT("%s", "BT initialization sequence successful.\n");

  return rc;
}

static void print_hex(struct cli_def *cli, char *buf, unsigned int size)
{
  const unsigned int line_size_hex_bytes = 8;
  unsigned int i;
  unsigned int j;
  unsigned int max;
  unsigned int delta;
  char hex_output_buf[line_size_hex_bytes * 8];

  i = 0;
  while (i < size)
  {
    max = line_size_hex_bytes;
    delta = size - i;
    max = (delta < 8) ? delta : line_size_hex_bytes;

    /* Empty string */
    hex_output_buf[0] = 0x00;
    for (j = 0; j < max; j++)
    {
      snprintf(hex_output_buf,
                sizeof(hex_output_buf),
                "%s %02x",
                hex_output_buf,
                buf[i + j]);
    }

    CLI_PRINT("%s", hex_output_buf);

    i += max;
  }

}

static ssize_t bt_response_handle(struct cli_def *cli,
                                  int fd,
                                  char *rx_buf,
                                  uint32_t size)
{
  ssize_t len = 0;

  len = read(fd, rx_buf, (ssize_t)size);
  if (len < 0) {
    fprintf(stderr, "Error! Failed to read from BT");
    return -1;
  }

  CLI_PRINT("bt_response_handle(): BT receive: Len = %d\n data_rx:", (int)len);

  print_hex(cli, rx_buf, len);

  return len;
}

atlk_rc_t bt_spi2uart_example_cli_connect(int serial_file_desc)
{
  atlk_rc_t rc;
  unsigned int i;
  cli_instance_t *cli = NULL;

  /* BT device not initialized yet ... */
  b_bt_device_initialized = FALSE;

  /* Initialize file descriptor for SPI2UART */
  serial_fd = serial_file_desc;

  for (i = CLI_INSTANCE_TYPE_UART; i <=
      CLI_INSTANCE_TYPE_TELNET; i++)
  {
    /* Get CRATON UART CLI instance */
    rc = cli_instance_get(&cli, i);
    if (atlk_error(rc)) {
      return rc;
    }

    /* register bt_tx CMD */
    cli_register_command(cli, NULL, "bt_tx", cmd_bt_tx,
        PRIVILEGE_UNPRIVILEGED, MODE_ANY, "Sends data over BT UART");

    /* register bt_rx CMD */
    cli_register_command(cli, NULL, "bt_rx", cmd_bt_rx,
           PRIVILEGE_UNPRIVILEGED, MODE_ANY, "Receives data from BT UART");

    /* register bt_spi2uart_example_bt_device_init CMD */
    cli_register_command(cli,
                          NULL,
                          "bt_device_init",
                          cmd_bt_device_init,
                          PRIVILEGE_UNPRIVILEGED, MODE_ANY,
                          "Initializes bluetooth device");
  }

  return ATLK_OK;
}


static int cmd_bt_device_init(struct cli_def *cli,
                              const char *command,
                              char *argv[],
                              int argc)
{
  atlk_rc_t rc;
  unsigned int baudrate;

  if (b_bt_device_initialized)
  {
    CLI_PRINT("%s", "Error! BT device already initialized...");
    return CLI_ERROR_ARG;
  }

  if ('?' == argv[0][0]) {
    CLI_PRINT("%s", command);
    return CLI_OK;
  }

  if (argc < 1) {
    CLI_PRINT("%s", "Initializes BT device\n"
              "usage: <baudrate>\n");
    return CLI_ERROR_ARG;
  }
  else {
    baudrate = atol(argv[0]);
    CLI_PRINT("baudrate = %d\n", baudrate);

    switch (baudrate) {
    case UART_SPEED_921600_BPS:
    case UART_SPEED_460800_BPS:
    case UART_SPEED_230400_BPS:
    case UART_SPEED_115200_BPS:
      break;
    default:
      CLI_PRINT("%s", "Error! invalid baudrate.");
      CLI_PRINT("%s", "Valid baudrates are: 921600/460800/230400/115200\n");
      return CLI_ERROR;
    }
  }

  rc = bt_device_init(cli, baudrate);
  if (atlk_error(rc)) {
    CLI_PRINT("Error! Failed to initialize the bluetooth device "
              "rc = %d\n", rc);
    return rc;
  }

  b_bt_device_initialized = TRUE;

  return CLI_OK;
}

static int cmd_bt_tx(struct cli_def *cli, const char *command,
                                char *argv[], int argc)
{
  ssize_t len;
  char *str_to_tx;
  int i;
  char str[256];

  if ('?' == argv[0][0]) {
    CLI_PRINT("%s", command);
    return CLI_OK;
  }

  if (argc < 1) {
    CLI_PRINT("%s", "bt_tx - Sends data to Bluetooth device\n"
                    "usage: bt_tx [string#1] [string#2] ... [string#N-1]");
    return CLI_ERROR_ARG;
  }
  else {
    if (argc > 1)
    {
      strcpy(str, argv[0]);
      for (i = 1; i < argc; i++)
      {
        snprintf(str, sizeof(str), "%s %s", str, argv[i]);
      }
      snprintf(str, sizeof(str), "%s\n", str);

      str_to_tx = str;
    }
    else {
        str_to_tx = argv[0];
    }

    CLI_PRINT("str_to_tx = %s\n", str_to_tx);
  }

  len = write(serial_fd, str_to_tx, strlen(str_to_tx));
  if (len < 0) {
    fprintf(stderr, "Failed to write to BT\n");
  }
  else {
    CLI_PRINT("bt_tx successfully written "
               "string=%s to BT UART, len=%d\n",
               str_to_tx,
               len);
  }

  return CLI_OK;
}

static int cmd_bt_rx(struct cli_def *cli, const char *command,
    char *argv[], int argc)
{
  int len;
  unsigned int size_to_rx;

  if ('?' == argv[0][0]) {
    CLI_PRINT("%s", command);
    return CLI_OK;
  }

  if (argc < 1) {
    CLI_PRINT("%s", "bt_rx - Receives data from Bluetooth device\n"
                    "usage: bt_rx [num_bytes_to_rx]");
    return CLI_ERROR_ARG;
  }
  else {
    size_to_rx = atol(argv[0]);
    CLI_PRINT("size_to_rx = %d\n", size_to_rx);
    if (size_to_rx > MAX_BT_RESPONSE_STRLEN)
    {
      CLI_PRINT("Error! Max size_to_rx is %d\n", MAX_BT_RESPONSE_STRLEN);
      return CLI_ERROR_ARG;
    }
  }

  /* Read response from UART */
  len = bt_response_handle(cli, serial_fd, g_bt_response_buf, size_to_rx);
  g_bt_response_buf[len] = (char )0x00;
  CLI_PRINT("\nBT response: %s, len=%d\n", g_bt_response_buf, len);

  return CLI_OK;
}

atlk_rc_t bt_spi2uart_example_init(void)
{
  /* atlk return code */
  atlk_rc_t rc = ATLK_OK;

  printf("UART name is %s\n", uart_name);

  /* Open BT device */
  serial_fd = open(uart_name, 0);
  if (serial_fd < 0) {
    fprintf(stderr, "Failed to open \"%s\", rc=%d\n", uart_name, serial_fd);
    return ATLK_E_UNSPECIFIED;
  }

  /* Connect CLI */
  rc = bt_spi2uart_example_cli_connect(serial_fd);
  if (atlk_error(rc)) {
    fprintf(stderr, "Failed to connect to CRATON CLI\n");
    return ATLK_E_UNSPECIFIED;
  }

  return rc;
}

void craton_user_init(void)
{
  atlk_rc_t rc = ATLK_OK;

  printf("Initialize bt-spi2uart example ...");

  rc = bt_spi2uart_example_init();
  if (atlk_error(rc)) {
    fprintf(stderr,
            "bt-spi2uart example init failed: %s\n",
            atlk_rc_to_str(rc));
    return;
  }

  printf("OK\n");
}
```

8.3 craton-threadx/build/main.c

```
/* Copyright (C) 2014-2015 Autotalks Ltd. */
#include <unistd.h>
```

```
#include "unit.h"

#ifdef _DO_NOTHING

void craton_user_init(void)
{
}

#else /* _DO_NOTHING */

void craton_user_init(void)
{
#ifdef _INSERT_DELAY
  usleep(2000000);
#endif

  cxx_unit_test();
}

#endif /* _DO_NOTHING */
```

8.4 craton-threadx/build/unit.h

```
/* Copyright (C) 2014-2015 Autotalks Ltd. */
#ifndef _UNIT_H
#define _UNIT_H

#ifdef __cplusplus
extern "C" {
#endif

/* Run C++11 demo unit test */
void cxx_unit_test(void);

#ifdef __cplusplus
}
#endif

#endif /* _UNIT_H */
```

8.5 craton-threadx/can/can-example.c

```
/* Copyright (C) 2014-2015 Autotalks Ltd. */
#include <stdio.h>
#include <stdint.h>
#include <assert.h>
#include <unistd.h>

#include <tx_api.h>

#include <atlk/sdk.h>
#include <atlk/can.h>
#include <atlk/can_service.h>

/*
  CRATON ThreadX CAN Example

  This example demonstrates basic usage of Autotalks CAN API
  for code running on top of CRATON processor with ThreadX RTOS.

  Two threads are created -- sending thread and receiving thread.

  The sending thread transmits a 5-octet CAN message with ID 0x50,
  every 300 milliseconds. The receiving thread receives CAN messages
  and displays them on the system's debug console.

  To see the demo in action you should execute can-example.img on two
  PANGAEA4 units whose CAN ports are connected.
*/

/* CAN ID used in this example */
#define EXAMPLE_CAN_ID 0x50UL

/* CAN device ID used in this example */
#define EXAMPLE_CAN_DEVICE_ID 0

/* Thread priorities */
#define CAN_SEND_THREAD_PRIORITY 40
#define CAN_RECEIVE_THREAD_PRIORITY 41

/* CAN send thread */
static TX_THREAD can_send_thread;
static uint8_t can_send_thread_stack[0x1000];
static void can_send_thread_entry(ULONG input);
```

```c
/* CAN receive thread */
static TX_THREAD can_receive_thread;
static uint8_t can_receive_thread_stack[0x1000];
static void can_receive_thread_entry(ULONG input);

/* CAN service */
static can_service_t *can_service = NULL;

/* CAN socket */
static can_socket_t *can_socket = NULL;

/* CAN ID filter array */
static const can_id_filter_t filter_array[] = {
  { .can_id = 0, .can_id_mask = 0 },
};

/* Release allocated resources */
static void example_cleanup(void)
{
  can_socket_delete(can_socket);
  can_service_delete(can_service);
}

void craton_user_init(void)
{
  /* ThreadX return value */
  ULONG trv = TX_SUCCESS;
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* CAN socket configuration */
  can_socket_config_t socket_config = CAN_SOCKET_CONFIG_INIT;

  /* Get default CAN service instance */
  rc = can_default_service_get(&can_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "can_default_service_get: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Set socket configuration */
  socket_config.filter_array_ptr = filter_array;
  socket_config.filter_array_size = 1;
  socket_config.device_id = EXAMPLE_CAN_DEVICE_ID;

  /* Create CAN socket */
  rc = can_socket_create(can_service, &can_socket, &socket_config);
  if (atlk_error(rc)) {
    fprintf(stderr, "can_socket_create: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Create CAN send thread */
  trv = tx_thread_create(&can_send_thread, "can_send_thread",
                         can_send_thread_entry, 0,
                         can_send_thread_stack,
                         sizeof(can_send_thread_stack),
                         CAN_SEND_THREAD_PRIORITY,
                         CAN_SEND_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  /* Create CAN receive thread */
  trv = tx_thread_create(&can_receive_thread, "can_receive_thread",
                         can_receive_thread_entry, 0,
                         can_receive_thread_stack,
                         sizeof(can_receive_thread_stack),
                         CAN_RECEIVE_THREAD_PRIORITY,
                         CAN_RECEIVE_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  return;

error:
  example_cleanup();
}

void can_send_thread_entry(ULONG input)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* CAN message data to send */
  uint8_t data[] = { 1, 2, 3, 4, 5 };
  /* Not using input */
  (void)input;
  while (1) {
    /* Send CAN message with CAN ID 0x50 */
    rc = can_send(can_socket, data, sizeof(data), EXAMPLE_CAN_ID, NULL);
    if (rc == ATLK_E_OUT_OF_MEMORY) {
      /* Sleeping for 1 millisecond to avoid a busy loop. The 'out of memory'
         error is expected when the TX queue is full.
       */
      usleep(1000);
      continue;
    }
    else if (atlk_error(rc)) {
      fprintf(stderr, "can_send: %s\n", atlk_rc_to_str(rc));
      goto error;
    }

    /* Print sent CAN message */
    printf("Example: Sent CAN message \"%d,%d,%d,%d,%d\", ID 0x%lx\n",
           data[0], data[1], data[2], data[3], data[4], EXAMPLE_CAN_ID);

    /* Change message content */
    data[0]++;

    /* Wait for 300 milliseconds */
    usleep(300000);
  }

error:
  example_cleanup();
}

void can_receive_thread_entry(ULONG input)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* Received CAN message data */
  uint8_t data[CAN_DATA_SIZE_MAX];
  /* Received CAN message data size */
  size_t data_size = sizeof(data);
  /* Received CAN ID */
  can_id_t can_id;
  /* Not using input */
  (void)input;

  while (1) {
    /* Receive CAN message */
    rc = can_receive(can_socket, data, &data_size, &can_id,
                     &atlk_wait_forever);
    if (atlk_error(rc)) {
      fprintf(stderr, "can_receive: %s\n", atlk_rc_to_str(rc));
      goto error;
    }

    /* Print received CAN message */
    printf("Example: Received CAN message \"%d,%d,%d,%d,%d\", ID 0x%lx\n",
           data[0], data[1], data[2], data[3], data[4], can_id);
  }

error:
  example_cleanup();
}
```

8.6 craton-threadx/can/can-hw-filter-example.c

```
/* Copyright (C) 2015 Autotalks Ltd. */
#include <stdio.h>
#include <stdint.h>
#include <inttypes.h>
#include <unistd.h>

#include <atlk/sdk.h>
#include <atlk/can.h>

#include <craton/can_driver.h>

/*
  CRATON ThreadX CAN Driver Example

  This example demonstrates how to configure CAN HW filters via CAN
  driver API.

  In HW RX buffers, CAN ID is represented as follows:

  For standard CAN ID (11 bits ID):
  =================================
  Bit 31  30  29  28  27  26  25  24  23  22  21  20 19       18...0
    +----+---+---+---+---+---+---+---+---+---+---+--+---+---------------+
    |ID10|ID9|ID8|ID7|ID6|ID5|ID4|ID3|ID2|ID1|ID0|DC|IDE|Don’t Care (DC)|
    +----+---+---+---+---+---+---+---+---+---+---+--+---+---------------+
  DC stands for "Don’t care". These bits are not related to CAN ID.
  IDE stands for "ID Extenstion". This bit is 1 when we receive a frame with
  extended ID.

  For extended CAN ID (29 bits ID):
  =================================
  Bit 31   30   29   28   27   26   25   24   23   22   21  20 19   18   17   16
    +----+----+----+----+----+----+----+----+----+----+----+--+---+--------------+
    |ID28|ID27|ID26|ID25|ID24|ID23|ID22|ID21|ID20|ID19|ID18|DC|IDE|ID17|ID16|ID15|
    +----+----+----+----+----+----+----+----+----+----+----+--+---+--------------+
  Bit 15   14   13   12   11   10  9   8   7   6   5   4   3   2   1   0
    +----+----+----+----+----+---+---+---+---+---+---+---+---+---+---+--+
    |ID14|ID13|ID12|ID11|ID10|ID9|ID8|ID7|ID6|ID5|ID4|ID3|ID2|ID1|ID0|DC|
    +----+----+----+----+----+---+---+---+---+---+---+---+---+---+---+--+
  DC stands for "Don’t care". These bits are not related to CAN ID.
  IDE stands for "ID Extenstion". This bit is 1 when we receive a frame with
  extended ID.

  Bits in GMASK work as follows:
  ==============================
  1 means: Don’t care about the corresponding bit in RX buffer, always accept.
  0 means: Match that corresponding bit of RX buffer with the bit specified in
           can_hw_buffer_config_t.buffer[<buffer_number>].id: If they don’t
           match, drop the frame.

  GMASK works on buffers 0..13 when they are configured to RX.

  Bits in BMASK works the same but for buffer number 14 when it is configured to
  RX.

  This example will demonstrate usage of HW filter on a single RX buffer for
  simplicity. Different HW filters may apply to different RX buffers.

  The example will have the following scenarios:

  Scenario 0: Set global mask to match exact ID, which is 0xA8, of standard
              frames.
  Scenario 1: Set global mask not to care about anything. Accept everything.
              Will accept any standard/extended frame
  Scenario 2: Set global mask to match exact first three bits of extended
              CAN ID. The three bits should be 101b (0x5)
  Scenario 3: Set global mask to match extended CAN IDs only
  Scenario 4: Set global mask to match standard CAN IDs only

  You can choose which scenario to run at compile-time.

  For more information, please refer to the CAN user guide.
*/

/* Helper macros */
#define BIT(N) (1U << (N))
#define BITMASK(N) ((1U << (N)) - 1)

#define IDE_BIT BIT(19)

#ifdef __CRATON_ARM

/*
  Helper function to create HW CAN ID for GMASK, BMASK and HW RX buffer
  CAN ID.
 */
static uint32_t
can_hw_id_from_can_id(uint32_t can_id,
                      int is_extended_id,
                      int is_mask)
{
  uint32_t can_hw_id;
  uint32_t chunk;

  can_hw_id = 0;

  if (is_extended_id) {
    /* These bits are don’t care bit in extended frames */
    can_hw_id |= BIT(0) | BIT(20);

    /* Take first chunk of 18 bits of CAN ID. Chunk is at offset of 1 bit */
    chunk = (can_id & BITMASK(18)) << 1;
    if (is_mask) {
      /* If this is GMASK or BMASK, care about these bits (0 means match) */
      chunk = ~chunk;
    }
    else {
      /* Turn extended bit ON in case of buffer CAN ID (is_mask == 0) */
      can_hw_id |= IDE_BIT;
    }
    /* Take relevant bits [18:1] */
    can_hw_id |= (0x7FFE & chunk);

    /* Take second chunk of 11 bits of CAN ID. Chunk is at offset of 21 bits */
    can_id >>= 18;
    chunk = (can_id & BITMASK(11)) << 21;
    if (is_mask) {
      /* If this is GMASK or BMASK, care about these bits (0 means match) */
      chunk = ~chunk;
    }

    /* Take relevant bits [31:21] */
    can_hw_id |= (0xFFE00000 & chunk);

    return can_hw_id;
  }

  /* Standard frames */

  /* These bits are don't care bit in standard frames */
  can_hw_id |= BITMASK(19) | BIT(20);

  /* Take 11 bits of CAN ID. CAN ID is at offset of 21 bits */
  chunk = (can_id & BITMASK(11)) << 21;
  if (is_mask) {
    /* If this is GMASK or BMASK, care about these bits (0 means match) */
    chunk = ~chunk;
  }

  /* Take relevant bits [31:21] */
  can_hw_id |= (0xFFE00000 & chunk);

  return can_hw_id;
}

/*
   Create a hardware mask where can_id_bits specifies
   which bits are compared, as follows:
   If bit N is 0: Don't care about bit in position N
   If bit N is 1: Match incoming bit in position N
 */
static uint32_t
gmask_can_id_check_mask(uint32_t can_id_bits,
                        int is_extended_id)
{
  return can_hw_id_from_can_id(can_id_bits,
                               is_extended_id,
                               1);
}

/*
   Create a hardware CAN ID where can_id specifies
   the desired CAN ID to be matched.
 */
static uint32_t
can_hw_id_create(uint32_t can_id,
                 int is_extended_id)
{
  return can_hw_id_from_can_id(can_id,
                               is_extended_id,
                               0);
}

void craton_user_init(void)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* CAN HW buffer configuration */
  can_hw_buffer_config_t config =
      CAN_HW_BUFFER_CONFIG_INIT;
  /* Scenario to test. Change to any one of the below scenarios */
  int scenario = 0;

  /* Set buffer 0 as RX buffer */
  config.buffers[0].direction = CAN_HW_BUFFER_DIRECTION_RX;

  /* Set buffers 1-14 as TX buffers; there's no need to set ID */
  for (int i = 1; i < 15; ++i) {
    config.buffers[i].direction = CAN_HW_BUFFER_DIRECTION_TX;
  }

  switch(scenario) {
  case 0:
    /* Set global mask to match exact ID of standard frames which is 0xA8 */
    config.gmask = gmask_can_id_check_mask(0x7FF, 0);
    config.buffers[0].id = can_hw_id_create(0xA8, 0);
    break;
  case 1:
    /* Set global mask not to care about anything. Accept everything.
       Will accept any standard/extended frame */
    config.gmask = 0xFFFFFFFF;
    /* buffer ID doesn't play a role here */
    break;
  case 2:
    /* Set global mask to match exact first three bits of extended CAN ID.
       The three bits should be 101b (0x5) */
    config.gmask = gmask_can_id_check_mask(0x7, 1);
    config.buffers[0].id = can_hw_id_create(0x5, 1);
    break;
  case 3:
    /* Set global mask to match extended CAN IDs only */
    config.gmask = gmask_can_id_check_mask(0, 1);
    config.buffers[0].id = can_hw_id_create(0, 1);
    break;
  case 4:
    /* Set global mask to match standard CAN IDs only */
    config.gmask = gmask_can_id_check_mask(0, 0);
    config.buffers[0].id = can_hw_id_create(0, 0);
    break;
  default:
    fprintf(stderr, "Invalid scenario number: %d\n", scenario);
    return;
  }

  /* Set CAN HW buffer configuration for device 0 */
  rc = can_hw_buffer_config_set(0, &config);
  if (atlk_error(rc)) {
    fprintf(stderr, "can_hw_config_set: %s\n", atlk_rc_to_str(rc));
    return;
  }

  printf("CAN HW configuration done.\n");
  printf("GMASK = %"PRIx32" , ID = %"PRIx32"\n",
         config.gmask, config.buffers[0].id);

  return;
}

#else /* __CRATON_ARM */

void craton_user_init(void)
{
}

#endif /* __CRATON_ARM */
```

8.7 craton-threadx/cli/cli-example.c

```
/* Copyright (C) 2014-2015 Autotalks Ltd. */
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <inttypes.h>

#include <libcli.h>

#include <craton/cli.h>

/*
  CRATON ThreadX CLI Example

  This example demonstrates basic usage of CRATON CLI API and libcli's API
  for code running on top of a CRATON processor with ThreadX RTOS.

  A CLI instance is retrieved via CRATON CLI API. The instance is used to
  register a basic CLI command which prints the message "Example command
  executed successfully" to the console when invoked.
*/

int cmd_example(cli_instance_t *cli, const char *command,
                char *argv[], int argc)
{
  /* Unused parameters */
  (void)command;
  (void)argv;
  (void)argc;

  cli_print(cli, "%s", "Example command executed successfully.");

  return CLI_OK;
}
```

```
void craton_user_init(void)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* CLI instance */
  cli_instance_t *cli = NULL;
  /* CLI command handle */
  struct cli_command *command = NULL;
  /* CLI instance type */
  cli_instance_type_t type;

  /* Register command on all CLI instances */
  for (type = CLI_INSTANCE_TYPE_MIN; type <=
      CLI_INSTANCE_TYPE_MAX; ++type) {
    /* Get CLI instance */
    rc = cli_instance_get(&cli, type);
    if (atlk_error(rc)) {
      fprintf(stderr, "cli_instance_get failed: %d\n", rc);
      continue;
    }

    /* Register example command */
    command = cli_register_command(cli, NULL, "example", cmd_example,
                                   PRIVILEGE_UNPRIVILEGED, MODE_ANY,
                                   "Example command");
    if (command == NULL) {
      fprintf(stderr, "cli_register_command returned NULL\n");
    }
  }

  return;
}
```

8.8 craton-threadx/crypto/aes-example.c

```
/* Copyright (C) 2014-2015 Autotalks Ltd. */
#include <stdio.h>
#include <string.h>
#include <inttypes.h>

#include <atlk/sdk.h>
#include <atlk/aes.h>

#if defined __CRATON_NO_ARC || defined __CRATON_ARC1

/*
  CRATON ThreadX AES Example

  This example demonstrates the usage of the AES-ECB, AES-CBC
  encryption/decryption and AES-CMAC generation APIs for code
  running on top of CRATON processor with ThreadX RTOS.
*/

/* Helper function for converting buffer to hex */
static void
buffer_to_line(const void *buf, size_t buf_len, char *line)
{
  const uint8_t *ptr = buf;
  char *pos = &line[0];
  size_t i;

  if (!buf_len) {
    *pos = '\0';
    return;
  }

  if (buf_len > 16) {
    buf_len = 16;
  }

  for (i = 0; i < buf_len - 1; i++) {
    pos += sprintf(pos, "%02x ", ptr[i]);
  }
  pos += sprintf(pos, "%02x", ptr[i]);
}

/* Print buffer to standard output */
static void
buffer_print(const void *buf, size_t len)
{
  const uint8_t *ptr = buf;
  size_t i, line_len, remaining = len;
  char line[80];

  for (i = 0; i < len; i += 16) {
    line_len = remaining < 16 ? remaining : 16;
    remaining -= 16;

    buffer_to_line(ptr + i, line_len, line);
    printf("  %.8lx: %s\n", (unsigned long)i, line);
  }
}

/*
 * AES-CBC example test vectors were taken from:
 *   NIST Special Publication 800-38A:
 *   Recommendation for Block Cipher Modes of Operation:
 *   Methods and Techniques,
 *   Appendix F.2
 */

/* Example AES key used for AES-CBC encryption/decryption */
static const aes_key_t aes_cbc_key = {
  { 0x2b, 0x7e, 0x15, 0x16, 0x28, 0xae, 0xd2, 0xa6,
    0xab, 0xf7, 0x15, 0x88, 0x09, 0xcf, 0x4f, 0x3c }
};

/* Example initialization vector used for AES-CBC encryption/decryption */
static const aes_cbc_iv_t aes_cbc_iv = {
  { 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
    0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f }
};

/* Example plaintext used for AES-CBC encryption */
static const uint8_t aes_cbc_plaintext[] = {
  0x6b, 0xc1, 0xbe, 0xe2, 0x2e, 0x40, 0x9f, 0x96,
  0xe9, 0x3d, 0x7e, 0x11, 0x73, 0x93, 0x17, 0x2a,
  0xae, 0x2d, 0x8a, 0x57, 0x1e, 0x03, 0xac, 0x9c,
  0x9e, 0xb7, 0x6f, 0xac, 0x45, 0xaf, 0x8e, 0x51,
  0x30, 0xc8, 0x1c, 0x46, 0xa3, 0x5c, 0xe4, 0x11,
  0xe5, 0xfb, 0xc1, 0x19, 0x1a, 0x0a, 0x52, 0xef,
  0xf6, 0x9f, 0x24, 0x45, 0xdf, 0x4f, 0x9b, 0x17,
  0xad, 0x2b, 0x41, 0x7b, 0xe6, 0x6c, 0x37, 0x10
};

/* Example ciphertext used for AES-CBC decryption */
static const uint8_t aes_cbc_ciphertext[] = {
  0x76, 0x49, 0xab, 0xac, 0x81, 0x19, 0xb2, 0x46,
  0xce, 0xe9, 0x8e, 0x9b, 0x12, 0xe9, 0x19, 0x7d,
  0x50, 0x86, 0xcb, 0x9b, 0x50, 0x72, 0x19, 0xee,
  0x95, 0xdb, 0x11, 0x3a, 0x91, 0x76, 0x78, 0xb2,
  0x73, 0xbe, 0xd6, 0xb8, 0xe3, 0xc1, 0x74, 0x3b,
  0x71, 0x16, 0xe6, 0x9e, 0x22, 0x22, 0x95, 0x16,
  0x3f, 0xf1, 0xca, 0xa1, 0x68, 0x1f, 0xac, 0x09,
  0x12, 0x0e, 0xca, 0x30, 0x75, 0x86, 0xe1, 0xa7
};

static void
aes_cbc_encrypt_example(void)
{
  /* Buffer for storing ciphertext */
  uint8_t ciphertext[sizeof(aes_cbc_plaintext)];
  /* Size of ciphertext */
  size_t ciphertext_size = sizeof(ciphertext);
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;

  printf("AES-CBC encryption example\n");

  printf("Encryption key:\n");
  buffer_print(&aes_cbc_key, sizeof(aes_cbc_key));

  printf("Initialization vector:\n");
  buffer_print(&aes_cbc_iv, sizeof(aes_cbc_iv));

  printf("Plaintext:\n");
  buffer_print(aes_cbc_plaintext, sizeof(aes_cbc_plaintext));

  /* Encrypt plaintext with AES-CBC */
  rc = aes_cbc_encrypt(&aes_cbc_key,
                       &aes_cbc_iv,
                       aes_cbc_plaintext, sizeof(aes_cbc_plaintext),
                       ciphertext, &ciphertext_size);
  if (atlk_error(rc)) {
    fprintf(stderr, "aes_cbc_encrypt: %s\n", atlk_rc_to_str(rc));
    return;
  }

  printf("Ciphertext:\n");
  buffer_print(ciphertext, ciphertext_size);

  /* Make sure ciphertext is correct */
  if (memcmp(aes_cbc_ciphertext, ciphertext, ciphertext_size) == 0) {
    printf("AES-CBC encryption succeeded\n");
  }
  else {
    printf("AES-CBC encryption failed\n");
  }
  printf("\n");
}

static void
aes_cbc_decrypt_example(void)
{
  /* Buffer for storing plaintext */
  uint8_t plaintext[sizeof(aes_cbc_ciphertext)];
  /* Size of plaintext */
  size_t plaintext_size = sizeof(plaintext);
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;

  printf("AES-CBC decryption example\n");

  printf("Decryption key:\n");
  buffer_print(&aes_cbc_key, sizeof(aes_cbc_key));

  printf("Initialization vector:\n");
  buffer_print(&aes_cbc_iv, sizeof(aes_cbc_iv));

  printf("Ciphertext:\n");
  buffer_print(aes_cbc_ciphertext, sizeof(aes_cbc_ciphertext));

  /* Decrypt ciphertext with AES-CBC */
  rc = aes_cbc_decrypt(&aes_cbc_key,
                       &aes_cbc_iv,
                       aes_cbc_ciphertext, sizeof(aes_cbc_ciphertext),
                       plaintext, &plaintext_size);
  if (atlk_error(rc)) {
    fprintf(stderr, "aes_cbc_decrypt: %s\n", atlk_rc_to_str(rc));
    return;
  }

  printf("Plaintext:\n");
  buffer_print(plaintext, plaintext_size);

  /* Make sure plaintext is correct */
  if (memcmp(aes_cbc_plaintext, plaintext, plaintext_size) == 0) {
    printf("AES-CBC decryption succeeded\n");
  }
  else {
    printf("AES-CBC decryption failed\n");
  }
  printf("\n");
}

/*
 * AES-CMAC example test vectors were taken from:
 *   NIST Special Publication 800-38B:
 *   Recommendation for Block Cipher Modes of Operation:
 *   The CMAC Mode for Authentication,
 *   Appendix D.1
 */

/* Example AES key used for AES-CMAC tag generation */
static const aes_key_t aes_cmac_key = {
  { 0x2b, 0x7e, 0x15 ,0x16, 0x28, 0xae, 0xd2, 0xa6,
    0xab, 0xf7, 0x15, 0x88, 0x09, 0xcf, 0x4f, 0x3c }
};

/* Example message for AES-CMAC tag generation */
static const uint8_t aes_cmac_msg[] = {
  0x6b, 0xc1, 0xbe, 0xe2, 0x2e, 0x40, 0x9f, 0x96,
  0xe9, 0x3d, 0x7e, 0x11, 0x73, 0x93, 0x17, 0x2a
};

/* Expected AES-CMAC tag */
static const aes_cmac_tag_t aes_cmac_tag = {
  { 0x07, 0x0a, 0x16, 0xb4, 0x6b, 0x4d, 0x41, 0x44,
    0xf7, 0x9b, 0xdd, 0x9d, 0xd0, 0x4a, 0x28, 0x7c }
};

static void
aes_cmac_example(void)
{
  /* AES-CMAC tag */
  aes_cmac_tag_t tag = AES_CMAC_TAG_INIT;
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;

  printf("AES-CMAC example:\n");
  printf("Key:\n");
  buffer_print(&aes_cmac_key, sizeof(aes_cmac_key));

  printf("Message:\n");
  buffer_print(aes_cmac_msg, sizeof(aes_cmac_msg));

  /* Compute AES-CMAC tag */
  rc = aes_cmac_compute(&aes_cmac_key, aes_cmac_msg, sizeof(aes_cmac_msg), &tag);
  if (atlk_error(rc)) {
    fprintf(stderr, "aes_cmac_compute: %s\n", atlk_rc_to_str(rc));
    return;
  }

  printf("AES-CMAC Tag:\n");
  buffer_print(&tag, sizeof(tag));

  /* Make sure tag is correct */
  if (memcmp(&aes_cmac_tag, &tag, sizeof(tag)) == 0) {
    printf("AES-CMAC generation succeeded\n");
  }
  else {
    printf("AES-CBC generation failed\n");
  }
  printf("\n");
}

/*
 * The AES-ECB Monte Carlo Test (MCT) example demonstrates the usage
 * of the AES-ECB encryption API to implement part of the MCT
 * as described in [1].
 * The test vectors were taken from [2] and [3].
 *
 * [1] http://csrc.nist.gov/groups/STM/cavp/documents/aes/AESAVS.pdf
 * [2] http://csrc.nist.gov/groups/STM/cavp/documents/aes/aesmct_intermediate.zip
 * [3] http://csrc.nist.gov/groups/STM/cavp/documents/aes/aesmct.zip
 */

/*
 * File: aesmct/ECBMCT128.rsp or aesmct_intermediate/ECBMCT128.txt.
 * Count: 37.
 */

static const aes_key_t aes_ecb_mct_key = {
  { 0x9d, 0xac, 0x1c, 0x1d, 0x31, 0x3d, 0xd0, 0x09,
    0x3e, 0xbb, 0x02, 0x89, 0xd5, 0x74, 0xb4, 0x76
  }
};

static const uint8_t aes_ecb_mct_plaintext[] = {
  0x59, 0x59, 0xeb, 0xd7, 0xa1, 0x16, 0x77, 0x13,
  0x42, 0x9e, 0xda, 0x69, 0x53, 0x8c, 0x53, 0x6b
};

static const uint8_t aes_ecb_mct_ciphertext[] = {
  0xf5, 0x71, 0x01, 0xd7, 0xfa, 0x19, 0xf9, 0x7a,
  0x31, 0xd6, 0x0b, 0x27, 0x63, 0x12, 0x71, 0x7c
};

#define AES_ECB_MCT_ITER_NUM 1000

static void
aes_ecb_mct_example(void)
{
  /* Intermediate plaintext/ciphertext */
  uint8_t text[sizeof(aes_ecb_mct_plaintext)] = { 0 };
  /* Text size */
  size_t size;
  /* Iteration variable */
  int i;
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;

  printf("AES-ECB MCT example:\n");

  printf("Key:\n");
  buffer_print(&aes_ecb_mct_key, sizeof(aes_ecb_mct_key));

  printf("Plaintext:\n");
  buffer_print(aes_ecb_mct_plaintext, sizeof(aes_ecb_mct_plaintext));

  printf("Ciphertext:\n");
  buffer_print(aes_ecb_mct_ciphertext, sizeof(aes_ecb_mct_ciphertext));

  /* Initialization */
  memcpy(text, aes_ecb_mct_plaintext, sizeof(text));
  size = sizeof(text);
  /* Run the MCT iterations */
  for (i = 0; i < AES_ECB_MCT_ITER_NUM; i++) {
    rc = aes_ecb_encrypt(&aes_ecb_mct_key, text, size, text, &size);
    if (atlk_error(rc)) {
      fprintf(stderr, "aes_ecb_encrypt: %s\n", atlk_rc_to_str(rc));
      return;
    }
  }

  /* Make sure the cipher text is correct */
  if (memcmp(text, aes_ecb_mct_ciphertext, sizeof(text)) == 0) {
    printf("AES-ECB MCT succeeded\n");
  }
  else {
    printf("AES-ECB MCT failed\n");
  }
  printf("\n");
}

void
craton_user_init(void)
{
  /* AES-CBC encryption example */
  aes_cbc_encrypt_example();

  /* AES-CBC decryption example */
  aes_cbc_decrypt_example();

  /* AES-CMAC example */
  aes_cmac_example();

  /* AES-ECB MCT example */
  aes_ecb_mct_example();
}

#else /* __CRATON_NO_ARC || __CRATON_ARC1 */

void craton_user_init(void)
{
}

#endif /* __CRATON_NO_ARC || __CRATON_ARC1 */
```

8.9 craton-threadx/crypto/ecdsa-benchmark.c

```
/* Copyright (C) 2015 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>

#include <tx_api.h>

#include <atlk/ecc_service.h>

#if defined __CRATON_NO_ARC || defined __CRATON_ARC1

/*
  CRATON ThreadX ECDSA Benchmark

  This program benchmarks CRATON ECDSA verification API with some of the
  elliptic curves supported by the API. The same verification request is
  used over and over for each curve. All verification are intended to result
  in success. This is checked using assert().

  To take advantage of CRATON HW parallelism the average number of
  ongoing requests (i.e. started but not completed) should be more than 1.
  Having more than 3 ongoing requests will not result in any throughput gains
  but is supported up to an implementation-defined upper bound. For reference,
  in SDK 4.5 this upper bound is 256.
*/

/* Total number of requests per benchmark */
#define NUM_REQUESTS 10000

/*
  Number of incomplete requests at any point in time.
  CRATON ECDSA HW will provide less than maximum throughput
  if this number is less than 3.
*/
#define NUM_ONGOING 3

/* Priority of benchmark thread */
#define ECDSA_BENCHMARK_THREAD_PRIORITY 40

/* Benchmark descriptor */
struct benchmark {
  const char *curve_name;
  ecc_request_t request;
};

/* Array of benchmark descriptors (one per curve) */
static const struct benchmark benchmarks[] = {
  {
    .curve_name = "NIST P256",
    .request =  {
      .context = {
        .request_id = 0,
        .request_type = ECC_REQUEST_TYPE_VERIFY,
        .curve = ECC_CURVE_NIST_P256
      },
      .params.verify_params = {
        .public_key = {
          .point_type = ECC_POINT_UNCOMPRESSED,
          .x_coordinate = {
            .value = {
              0xbc3fdd5d, 0x620d0a14, 0x5d867d8b, 0x286867ec,
              0x92c47d90, 0x8a772d43, 0x44eb3895, 0x26f3751e
            }
          },
          .y_coordinate = {
            .value = {
              0x96fc56f1, 0xf79baeaa, 0xff5b3542, 0xb7ffb678,
              0xc22d9ddb, 0x3dc0cb4d, 0xf0e24af5, 0x1606db3b
            }
          }
        },
        .digest = {
          .value = {
            0xb9, 0x3d, 0x12, 0xb2, 0xc6, 0x02, 0x7b, 0x0b,
            0xa4, 0xd4, 0xd8, 0xc2, 0xbc, 0x20, 0xda, 0x88,
            0x8b, 0xe2, 0x42, 0x2f, 0x08, 0x9b, 0xe3, 0x24,
            0x3a, 0x6c, 0x44, 0xe5, 0x0d, 0xde, 0xf0, 0xcb
          },
          .value_size = 32
        },
        .signature = {
          .r_scalar = {
            .value = {
              0x4e3a775c, 0x71a5c259, 0xfad57a8e, 0xd1e45591,
              0x030fbb65, 0x94d2300b, 0x7ceccd7d, 0xbc70ad36
            }
          },
          .s_scalar = {
            .value = {
              0xbc05d39c, 0xd2c5f32b, 0xf10502c6, 0xb91de10c,
              0x8599d089, 0x0873e8ae, 0x7b137225, 0xd51dd454
            }
          }
        }
      }
    }
  },
  {
    .curve_name = "Brainpool P256t1",
    .request = {
      .context = {
        .request_id = 0,
        .request_type = ECC_REQUEST_TYPE_VERIFY,
        .curve = ECC_CURVE_BRAINPOOL_P256t1
      },
      .params.verify_params = {
        .public_key = {
          .point_type = ECC_POINT_UNCOMPRESSED,
          .x_coordinate = {
            .value = {
              0xfc7fc794, 0x3a85ed28, 0xc3ebcaaf, 0x2a326938,
              0xef9ed9c7, 0x779df5c2, 0x6e220a95, 0x6344dff5
            }
          },
          .y_coordinate = {
            .value = {
              0x365babff, 0xf6cfc69f, 0xc2a9f394, 0x94cd22bf,
              0x46cbf110, 0x273452ae, 0xf55a41f3, 0x2e2e94a8
            }
          }
        },
        .digest = {
          .value = {
            0x8e, 0x89, 0x03, 0x45, 0x87, 0x5b, 0xef, 0x0b,
            0xaa, 0xa0, 0xe0, 0x98, 0xbf, 0xf2, 0x78, 0xdd,
            0xbf, 0x00, 0xee, 0x06, 0xcc, 0x08, 0x07, 0xa9,
            0xd8, 0xf6, 0x4c, 0x93, 0x29, 0xb0, 0xd2, 0x2d
          },
          .value_size = 32
        },
        .signature = {
          .r_scalar = {
            .value = {
              0x15a73647, 0xb0ed3efa, 0x6f44c325, 0x7607b1a5,
              0xa06cf2a1, 0xc5f298a9, 0x13c2c3bc, 0x9168331f
            }
          },
          .s_scalar = {
            .value = {
              0xec7d28a2, 0x396dbb17, 0xbfc33ae6, 0xf0832dd6,
              0x2adf90bb, 0x4b422130, 0x46ad044f, 0x353f89ca
            }
          }
        }
      }
    }
  }
};

static ecc_service_t *service = NULL;
static ecc_socket_t *socket = NULL;
static TX_THREAD ecdsa_benchmark_thread;
static uint8_t ecdsa_benchmark_thread_stack[1 << 12];

static void
run_benchmark(const struct benchmark *benchmark)
{
  atlk_rc_t rc;
  ecc_response_t response;

  /* Start benchmark */
  printf("Benchmarking ECDSA verification with curve \"%s\"...\n",
     benchmark->curve_name);
  uint32_t start_time = tx_time_get();

  /* Start a few requests to take advantage of HW parallelism */
  for (int i = 0; i < NUM_ONGOING; i++) {
    rc = ecc_request_send(socket, &benchmark->request, NULL);
    assert(!atlk_error(rc));
  }

  /* Start a new request whenever an ongoing request completes */
  for (int i = 0; i < NUM_REQUESTS - NUM_ONGOING; i++) {
    rc = ecc_response_receive(socket, &response, &
      atlk_wait_forever);
    assert(!atlk_error(rc));
    assert(response.rc == ECC_OK);

    rc = ecc_request_send(socket, &benchmark->request, NULL);
    assert(!atlk_error(rc));
  }

  /* Wait for all ongoing requests to complete */
  for (int i = 0; i < NUM_ONGOING; i++) {
    rc = ecc_response_receive(socket, &response, &
      atlk_wait_forever);
    assert(!atlk_error(rc));
    assert(response.rc == ECC_OK);
  }

  /* Finish benchmark */
  int32_t elapsed_time = tx_time_get() - start_time;
  printf("ECDSA verification throughput with curve \"%s\" is %.1f Hz\n",
     benchmark->curve_name,
     (float)NUM_REQUESTS / ((float)elapsed_time / TX_TICK_RATE));
}

void ecdsa_benchmark_thread_entry(ULONG input)
{
  (void)input;

  printf("*** Start of ECDSA benchmark suite ***\n");

  for (size_t i = 0; i < sizeof(benchmarks) / sizeof(benchmarks[0]); i++) {
    run_benchmark(&benchmarks[i]);
  }

  printf("*** End of ECDSA benchmark suite ***\n");
}

void craton_user_init(void)
{
  UINT trv;
  atlk_rc_t rc;
  rc = ecc_default_service_get(&service);
  assert(!atlk_error(rc));

  rc = ecc_socket_create(service, &socket);
  assert(!atlk_error(rc));

  trv = tx_thread_create(&ecdsa_benchmark_thread,
                         "ecdsa_benchmark_thread",
                         ecdsa_benchmark_thread_entry, 0,
                         ecdsa_benchmark_thread_stack,
                         sizeof(ecdsa_benchmark_thread_stack),
                         ECDSA_BENCHMARK_THREAD_PRIORITY,
                         ECDSA_BENCHMARK_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);
}

#else /* __CRATON_NO_ARC || __CRATON_ARC1 */

void craton_user_init(void)
{
}

#endif /* __CRATON_NO_ARC || __CRATON_ARC1 */
```

8.10 craton-threadx/crypto/ecdsa-example.c

```
/* Copyright (C) 2014-2015 Autotalks Ltd. */
#include <stdio.h>
#include <string.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>

#include <tx_api.h>

#include <atlk/sdk.h>
#include <atlk/sha.h>
#include <atlk/ecc.h>
#include <atlk/ecdsa.h>
#include <atlk/ecc_service.h>
#include <atlk/hsm_service.h>
#include <atlk/hsm_emulator.h>

#include <craton/sha_hw.h>

#if defined __CRATON_NO_ARC || defined __CRATON_ARC1

/*
  CRATON ThreadX ECDSA Example

  This example demonstrates a basic ECDSA signing/verification scenario using
  the HSM API, ECC API and CRATON SHA API for code running on top of CRATON
  processor with ThreadX RTOS.

  The device used in this example is a "HSM emulator", a term used to describe
  an emulated HSM device. The differences between an emulated HSM service
  and a real one are:
  - The emulated HSM service is created via hsm_emulator_create().
  - The implementation is not tamper-resistant because it uses general purpose
    hardware instead of tamper-resistant hardware.

  The purpose of the emulated HSM is basic software integration with
  the HSM API on a hardware platform that doesn't have a working HSM chip.
*/

/* HSM emulator service */
static hsm_service_t *example_hsm_service = NULL;

/* ECC service */
static ecc_service_t *example_ecc_service = NULL;

/* ECC socket */
static ecc_socket_t *example_ecc_socket = NULL;

/* Number of NVM cells to configure for HSM */
#define ECDSA_EXAMPLE_HSM_NVM_NUM_CELLS 128

/* HSM emulator filename */
#define HSM_EMULATOR_FILENAME "B:/hsm-emu.dat"

/* ECDSA example message maximum data size in octets */
#define ECDSA_EXAMPLE_MSG_MAX_DATA_SIZE 64

/* ECDSA example message */
typedef struct {
  /* Data (octet string) */
  uint8_t data[ECDSA_EXAMPLE_MSG_MAX_DATA_SIZE];

  /* Data size in octets */
  size_t data_size;

  /* ECC elliptic curve */
  ecc_curve_t curve;

  /* ECC public key */
  ecc_point_t public_key;

  /* ECDSA fast verification signature */
  ecc_fast_verification_signature_t signature;

} ecdsa_example_message_t;

/* Format string for ECC scalar */
#define ECC_SCALAR_FMT \
  "0x%08lx,0x%08lx,0x%08lx,0x%08lx,0x%08lx,0x%08lx,0x%08lx,0x%08lx," \
  "0x%08lx,0x%08lx,0x%08lx,0x%08lx"

/* Format argument list for ecc_scalar_t */
#define ECC_SCALAR_FMT_ARGS(x)                        \
  x.value[0], x.value[1], x.value[2], x.value[3], \
  x.value[4], x.value[5], x.value[6], x.value[7], \
  x.value[8], x.value[9], x.value[10], x.value[11]

/* Format string for SHA digest */
#define SHA_256_DIGEST_FMT \
  "%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x" \
  "%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x"

/* Format argument list for SHA digest */
#define SHA_256_DIGEST_FMT_ARGS(x)                          \
  x.value[0], x.value[1], x.value[2], x.value[3],       \
  x.value[4], x.value[5], x.value[6], x.value[7],       \
  x.value[8], x.value[9], x.value[10], x.value[11],     \
  x.value[12], x.value[13], x.value[14], x.value[15],   \
  x.value[16], x.value[17], x.value[18], x.value[19],   \
  x.value[20], x.value[21], x.value[22], x.value[23],   \
  x.value[24], x.value[25], x.value[26], x.value[27],   \
  x.value[28], x.value[29], x.value[30], x.value[31]

static atlk_rc_t
ecdsa_example_alice(ecdsa_example_message_t *msg)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* HSM secure storage cell index */
  hsm_cell_index_t cell_index;
  /* Private key information */
  hsm_ecc_private_key_info_t private_key_info =
      HSM_ECC_PRIVATE_KEY_INFO_INIT;
  /* ECC point representing a public key */
  ecc_point_t public_key = ECC_POINT_INIT;
  /* SHA digest */
  sha_digest_t digest = SHA_DIGEST_INIT;
  /* ECDSA fast verification signature */
  ecc_fast_verification_signature_t signature =
    ECC_FAST_VERIFICATION_SIGNATURE_INIT;
  /* Example message */
  static const char example_msg[] =
    "Autotalks - The Confidence of Knowing Ahead";

  printf("\n>>> Alice\n");

  /* Print the message data */
  printf("Message data: %s\n", example_msg);
  printf("Message data size: %lu\n", (long unsigned int)sizeof(example_msg));

  /* Arbitrarily chosen HSM cell index for the sake of this example */
  cell_index = 6;
  printf("Using HSM cell index: %lu\n", cell_index);

  /* Using NIST P-256 elliptic curve and an Isolated key */
  private_key_info.key_curve = ECC_CURVE_NIST_P256;
  private_key_info.key_type = HSM_PRIVATE_KEY_TYPE_ISOLATED;

  printf("Using elliptic curve ID: %u\n", private_key_info.key_curve);
  printf("Using key type ID: %u\n", private_key_info.key_type);

  /* Create private key and store it in the chosen cell */
  rc = hsm_ecc_private_key_create(example_hsm_service,
                                  cell_index,
                                  &private_key_info);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_ecc_private_key_create: %s\n", atlk_rc_to_str(rc));
    return rc;
  }

  printf("ECC private key created\n");

  /* Retrieve public key for this cell's private key */
  rc = hsm_ecc_public_key_get(example_hsm_service, cell_index, &public_key);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_ecc_public_key_get: %s\n", atlk_rc_to_str(rc));
    return rc;
  }

  assert(public_key.point_type == ECC_POINT_UNCOMPRESSED);

  /* Print retrieved ECC public key */
  printf("ECC public key created:\n");
  printf("  x: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(public_key.x_coordinate));
  printf("  y: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(public_key.y_coordinate));

  /* Compute SHA-256 digest of example message */
  rc = sha_hw_sha256_compute(example_msg, sizeof(example_msg), &digest);
  if (atlk_error(rc)) {
    fprintf(stderr, "sha_hw_sha256_compute: %s", atlk_rc_to_str(rc));
    return rc;
  }

  /* Print computed SHA-256 digest */
  printf("SHA-256 hash digest computed:\n");
  printf("  Digest: " SHA_256_DIGEST_FMT "\n", SHA_256_DIGEST_FMT_ARGS(digest));

  /* Generate ECDSA fast verification signature */
  rc = hsm_ecdsa_sign(example_hsm_service, cell_index, &digest, &signature);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_ecdsa_sign: %s\n", atlk_rc_to_str(rc));
    return rc;
  }

  assert(signature.R_point.point_type == ECC_POINT_UNCOMPRESSED);

  /* Print generated ECDSA signature */
  printf("ECDSA signature generated:\n");
  printf("  Rx: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(signature.R_point.x_coordinate));
  printf("  Ry: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(signature.R_point.y_coordinate));
  printf("  s: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(signature.s_scalar));

  /* Make sure the example message can fit into the data */
  assert(sizeof(example_msg) <= sizeof(msg->data));

  /* Produce the message */
  msg->data_size = sizeof(example_msg);
  memcpy(msg->data, example_msg, msg->data_size);
  msg->curve = private_key_info.key_curve;
  msg->public_key = public_key;
  msg->signature = signature;

  return ATLK_OK;
}

static atlk_rc_t
ecdsa_example_bob(const ecdsa_example_message_t *msg)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* SHA digest */
  sha_digest_t digest = SHA_DIGEST_INIT;
  /* ECDSA signature */
  ecc_signature_t signature = ECC_SIGNATURE_INIT;
  /* ECC request */
  ecc_request_t request = ECC_REQUEST_INIT;
  /* ECC response */
  ecc_response_t response = ECC_RESPONSE_INIT;
  /* ECC request identifier */
  ecc_request_id_t request_id;

  printf("\n>>> Bob\n");

  /* Print received message */
  printf("Message data: %s\n", msg->data);
  printf("Message data size: %lu\n", (long unsigned int)msg->data_size);
  printf("Using elliptic curve ID: %u\n", msg->curve);
  assert(msg->public_key.point_type == ECC_POINT_UNCOMPRESSED);

  /* Print received ECC public key */
  printf("ECC public key:\n");
  printf("  x: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(msg->public_key.x_coordinate));
  printf("  y: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(msg->public_key.y_coordinate));

  /* Print received ECDSA signature for fast verification */
  printf("ECDSA signature:\n");
  printf("  Rx: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(msg->signature.R_point.x_coordinate));
  printf("  Ry: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(msg->signature.R_point.y_coordinate));
  printf("  s: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(msg->signature.s_scalar));

  /* Compute SHA-256 hash value of received message */
  rc = sha_hw_sha256_compute(msg->data, msg->data_size, &digest);
  if (atlk_error(rc)) {
    fprintf(stderr, "sha_hw_sha256_compute: %s", atlk_rc_to_str(rc));
    return rc;
  }

  /* Print computed SHA-256 digest */
  printf("SHA-256 hash digest computed:\n");
  printf("  Digest: " SHA_256_DIGEST_FMT "\n", SHA_256_DIGEST_FMT_ARGS(digest));

  /* Convert ECDSA signature for fast verification */
  rc = ecdsa_signature_convert(msg->curve, &msg->signature, &signature);
  if (atlk_error(rc)) {
    fprintf(stderr, "ecdsa_signature_convert: %s", atlk_rc_to_str(rc));
    return rc;
  }

  /* Print converted ECDSA signature for fast verification */
  printf("Converted ECDSA signature for fast verification:\n");
  printf("  r: " ECC_SCALAR_FMT "\n", ECC_SCALAR_FMT_ARGS(signature.r_scalar));
  printf("  s: " ECC_SCALAR_FMT "\n", ECC_SCALAR_FMT_ARGS(signature.s_scalar));

  /* Arbitrary request identifier */
  request_id = 10;

  /* Fill ECC request */
  request.context.request_id = request_id;
  request.context.request_type = ECC_REQUEST_TYPE_VERIFY;
  request.context.curve = msg->curve;
  request.params.verify_params.public_key = msg->public_key;
  request.params.verify_params.digest = digest;
  request.params.verify_params.signature = signature;

  /* Send ECC request */
  rc = ecc_request_send(example_ecc_socket, &request, NULL);
  if (atlk_error(rc)) {
    fprintf(stderr,"ecc_request_send: %s\n", atlk_rc_to_str(rc));
    return rc;
  }

  /* Print ECC request ID */
  printf("Sent ECC request with ID %" PRIu32 "\n", request_id);

  /* Receive ECC response */
  rc = ecc_response_receive(example_ecc_socket, &response, &
      atlk_wait_forever);
  if (atlk_error(rc)) {
    fprintf(stderr,"ecc_response_receive: %s\n", atlk_rc_to_str(rc));
    return rc;
  }

  /* Print ECC response */
  printf("ECC response for request ID %" PRIu32 ": %d\n",
    response.context.request_id, response.rc);

  /* Print ECC response verification result */
  if (response.rc == ECC_OK) {
    printf("SUCCESS\n");
  }
  else {
    printf("FAILURE\n");
  }

  return rc;
}

void
craton_user_init(void)
```

```
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* ECDSA example message */
  ecdsa_example_message_t message;
  /* HSM capability information */
  hsm_capability_info_t hsm_capability_info =
      HSM_CAPABILITY_INFO_INIT;
  /* HSM NVM configuration */
  hsm_nvm_config_t hsm_nvm_config = HSM_NVM_CONFIG_INIT;
  /* HSM emulator configuration */
  hsm_emulator_config_t hsm_emulator_config =
      HSM_EMULATOR_CONFIG_INIT;

  /* Initialize the HSM emulator configuration */
  rc = ecc_default_service_get(&example_ecc_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "ecc_default_service_get: %s\n", atlk_rc_to_str(rc));
    goto out;
  }

  hsm_emulator_config.ecc_service_ptr = example_ecc_service;
#ifdef FS_EXIST
  hsm_emulator_config.nvm_file_path = HSM_EMULATOR_FILENAME;
#else
  /* In case of Multi-Core, system will not use nor flash but ram */
  hsm_emulator_config.nvm_file_path = NULL;
#endif /* FS_EXIST*/


  /* Create HSM emulator service */
  rc = hsm_emulator_create(&hsm_emulator_config, &example_hsm_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_emulator_create: %s\n", atlk_rc_to_str(rc));
    goto out;
  }

  /* Get HSM capability information */
  rc = hsm_capability_info_get(example_hsm_service, &hsm_capability_info);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_capability_info_get: %s", atlk_rc_to_str(rc));
    goto out;
  }

  printf("HSM capability information:\n");
  printf("  Maximum number of NVM cells: %lu\n",
    hsm_capability_info.max_num_of_cells);
  printf("  Current number of NVM cells: %lu\n",
    hsm_capability_info.current_num_of_cells);
  printf("  Maximum number of cell ranges supported by "
         "hsm_csr_ecdsa_public_keys_sign(): %lu\n",
    hsm_capability_info.max_num_of_cell_ranges_for_csr);

  printf("Initializing NVM to contain %u cells\n",
    ECDSA_EXAMPLE_HSM_NVM_NUM_CELLS);

  hsm_nvm_config.num_of_cells = ECDSA_EXAMPLE_HSM_NVM_NUM_CELLS;

  /* Initialize HSM NVM */
  rc = hsm_nvm_init(example_hsm_service, &hsm_nvm_config);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_nvm_init: %s", atlk_rc_to_str(rc));
    goto out;
  }

  /* Create ECC socket */
  rc = ecc_socket_create(example_ecc_service, &example_ecc_socket);
  if (atlk_error(rc)) {
    fprintf(stderr, "ecc_socket_create: %s\n", atlk_rc_to_str(rc));
    goto out;
  }

  /* Produce example message by Alice */
  rc = ecdsa_example_alice(&message);
  if (atlk_error(rc)) {
    goto out;
  }

  /* Consume example message by Bob */
  rc = ecdsa_example_bob(&message);
  if (atlk_error(rc)) {
    goto out;
  }

out:
  if (atlk_error(rc)) {
    fprintf(stderr, "ERROR\n");
  }

  /* Delete ECC socket */
  ecc_socket_delete(example_ecc_socket);

  /* Delete ECC service */
  ecc_service_delete(example_ecc_service);

  /* Delete HSM emulator service */
  hsm_service_delete(example_hsm_service);
}

#else /* __CRATON_NO_ARC || __CRATON_ARC1 */

void craton_user_init(void)
{
}

#endif /* __CRATON_NO_ARC || __CRATON_ARC1 */
```

8.11 craton-threadx/crypto/ecies-example.c

```
/* Copyright (C) 2014-2016 Autotalks Ltd. */
#include <stdio.h>
#include <string.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>

#include <tx_api.h>

#include <atlk/sdk.h>
#include <atlk/rng.h>
#include <atlk/sha.h>
#include <atlk/ecc.h>
#include <atlk/ecies.h>
#include <atlk/aes.h>
#include <atlk/hsm_service.h>
#include <atlk/hsm_emulator.h>

#if defined __CRATON_NO_ARC || defined __CRATON_ARC1

/*
  CRATON ThreadX ECIES Example

  This example demonstrates a basic ECIES and AES-CCM encryption/decryption
  scenario inspired by IEEE Std. 1609.2-2016 using the HSM API, ECC API and
  RNG API for code running on top of CRATON processor with ThreadX RTOS.

  The device used in this example is a "HSM emulator", a term used to describe
  an emulated HSM device. The differences between an emulated HSM service
  and a real one are:
  - The emulated HSM service is created via hsm_emulator_create().
  - The implementation is not tamper-resistant because it uses general purpose
    hardware instead of tamper-resistant hardware.

  The purpose of the emulated HSM is basic software integration with
  the HSM API on a hardware platform that doesn't have a working HSM chip.
*/

/* HSM emulator service */
static hsm_service_t *example_hsm_service = NULL;

/* ECC service */
static ecc_service_t *example_ecc_service = NULL;

/* Format string for ECC scalar */
#define ECC_SCALAR_FMT \
  "0x%08lx,0x%08lx,0x%08lx,0x%08lx,0x%08lx,0x%08lx,0x%08lx,0x%08lx," \
  "0x%08lx,0x%08lx,0x%08lx,0x%08lx"

/* Format argument list for ecc_scalar_t */
#define ECC_SCALAR_FMT_ARGS(x)                              \
  x.value[0], x.value[1], x.value[2], x.value[3],    \
  x.value[4], x.value[5], x.value[6], x.value[7],    \
  x.value[8], x.value[9], x.value[10], x.value[11]

/* ECIES HMAC key size in octets */
#define ECIES_HMAC_KEY_SIZE 32

/* Number of NVM cells to configure for HSM */
#define ECIES_EXAMPLE_HSM_NVM_NUM_CELLS 128

/* HSM emulator filename */
#define HSM_EMULATOR_FILENAME "B:/hsm-emu.dat"
```

```
/* ECIES example shared infomation between Alice and Bob */
typedef struct {
  /* Elliptic curve used */
  ecc_curve_t curve;

  /* Bob ECC public key */
  ecc_point_t bob_public_key;

} ecies_example_shared_info_t;

static ecies_example_shared_info_t ecies_example_shared_info = {
  .curve = ECC_CURVE_NIST_P224,
  .bob_public_key = ECC_POINT_INIT
};

/* ECIES example Bob's private information */
typedef struct {
  /* Index of the ECC private key */
  hsm_cell_index_t private_key_index;

} ecies_example_bob_info_t;

static ecies_example_bob_info_t ecies_example_bob_info = {
  .private_key_index = HSM_CELL_INDEX_NA
};

/* ECIES example maximum message size in octets */
#define ECIES_EXAMPLE_MESSAGE_MAX_SIZE 64

/* ECIES example message */
typedef struct {
  /* Ephemeral public key used for ECIES encryption */
  ecc_point_t ecies_ephemeral_public_key;

  /* AES-CCM key encrypted using ECIES */
  uint8_t ecies_encrypted_aes_key[AES_KEY_SIZE];

  /* ECIES authentication tag */
  ecies_authentication_tag_t ecies_authentication_tag;

  /* Ciphertext encrypted using AES-CCM */
  uint8_t aes_ccm_ciphertext[ECIES_EXAMPLE_MESSAGE_MAX_SIZE];

  /* Ciphertext size in octets */
  size_t aes_ccm_ciphertext_size;

  /* AES-CCM nonce */
  aes_ccm_nonce_t aes_ccm_nonce;

  /* AES-CCM authentication tag */
  aes_ccm_authentication_tag_t aes_ccm_tag;

} ecies_example_message_t;

static void
ecies_example_print_buffer(const uint8_t *buf, size_t buf_len)
{
  size_t i;

  for (i = 0; i < buf_len; i++) {
    printf("%02x", buf[i]);
  }
}

static atlk_rc_t
ecies_example_init(void)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* Private key information */
  hsm_ecc_private_key_info_t private_key_info =
      HSM_ECC_PRIVATE_KEY_INFO_INIT;
  /* HSM cell index to store Bob's private key */
  hsm_cell_index_t private_key_index = HSM_CELL_INDEX_NA;
  /* Bob's ECC public key */
  ecc_point_t bob_public_key = ECC_POINT_INIT;
  /* HSM capability information */
  hsm_capability_info_t hsm_capability_info =
      HSM_CAPABILITY_INFO_INIT;
  /* HSM NVM configuration */
  hsm_nvm_config_t hsm_nvm_config = HSM_NVM_CONFIG_INIT;

  printf("\n>>> Initialization\n");

  /* Get HSM capability information */
  rc = hsm_capability_info_get(example_hsm_service, &hsm_capability_info);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_capability_info_get: %s", atlk_rc_to_str(rc));
    return rc;
  }

  printf("HSM capability information:\n");
  printf("  Maximum number of NVM cells: %lu\n",
    hsm_capability_info.max_num_of_cells);
  printf("  Current number of NVM cells: %lu\n",
    hsm_capability_info.current_num_of_cells);
  printf("  Maximum number of cell ranges supported by "
         "hsm_csr_ecdsa_public_keys_sign(): %lu\n",
    hsm_capability_info.max_num_of_cell_ranges_for_csr);

  printf("Initializing NVM to contain %u cells\n",
    ECIES_EXAMPLE_HSM_NVM_NUM_CELLS);

  hsm_nvm_config.num_of_cells = ECIES_EXAMPLE_HSM_NVM_NUM_CELLS;

  /* Initialize HSM NVM */
  rc = hsm_nvm_init(example_hsm_service, &hsm_nvm_config);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_nvm_init: %s", atlk_rc_to_str(rc));
    return rc;
  }

  /* Use NIST P-256 elliptic curve and an Isolated key for ECIES algorithm */
  private_key_info.key_curve = ECC_CURVE_NIST_P256;
  private_key_info.key_type = HSM_PRIVATE_KEY_TYPE_ISOLATED;
  private_key_info.key_algorithm = HSM_PUBLIC_KEY_ALGORITHM_ECIES
     ;

  printf("Using elliptic curve ID: %u\n", private_key_info.key_curve);
  printf("Using key_type ID: %u\n", private_key_info.key_type);
  printf("Using key_algorithm ID: %u\n", private_key_info.key_algorithm);

  /* Use the first available cell to store Bob's private key */
  private_key_index = 0;

  printf("Using HSM cell index: %lu\n", private_key_index);

  /* Create Bob's private key */
  rc = hsm_ecc_private_key_create(example_hsm_service,
                                  private_key_index,
                                  &private_key_info);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_ecc_private_key_create: %s", atlk_rc_to_str(rc));
    return rc;
  }

  printf("Bob's ECC private key created\n");

  /* Get Bob's public key */
  rc = hsm_ecc_public_key_get(example_hsm_service,
                              private_key_index,
                              &bob_public_key);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_ecc_public_key_get: %s", atlk_rc_to_str(rc));
    return rc;
  }

  assert(bob_public_key.point_type == ECC_POINT_UNCOMPRESSED);

  /* Print retrieved ECC public key */
  printf("Bob's ECC public key created:\n");
  printf("  x: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(bob_public_key.x_coordinate));
  printf("  y: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(bob_public_key.y_coordinate));

  /* Store shared information */
  ecies_example_shared_info.curve = private_key_info.key_curve;
  ecies_example_shared_info.bob_public_key = bob_public_key;

  /* Store Bob's private information */
  ecies_example_bob_info.private_key_index = private_key_index;

  return ATLK_OK;
}

static atlk_rc_t
ecies_example_alice(ecies_example_message_t *msg)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* AES key */
  aes_key_t aes_key = AES_KEY_INIT;
  /* AES-CCM nonce */
```

```
aes_ccm_nonce_t aes_ccm_nonce = AES_CCM_NONCE_INIT;
/* AES-CCM authentication tag */
aes_ccm_authentication_tag_t aes_ccm_tag =
    AES_CCM_AUTHENTICATION_TAG_INIT;
/* AES-CCM ciphertext */
uint8_t ciphertext[ECIES_EXAMPLE_MESSAGE_MAX_SIZE] = { 0 };
/* AES-CCM ciphertext size in octets */
size_t ciphertext_size = sizeof(ciphertext);
/* Example message */
static const char example_msg[] =
  "Autotalks - The Confidence of Knowing Ahead";
/* ECIES key */
uint8_t ecies_key[sizeof(aes_key) + ECIES_HMAC_KEY_SIZE] = { 0 };
/* ECIES ephemeral public key */
ecc_point_t ephemeral_public_key = ECC_POINT_INIT;
/* AES key encrypted using ECIES */
uint8_t encrypted_aes_key[AES_KEY_SIZE] = { 0 };
/* AES key size in octets */
size_t encrypted_aes_key_size = sizeof(encrypted_aes_key);
/* ECIES authentication tag */
ecies_authentication_tag_t ecies_authentication_tag =
  ECIES_AUTHENTICATION_TAG_INIT;

printf("\n>>> Alice\n");

/* Print the message data */
printf("Message: %s\n", example_msg);
printf("Message size: %lu\n", (long unsigned int)sizeof(example_msg));

/* Get random AES key */
rc = rng_data_get(&aes_key, sizeof(aes_key));
if (atlk_error(rc)) {
  fprintf(stderr, "rng_data_get: %s", atlk_rc_to_str(rc));
  return rc;
}

/* Print random AES key */
printf("AES key: ");
ecies_example_print_buffer(aes_key.value, sizeof(aes_key));
printf("\n");

/* Get random AES-CCM nonce */
rc = rng_data_get(&aes_ccm_nonce, sizeof(aes_ccm_nonce));
if (atlk_error(rc)) {
  fprintf(stderr, "rng_data_get: %s", atlk_rc_to_str(rc));
  return rc;
}

/* Print random AES-CCM nonce */
printf("AES-CCM nonce: ");
ecies_example_print_buffer(aes_ccm_nonce.value, sizeof(aes_ccm_nonce));
printf("\n");

/* Encrypt message with AES-CCM */
rc = aes_ccm_encrypt(&aes_key,
                     &aes_ccm_nonce,
                     example_msg,
                     sizeof(example_msg),
                     ciphertext,
                     &ciphertext_size,
                     &aes_ccm_tag);
if (atlk_error(rc)) {
  fprintf(stderr, "aes_ccm_encrypt: %s", atlk_rc_to_str(rc));
  return rc;
}

/* Print AES-CCM encrypted message and authentication tag */
printf("AES-CCM encrypted message: ");
ecies_example_print_buffer(ciphertext, ciphertext_size);
printf("\n");

printf("AES-CCM authentication tag: ");
ecies_example_print_buffer(aes_ccm_tag.value, sizeof(aes_ccm_tag));
printf("\n");

/* ECIES key size should be equal to: plaintext size + HMAC key size */

/* Create ECIES key and ephemeral public key */
rc = ecies_key_create(ecies_example_shared_info.curve,
                      &ecies_example_shared_info.bob_public_key,
                      &ephemeral_public_key,
                      ecies_key,
                      sizeof(ecies_key),
                      NULL,
                      0);
if (atlk_error(rc)) {
  fprintf(stderr, "ecdh_secret_create: %s", atlk_rc_to_str(rc));
```

```
    return rc;
  }

  assert(ephemeral_public_key.point_type == ECC_POINT_UNCOMPRESSED);

  /* Print ephemeral public key */
  printf("ECC ephemeral public key created:\n");
  printf("  x: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(ephemeral_public_key.x_coordinate));
  printf("  y: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(ephemeral_public_key.y_coordinate));

  /* Print ECIES key */
  printf("ECIES key: ");
  ecies_example_print_buffer(ecies_key, sizeof(ecies_key));
  printf("\n");

  /* Encrypt AES key using ECIES */
  rc = ecies_encrypt(SHA_256,
                     ecies_key,
                     sizeof(ecies_key),
                     &aes_key,
                     sizeof(aes_key),
                     encrypted_aes_key,
                     &encrypted_aes_key_size,
                     &ecies_authentication_tag);
  if (atlk_error(rc)) {
    fprintf(stderr, "ecies_encrypt: %s", atlk_rc_to_str(rc));
    return rc;
  }

  /* Print ECIES encrypted AES key and authentication tag */
  printf("ECIES encrypted AES key: ");
  ecies_example_print_buffer(encrypted_aes_key, encrypted_aes_key_size);
  printf("\n");

  printf("ECIES authentication tag: ");
  ecies_example_print_buffer(ecies_authentication_tag.value,
    sizeof(ecies_authentication_tag));
  printf("\n");

  /* Produce message */
  memcpy(msg->aes_ccm_ciphertext, ciphertext, ciphertext_size);
  msg->aes_ccm_ciphertext_size = ciphertext_size;
  msg->aes_ccm_nonce = aes_ccm_nonce;
  msg->aes_ccm_tag = aes_ccm_tag;
  memcpy(msg->ecies_encrypted_aes_key, &encrypted_aes_key, AES_KEY_SIZE);
  msg->ecies_authentication_tag = ecies_authentication_tag;
  msg->ecies_ephemeral_public_key = ephemeral_public_key;

  return ATLK_OK;
}

static atlk_rc_t
ecies_example_bob(const ecies_example_message_t *msg)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* ECIES decrypted AES key */
  aes_key_t aes_key = AES_KEY_INIT;
  /* AES key size in octets */
  size_t aes_key_size = sizeof(aes_key);
  /* ECIES key */
  uint8_t ecies_key[sizeof(aes_key) + ECIES_HMAC_KEY_SIZE] = { 0 };
  /* ECIES authentication tag */
  ecies_authentication_tag_t ecies_authentication_tag =
    ECIES_AUTHENTICATION_TAG_INIT;
  /* AES-CCM authentication tag */
  aes_ccm_authentication_tag_t aes_ccm_tag =
      AES_CCM_AUTHENTICATION_TAG_INIT;
  /* AES-CCM plaintext */
  uint8_t plaintext[ECIES_EXAMPLE_MESSAGE_MAX_SIZE] = { 0 };
  /* AES-CCM plaintext size in octets */
  size_t plaintext_size = sizeof(plaintext);
  /* Example failure indication */
  int failed = 1;

  printf("\n>>> Bob\n");

  /* Derive ECIES key */
  rc = hsm_ecies_key_derive(example_hsm_service,
                            ecies_example_bob_info.private_key_index,
                            &msg->ecies_ephemeral_public_key,
                            ecies_key,
                            sizeof(ecies_key),
                            NULL,
                            0);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_ecdh_secret_derive: %s", atlk_rc_to_str(rc));
    return rc;
  }

  /* Print ECIES key */
  printf("Derived ECIES key: ");
  ecies_example_print_buffer(ecies_key, sizeof(ecies_key));
  printf("\n");

  /* Decrypt AES key with ECIES */
  rc = ecies_decrypt(SHA_256,
                     ecies_key,
                     sizeof(ecies_key),
                     msg->ecies_encrypted_aes_key,
                     sizeof(msg->ecies_encrypted_aes_key),
                     aes_key.value,
                     &aes_key_size,
                     &ecies_authentication_tag);
  if (atlk_error(rc)) {
    fprintf(stderr, "ecies_decrypt: %s", atlk_rc_to_str(rc));
    return rc;
  }

  /* Print ECIES decrypted AES key and authentication tag */
  printf("ECIES decrypted AES key: ");
  ecies_example_print_buffer(aes_key.value, aes_key_size);
  printf("\n");

  printf("ECIES authentication tag: ");
  ecies_example_print_buffer(ecies_authentication_tag.value,
    sizeof(ecies_authentication_tag));
  printf("\n");

  /* Compare ECIES authentication tags */
  if (memcmp(&ecies_authentication_tag, &msg->ecies_authentication_tag,
      sizeof(ecies_authentication_tag)) != 0) {
    printf("ECIES encryption/decryption failed\n");
    goto out;
  }
  else {
    printf("ECIES encryption/decryption succeeded\n");
  }

  /* Decrypt message using AES-CCM */
  rc = aes_ccm_decrypt(&aes_key,
                       &msg->aes_ccm_nonce,
                       msg->aes_ccm_ciphertext,
                       msg->aes_ccm_ciphertext_size,
                       plaintext,
                       &plaintext_size,
                       &aes_ccm_tag);
  if (atlk_error(rc)) {
    fprintf(stderr, "aes_ccm_decrypt: %s", atlk_rc_to_str(rc));
    return rc;
  }

  /* Print AES-CCM decrypted message and authentication tag */
  printf("Decrypted message: %s\n", plaintext);
  printf("Decrypted message size: %lu\n", (long unsigned int)plaintext_size);

  printf("AES-CCM authentication tag: ");
  ecies_example_print_buffer(aes_ccm_tag.value, sizeof(aes_ccm_tag));
  printf("\n");

  /* Compare AES-CCM authentication tags */
  if (memcmp(&aes_ccm_tag, &msg->aes_ccm_tag, sizeof(aes_ccm_tag)) != 0) {
    printf("AES-CCM encryption/decryption failed\n");
    goto out;
  }
  else {
    printf("AES-CCM encryption/decryption succeeded\n");
  }

  /* Set failure indication flag */
  failed = 0;

out:
  printf("%s\n", failed ? "FAILURE" : "SUCCESS");

  return ATLK_OK;
}

/* ECIES example thread */
static TX_THREAD ecies_example_thread;
static uint8_t ecies_example_thread_stack[0x8000];
```

```
/* ECIES example thread priority */
#define ECIES_EXAMPLE_THREAD_PRIORITY 20

static void
ecies_example_thread_entry(ULONG input)
{
  /* Not using input */
  (void)input;
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* ECDSA example message */
  ecies_example_message_t message;
  /* HSM emulator configuration */
  hsm_emulator_config_t hsm_emulator_config =
      HSM_EMULATOR_CONFIG_INIT;

  /* Initialize the HSM emulator configuration */
  rc = ecc_default_service_get(&example_ecc_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "ecc_default_service_get: %s\n", atlk_rc_to_str(rc));
    goto out;
  }

  hsm_emulator_config.ecc_service_ptr = example_ecc_service;
#ifdef FS_EXIST
  hsm_emulator_config.nvm_file_path = HSM_EMULATOR_FILENAME;
#else
  hsm_emulator_config.nvm_file_path = NULL;
#endif /* FS_EXIST */
  /* Create HSM emulator service */
  rc = hsm_emulator_create(&hsm_emulator_config, &example_hsm_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_emulator_create: %s\n", atlk_rc_to_str(rc));
    goto out;
  }

  /* Initialize example message */
  rc = ecies_example_init();
  if (atlk_error(rc)) {
    goto out;
  }

  /* Produce example message by Alice */
  rc = ecies_example_alice(&message);
  if (atlk_error(rc)) {
    goto out;
  }

  /* Consume example message by Bob */
  rc = ecies_example_bob(&message);
  if (atlk_error(rc)) {
    goto out;
  }

out:
  if (atlk_error(rc)) {
    fprintf(stderr, "ERROR\n");
  }

  /* Delete HSM emulator service */
  hsm_service_delete(example_hsm_service);
}

void
craton_user_init(void)
{
  /* ThreadX return value */
  ULONG trv = TX_SUCCESS;

  /* Create example thread */
  trv = tx_thread_create(&ecies_example_thread, "ecies_example_thread",
                         ecies_example_thread_entry, 0,
                         ecies_example_thread_stack,
                         sizeof(ecies_example_thread_stack),
                         ECIES_EXAMPLE_THREAD_PRIORITY,
                         ECIES_EXAMPLE_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  return;
}

#else /* __CRATON_NO_ARC || __CRATON_ARC1 */

void craton_user_init(void)
{
}
```

```
#endif /* __CRATON_NO_ARC || __CRATON_ARC1 */
```

8.12 craton-threadx/crypto/secure-storage-example.c

```
/* Copyright (C) 2014-2015 Autotalks Ltd. */
#include <stdio.h>
#include <string.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>

#include <atlk/sdk.h>
#include <atlk/aes.h>
#include <atlk/hsm_service.h>
#include <atlk/hsm_emulator.h>

#if defined __CRATON_NO_ARC || defined __CRATON_ARC1

/*
  CRATON ThreadX Secure Storage Example

  This example demonstrates usage of the HSM API to implement
  secure certificate storage using non-secure NVM for code running on top
  of CRATON processor with ThreadX RTOS.

  The device used in this example is a "HSM emulator", a term used to describe
  an emulated HSM device. The differences between an emulated HSM service
  and a real one are:
  - The emulated HSM service is created via hsm_emulator_create().
  - The implementation is not tamper-resistant because it uses general purpose
    hardware instead of tamper-resistant hardware.

  The purpose of the emulated HSM is basic software integration with
  the HSM API on a hardware platform that doesn't have a working HSM chip.
*/

/* HSM emulator service */
static hsm_service_t *example_hsm_service = NULL;

/* ECC service */
static ecc_service_t *example_ecc_service = NULL;

/* Number of NVM cells to configure for HSM */
#define SECURE_STORAGE_EXAMPLE_HSM_NVM_NUM_CELLS 128

/* Helper macros */
#define DIV_ROUND_UP(n, d) (((n) + (d) - 1) / (d))
#define ROUND_UP(n, d) (DIV_ROUND_UP(n, d) * (d))

/* Helper function for converting buffer to hex */
static void
buffer_to_line(const void *buf, size_t buf_len, char *line)
{
  const uint8_t *ptr = buf;
  char *pos = &line[0];
  size_t i;

  if (!buf_len) {
    *pos = '\0';
    return;
  }

  if (buf_len > 16) {
    buf_len = 16;
  }

  for (i = 0; i < buf_len - 1; i++) {
    pos += sprintf(pos, "%02x ", ptr[i]);
  }
  pos += sprintf(pos, "%02x", ptr[i]);
}

/* Print buffer to standard output */
static void
buffer_print(const void *buf, size_t len)
{
  const uint8_t *ptr = buf;
  size_t i, line_len, remaining = len;
  char line[80];

  for (i = 0; i < len; i += 16) {
    line_len = remaining < 16 ? remaining : 16;
    remaining -= 16;
    buffer_to_line(ptr + i, line_len, line);
    printf("  %.8lx: %s\n", (unsigned long)i, line);
  }
}

static atlk_rc_t
secure_storage_example(void)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* HSM NVM configuration */
  hsm_nvm_config_t hsm_nvm_config = HSM_NVM_CONFIG_INIT;
  /* Dummy root certificate */
  const char root_cert[] = "Dummy Root Certificate";
  /* Dummy pseudonym certificate */
  const char pseudonym_cert[] = "Dummy Pseudonym Certificate";
  /* AES-CMAC authentication tag of root certificate */
  aes_cmac_tag_t root_cert_tag = AES_CMAC_TAG_INIT;
  /* Buffer to store plaintext/ciphertext */
  uint8_t buf[64];
  /* Buffer size */
  size_t buf_size;
  /* AES-CBC initialization vector */
  aes_cbc_iv_t iv;

  printf("Initializing NVM to contain %u cells\n",
    SECURE_STORAGE_EXAMPLE_HSM_NVM_NUM_CELLS);

  hsm_nvm_config.num_of_cells = SECURE_STORAGE_EXAMPLE_HSM_NVM_NUM_CELLS;

  /* Initialize HSM NVM */
  rc = hsm_nvm_init(example_hsm_service, &hsm_nvm_config);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_nvm_init: %s", atlk_rc_to_str(rc));
    return rc;
  }

  printf("Root certificate:\n");
  buffer_print(root_cert, sizeof(root_cert));

  /* Compute AES-CMAC for the root certificate */
  rc = hsm_host_nvm_aes_cmac_compute(example_hsm_service,
                                     root_cert,
                                     sizeof(root_cert),
                                     &root_cert_tag);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_host_nvm_aes_cmac_compute: %s\n", atlk_rc_to_str(rc));
    return rc;
  }

  printf("Root certificate authentication tag:\n");
  buffer_print(root_cert_tag.value, sizeof(root_cert_tag.value));

  /* Now the root certificate + tag can be stored on a non-secure NVM */

  printf("Pseudonym certificate:\n");
  buffer_print(pseudonym_cert, sizeof(pseudonym_cert));

  /*
   * Pad the pseudonym certificate so the total size will be a multiple of
   * the AES block size (= 16).
   */
  memset(buf, 0, sizeof(buf));
  memcpy(buf, pseudonym_cert, sizeof(pseudonym_cert));
  buf_size = ROUND_UP(sizeof(pseudonym_cert), 16);

  /* Encrypt the pseudonym certificate with AES-CBC */
  rc = hsm_host_nvm_aes_cbc_encrypt(example_hsm_service,
                                    buf, buf_size, &iv, buf, &buf_size);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_host_nvm_aes_cbc_encrypt: %s\n", atlk_rc_to_str(rc));
    return rc;
  }

  printf("AES-CBC initialization vector:\n");
  buffer_print(iv.value, sizeof(iv.value));

  printf("AES-CBC encrypted pseudonym certificate (with padding):\n");
  buffer_print(buf, buf_size);

  /* Decrypt the pseudonym certificate with AES-CBC */
  rc = hsm_host_nvm_aes_cbc_decrypt(example_hsm_service,
                                    &iv, buf, buf_size, buf, &buf_size);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_host_nvm_aes_cbc_decrypt: %s\n", atlk_rc_to_str(rc));
    return rc;
  }
  printf("AES-CBC decrypted pseudonym certificate (with padding):\n");
  buffer_print(buf, buf_size);

  printf("AES-CBC decrypted pseudonym certificate (without padding):\n");
  buffer_print(buf, sizeof(pseudonym_cert));

  return ATLK_OK;
}

void
craton_user_init(void)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* HSM emulator configuration */
  hsm_emulator_config_t hsm_emulator_config =
      HSM_EMULATOR_CONFIG_INIT;

  /* Get default ECC service */
  rc = ecc_default_service_get(&example_ecc_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "ecc_default_service_get: %s\n", atlk_rc_to_str(rc));
    goto out;
  }

  /* Initialize the HSM emulator configuration */
  hsm_emulator_config.ecc_service_ptr = example_ecc_service;
  hsm_emulator_config.nvm_file_path = NULL;

  /* Initialize AES keys to be used for secure certificate storage */
  memset(hsm_emulator_config.host_nvm_authentication_key.value, 0xAA,
         sizeof(hsm_emulator_config.host_nvm_authentication_key.value));
  memset(hsm_emulator_config.host_nvm_encryption_key.value, 0xBB,
         sizeof(hsm_emulator_config.host_nvm_encryption_key.value));

  /* Create HSM emulator service */
  rc = hsm_emulator_create(&hsm_emulator_config, &example_hsm_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_emulator_create: %s\n", atlk_rc_to_str(rc));
    goto out;
  }

  /* Run example */
  rc = secure_storage_example();
  if (rc) {
    goto out;
  }

out:
  if (atlk_error(rc)) {
    fprintf(stderr, "ERROR\n");
  }

  /* Delete HSM emulator service */
  hsm_service_delete(example_hsm_service);

  /* Delete ECC service */
  ecc_service_delete(example_ecc_service);
}

#else /* __CRATON_NO_ARC || __CRATON_ARC1 */

void craton_user_init(void)
{
}

#endif /* __CRATON_NO_ARC || __CRATON_ARC1 */
```

8.13 craton-threadx/diagnostics/craton-user-abort-example.c

```
/* Copyright (C) 2015 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>

#if defined __CRATON_NO_ARC

#include <craton/user.h>

/*
  CRATON User Abort Handler Example

  This example demonstrates basic usage of craton_user_abort_handler
  to override default exception handling on ARM CPU.
  craton_user_abort_handler can also be defined for each of the
  ARC CPUs

  The example will create an exception by writing to illegal address. The
  user-specific exception handler will be called upon exception.
*/

/* Illegal address for writing */
#define EXAMPLE_ILLEGAL_ADDRESS 0x0

/* Helper function to display informative string */
char *
example_exception_reason_to_string(exception_arm_reason_t reason)
{
  switch(reason) {
  case EXC_ARM_REASON_BACKGROUND:
    return "Background";
  case EXC_ARM_REASON_ALIGNMENT:
    return "Bad alignment";
  case EXC_ARM_REASON_DEBUG_EVENT:
    return "Debug event";
  case EXC_ARM_REASON_SYNC_EXTERNAL:
    return "Synchronous external event";
  case EXC_ARM_REASON_ASYNC_EXTERNAL:
    return "Asynchronous external event";
  case EXC_ARM_REASON_PERMISSION:
    return "Permission denied";
  case EXC_ARM_REASON_ASYNC_ECC:
    return "Asynchronous ECC";
  case EXC_ARM_REASON_SYNC_ECC:
    return "Synchronous ECC";
  case EXC_ARM_REASON_ABNORMAL_EXIT:
    return "Abnormal exit";
  default:
    return "Unknown reason";
  }
}

/* User-specific implementation for abort handling */
void
craton_user_abort_handler(const exception_info_t *info)
{
  printf("Craton user abort handler called.\n");
  printf("Exception reason: %s\n",
         example_exception_reason_to_string(info->reason));
}

void craton_user_init(void)
{
  printf("Application will cause an exception within 5 seconds...\n");
  usleep(5000000);

  /* Create exception by writing to illegal address */
  *(int *)EXAMPLE_ILLEGAL_ADDRESS = 1;

  return;
}

#else /* __CRATON_NO_ARC */

void craton_user_init(void)
{
}

#endif  /* __CRATON_NO_ARC  */
```

8.14 craton-threadx/dot4/dot4-channel-switching-example.c

```
/* Copyright (C) 2015 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>

#include <tx_api.h>

#include <atlk/sdk.h>
#include <atlk/v2x.h>
#include <atlk/v2x_service.h>
#include <atlk/mib_service.h>
#include <atlk/mibs/nav-mib.h>

/*
  CRATON ThreadX IEEE 1609.4 Channel Switching Example
  This example demonstrates basic usage of 1609.4 channel switching for
  code running on top of CRATON processor with ThreadX RTOS.

  Channel switching is configured such that channel A takes the role of CCH
  and channel B takes the role of SCH.

  To simplify, this example sends the same frame on CCH and SCH every 20
  milliseconds in a single thread. Frames are physically transmitted when
  their channel is enabled, otherwise they are queued.

  @todo Note that sync tolerance is not currently implemented, i.e.
  when active, channel switching is enabled regardless of system time status.
*/

/* Example thread priority */
#define TX_THREAD_PRIORITY 40

#if defined __CRATON_NO_ARC || defined __CRATON_ARC1

/* TX thread */
static TX_THREAD tx_thread;
static uint8_t tx_thread_stack[0x1000];
static void tx_thread_entry(ULONG input);

/* End indication thread */
static TX_THREAD end_indication_thread;
static uint8_t end_indication_thread_stack[0x1000];
static void end_indication_thread_entry(ULONG input);

/* Sync loss indication */
static int is_sync_loss = 0;

/* Example channel access configuration */
#define V2X_DOT4_CHANNEL_START_REQUEST_INIT {  \
 .if_index = V2X_IF_INDEX_NA,                  \
 .channel_id = V2X_CHANNEL_ID_INIT,            \
 .time_slot = V2X_TIME_SLOT_NA,                \
 .immediate_access = 0                          \
}

/* V2X service */
static v2x_service_t *v2x_service = NULL;

/* V2X sockets */
static v2x_socket_t *v2x_socket = NULL;

/* CCH interface index used in this example */
#define IF_INDEX 1

/* Protocol identifier for example messages */
#define PROTO_ID 0x102ULL

/* CCH and SCH frequencies used in this example */
#define CCH_CHANNEL_NUM 172
#define SCH_CHANNEL_NUM 178

/* Example message format string: <ch>: Example <seq_num> */
static const char msg_fmt[] = "%s: Example %" PRIu32;

/* Example message string maximum length */
static const size_t msg_size_max = sizeof(msg_fmt) + 10;

void craton_user_init(void)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* ThreadX return value */
  ULONG trv = TX_SUCCESS;
  /* V2X socket configuration */
  v2x_socket_config_t config = V2X_SOCKET_CONFIG_INIT;
  /* MIB service */
  mib_service_t *mib_service = NULL;
  /* NAV system accrutacy */
  mib_navSysTimeAccuracy_t system_accuracy;

  /* CCH- V2X DOT4 channel start request */
  v2x_dot4_channel_start_request_t cch_start_request =
      V2X_DOT4_CHANNEL_START_REQUEST_INIT;

  /* SCH- V2X DOT4 channel start request */
  v2x_dot4_channel_start_request_t sch_start_request =
      V2X_DOT4_CHANNEL_START_REQUEST_INIT;

  /* Get default V2X service instance */
  rc = v2x_default_service_get(&v2x_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "v2x_default_service_get: %s\n", atlk_rc_to_str(rc));
    return;
  }

  /* Get default MIB service instance */
  rc = mib_default_service_get(&mib_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "mib_default_service_get: %s\n", atlk_rc_to_str(rc));
    return;
  }

  printf("\nWaiting for NAV accuracy to be 1 ms for\n"
         "demonstraing 1609.4 alternating mode\n");

  while (1) {
    rc = mib_get_navSysTimeAccuracy(mib_service, &system_accuracy);
    if (atlk_error(rc)) {
      fprintf(stderr, "mib_get_navSysTimeAccuracy: %s\n", atlk_rc_to_str(rc));
      return;
    }
    if (system_accuracy != MIB_navSysTimeAccuracy_milliSec1) {
      printf("System is not within 1 ms accuracy, retry within 10 seconds\n");
      usleep(10000000);
    }
    else {
      printf("System accuracy is OK, starting transmission\n");
      break;
    }
  }

  /* Configure IEEE Std 1609.4-2016 multi-channel access */

  /* CCH: slot = 0, immediate = 0 */
  cch_start_request.if_index = IF_INDEX;
  cch_start_request.time_slot = V2X_TIME_SLOT_0;
  cch_start_request.channel_id.op_class =
      V2X_OP_CLASS_US_ITS_5GHZ_SPACING_10MHZ;
  cch_start_request.channel_id.channel_num = CCH_CHANNEL_NUM;
  rc = v2x_dot4_channel_start(v2x_service,
                              &cch_start_request,
                              &atlk_wait_forever);
  if (atlk_error(rc)) {
    fprintf(stderr, "CCH v2x_dot4_channel_start: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* SCH: slot = 1, immediate = 0 */
  sch_start_request.if_index = IF_INDEX;
  sch_start_request.time_slot = V2X_TIME_SLOT_1;
  sch_start_request.channel_id.op_class =
      V2X_OP_CLASS_US_ITS_5GHZ_SPACING_10MHZ;
  sch_start_request.channel_id.channel_num = SCH_CHANNEL_NUM;
  rc = v2x_dot4_channel_start(v2x_service,
                              &sch_start_request,
                              &atlk_wait_forever);
  if (atlk_error(rc)) {
    fprintf(stderr, "SCH v2x_dot4_channel_start: %s\n", atlk_rc_to_str(rc));
    goto error;
  }
  /* Set socket configuration for CCH */
  config.if_index = IF_INDEX;
  config.protocol.protocol_id = PROTO_ID;

  /* Create a V2X socket bound to CCH */
  rc = v2x_socket_create(v2x_service, &v2x_socket, &config);
  if (atlk_error(rc)) {
    fprintf(stderr, "v2x_socket_create: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Create End Indication thread */
  trv = tx_thread_create(&end_indication_thread, "end_indication_thread",
                         end_indication_thread_entry, 0,
                         &end_indication_thread_stack,
                         sizeof(end_indication_thread_stack),
                         TX_THREAD_PRIORITY,
                         TX_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  /* Create TX thread */
  trv = tx_thread_create(&tx_thread, "tx_thread",
                         tx_thread_entry, 0,
                         &tx_thread_stack,
                         sizeof(tx_thread_stack),
                         TX_THREAD_PRIORITY,
                         TX_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  return;

error:
  /* Clean-up resources */
  v2x_socket_delete(v2x_socket);
  v2x_service_delete(v2x_service);

  return;
}

void
end_indication_thread_entry(ULONG input)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;

  /* DOT4 channel end indication */
  v2x_dot4_channel_end_indication_t indication;

  /* Not using input */
  (void)input;

  indication.if_index = IF_INDEX;

  while (1) {

    /* polling for channel indication */
    rc = v2x_dot4_channel_end_receive(v2x_service, &indication, NULL);
    switch (rc) {
    case ATLK_OK:
      printf("Received 1609.4 MLMEX-CHEND.indication: "
             "interface %d, channel num %d, op class %d, reason %d\n",
             indication.if_index,
             indication.channel_id.op_class,
             indication.channel_id.channel_num,
             indication.reason);

      printf("Suspending sending on service channel.\n");
      /* Inform TX thread that we have sync loss */
      is_sync_loss = 1;
      break;

    case ATLK_E_NOT_READY:
      break;

    default:
      printf("v2x_dot4_channel_end_receive: %s", atlk_rc_to_str(rc));
      return;
    }

    /* sleep 10msec between each poll */
    usleep(10000);
  }
}

void tx_thread_entry(ULONG input)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* CCH Send parameters */
  v2x_send_params_t cch_send_params = V2X_SEND_PARAMS_INIT;
  /* SCH Send parameters */
  v2x_send_params_t sch_send_params = V2X_SEND_PARAMS_INIT;
  /* Message counter */
  uint32_t msg_count = 0;

  /* Not using input */
  (void)input;

  /* set CCH send parama structure */
  cch_send_params.power_dbm8 = -80;
  cch_send_params.channel_id.op_class =
      V2X_OP_CLASS_US_ITS_5GHZ_SPACING_10MHZ;
  cch_send_params.channel_id.channel_num = CCH_CHANNEL_NUM;

  /* set SCH send parama structure */
  sch_send_params.power_dbm8 = -80;
  sch_send_params.channel_id.op_class =
      V2X_OP_CLASS_US_ITS_5GHZ_SPACING_10MHZ;
  sch_send_params.channel_id.channel_num = SCH_CHANNEL_NUM;

  while (1) {
    /* TX buffer */
    char buf[msg_size_max];
    /* Print message into buffer (with terminating \0) and update its size */
    size_t size = 1 + snprintf(buf, sizeof(buf), msg_fmt, "CCH", msg_count);

    /* Transmit V2X PDU on CCH */
    rc = v2x_send(v2x_socket, buf, size, &cch_send_params, NULL);
    if (atlk_error(rc)) {
      fprintf(stderr, "CCH v2x_send: %s\n", atlk_rc_to_str(rc));
    }

    /* In case of sync loss, dot4 system goes back to continuous mode on
       slot 0. This means we shouldn't transmit on slot 1 (SCH in our
       example)
    */
    if (!is_sync_loss) {

      /* Print message into buffer (with terminating \0) and update its size */
      size = 1 + snprintf(buf, sizeof(buf), msg_fmt, "SCH", msg_count);

      /* Transmit V2X PDU on SCH */

      rc = v2x_send(v2x_socket, buf, size, &sch_send_params, NULL);
      if (atlk_error(rc)) {
        fprintf(stderr, "SCH v2x_send: %s\n", atlk_rc_to_str(rc));
      }

    }

    /* Increment message count */
    msg_count++;

    /* Sleep 20 milliseconds between transmissions */
    usleep(20000);
  }

}

#else /* __CRATON_NO_ARC || __CRATON_ARC1 */

void craton_user_init(void)
{
}

#endif /* __CRATON_NO_ARC || __CRATON_ARC1 */
```

8.15 craton-threadx/firmware/fw-update-example.c

```
/* Copyright (C) 2015 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/stat.h>

#include <tx_api.h>

#include <atlk/sdk.h>

#include <craton/fs.h>
#include <craton/nor_flash.h>
#include <craton/fw_uimage.h>

/*
  CRATON ThreadX Firmware Update Example

  This example demonstrates basic usage of NOR Flash API for code running on
  top of CRATON processor with ThreadX RTOS.

  Expected firmware image format is U-Boot uImage. For simplicty, the example
  assumes a valid firmware image is at 'A:/uImage' (i.e. on microSD).

  Warning! When 'A:/uImage' exists and is valid, this example will re-write
  main firmware image partition with this file.
*/

/* Firmware update thread priority */
#define FW_UPDATE_THREAD_PRIORITY 20

/* Firmware update thread */
static TX_THREAD fw_update_thread;
static uint8_t fw_update_thread_stack[0x8000];
static void fw_update_thread_entry(ULONG input);

/* Firmware image path */
#define FW_IMAGE_PATH "A:/uImage"
```

```
/* Firmware image max size */
#define FW_IMAGE_SIZE_MAX 0x200000

/* Firmware image buffer */
static uint8_t fw_image_buffer[FW_IMAGE_SIZE_MAX];

void craton_user_init(void)
{
  /* ThreadX return value */
  ULONG trv = TX_SUCCESS;

  /* Create firmware update thread */
  trv = tx_thread_create(&fw_update_thread, "fw_update_thread",
                         fw_update_thread_entry, 0,
                         fw_update_thread_stack,
                         sizeof(fw_update_thread_stack),
                         FW_UPDATE_THREAD_PRIORITY,
                         FW_UPDATE_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  return;
}

static void fw_update_thread_entry(ULONG input)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* POSIX return value */
  int rv = 0;
  /* File statistics */
  struct stat st;
  /* Partition table */
  norfl_part_table_t part_table = NORFL_PART_TABLE_INIT;
  /* Size of main firmware image partition */
  uint32_t main_fw_part_size;
  /* Size of firmware image file */
  uint32_t fw_image_size;
  /* File descriptor */
  int fd = -1;
  /* Not using input */
  (void)input;

  printf("Start firmware update example...\n");

  /* Enable the usage of file system in this thread */
  rv = fs_thread_enable();
  if (rv == -1) {
    fprintf(stderr, "fs_thread_enable failed, errno: %d\n", errno);
    return;
  }

  /* Get firmware image file statistics */
  rv = stat(FW_IMAGE_PATH, &st);
  if (rv == -1) {
    fprintf(stderr, "stat failed, errno: %d\n", errno);
    return;
  }

  /* Size of firmware image file */
  fw_image_size = (uint32_t)st.st_size;

  /* Read partition table */
  rc = norfl_part_table_read(&part_table);
  if (atlk_error(rc)) {
    fprintf(stderr, "norfl_part_table_read: %s\n", atlk_rc_to_str(rc));
    return;
  }

  /* Size of main firmware image partition */
  main_fw_part_size =
    part_table.part_info[NORFL_PART_FIRMWARE_MAIN].part_size;

  /* Make sure firmware image size makes sense */
  assert(fw_image_size <= main_fw_part_size);
  assert(fw_image_size <= sizeof(fw_image_buffer));

  /* Open firmware image in read-only mode */
  fd = open(FW_IMAGE_PATH, O_RDONLY);
  if (fd == -1) {
    fprintf(stderr, "open failed, errno %d\n", errno);
    return;
  }

  /* Read image into buffer */
  rv = read(fd, fw_image_buffer, fw_image_size);
  if (rv == -1) {
    fprintf(stderr, "read failed, errno %d\n", errno);
    goto exit;
  }
  assert((uint32_t)rv == fw_image_size);

  /* Make sure firmware image is a valid uImage file */
  if (!fw_uimage_valid(fw_image_buffer, fw_image_size)) {
    fprintf(stderr, "fw_uimage_valid returned false\n");
    goto exit;
  }

  /* Re-write main firmware image */
  rc = norfl_part_rewrite(NORFL_PART_FIRMWARE_MAIN,
                          fw_image_buffer, fw_image_size);
  if (atlk_error(rc)) {
    fprintf(stderr, "norfl_part_rewrite: %s\n", atlk_rc_to_str(rc));
    goto exit;
  }

  printf("Firmware updated successfully!\n");

exit:
  /* Clean-up resources */
  close(fd);

  return;
}
```

8.16 craton-threadx/fs/fs-example.c

```
/* Copyright (C) 2014-2015 Autotalks Ltd. */
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <dirent.h>

#include <sys/stat.h>

#include <tx_api.h>
#include <tx_posix.h>

#include <craton/fs.h>

/*
  CRATON ThreadX File System Example

  This example demonstrates usage of file system API for code running on top
  of CRATON processor with ThreadX RTOS.

  The example demonstrates supported file system functions. It can be run on
  microSD or flash devices:

    - Device 'A' is mapped to microSD card (available on some boards).
    - Device 'B' is mapped to flash device.
*/

/* Device used in this example */
#define EXAMPLE_FS_DEVICE "B:/"

/* Maximum number of characters in path name */
#define EXAMPLE_DIR_NAME_MAX 80

/* Size of buffer used in this example */
#define EXAMPLE_BUFFER_SIZE 128

/* Name of directory used in this example */
#define EXAMPLE_DIR "example_dir"

/* Name of 1st file used in this example */
#define EXAMPLE_FILE_1 "example_file_1"

/* Name of 2nd file used in this example */
#define EXAMPLE_FILE_2 "example_file_2"

/* Format of example line */
#define EXAMPLE_LINE_FMT "example_line_%d"

#ifdef __CRATON_ARM

int example_cleanup(void)
{
  /* Directory stream object */
  DIR *dir = NULL;
  /* Directory entry object */
  struct dirent *dirent = NULL;
  /* Directory statistics */
  struct fs_dirstat dirstat;
  /* Return code */
  int rc = 0;

  /* Change directory to root of device */
  rc = chdir(EXAMPLE_FS_DEVICE);
  if (rc == -1) {
    rc = -errno;
    fprintf(stderr, "chdir failed, errno %d\n", rc);
    return rc;
  }
  printf("Changed to directory '%s'\n", EXAMPLE_FS_DEVICE);

  /* Obtain directory statistics */
  rc = fs_dirstat(EXAMPLE_DIR, &dirstat, FS_DIRSTAT_DEPTH_MAX);
  if (rc == -1) {
    rc = -errno;
    fprintf(stderr, "fs_dirstat failed, errro %d\n", rc);
    return rc;
  }
  printf("Directory statistics: number of dirs %d, number of files %d\n",
         dirstat.num_of_dirs, dirstat.num_of_files);

  /* Open the example directory */
  dir = opendir(EXAMPLE_DIR);
  if (dir == NULL) {
    rc = -errno;
    fprintf(stderr, "opendir failed, errno %d\n", rc);
    goto error;
  }
  printf("Opened directory '%s'\n", EXAMPLE_DIR);

  printf("Recursing over directory tree...\n");
  while ((dirent = readdir(dir))) {
    struct stat status;
    char dpath[DIRENT_NAME_MAX];

    /* Stat expects a relative path */
    sprintf(dpath, "%s/%s", EXAMPLE_DIR, dirent->d_name);

    /* Get file status */
    rc = stat(dpath, &status);
    if (rc == -1) {
      rc = -errno;
      fprintf(stderr, "stat failed, errno %d", rc);
      goto error;
    }

    switch(status.st_mode) {
    case S_IFDIR:
      printf("Found directory '%s'\n", dirent->d_name);
      break;
    case S_IFREG:
      printf("Found file '%s', removing it...\n", dirent->d_name);

      /* Delete the file */
      rc = remove(dpath);
      if (rc == -1) {
        rc = -errno;
        fprintf(stderr, "remove failed, errno %d\n", rc);
        goto error;
      }
      break;
    default:
      fprintf(stderr, "Unsupported file type %d", status.st_mode);
      break;
    }
  }

error:
  /* Close directory (cleanup resources used by opendir) */
  rc = closedir(dir);
  if (rc == -1) {
    rc = -errno;
    fprintf(stderr, "closedir failed, errno %d\n", rc);
  }
  printf("Closed directory '%s'\n", EXAMPLE_DIR);

  /* Remove example directory */
  rc = rmdir(EXAMPLE_DIR);
  if (rc == -1) {
    rc = -errno;
    fprintf(stderr, "rmdir failed, errno %d\n", rc);
  }
  printf("Removed directory '%s'\n", EXAMPLE_DIR);
  return rc;
}

void craton_user_init(void)
{
  /* Example buffer */
  char buf[EXAMPLE_BUFFER_SIZE];
  /* Directory path */
  char dirpath[EXAMPLE_DIR_NAME_MAX];
  /* File descriptor */
  int fd = -1;
  /* Current directory */
  char *current_dir = NULL;
  /* Size of read data */
  ssize_t size = 0;
  /* Return code */
  int rc = 0;

  printf("Start file system example...\n");

  /* Change directory to root of device */
  rc = chdir(EXAMPLE_FS_DEVICE);
  if (rc == -1) {
    rc = -errno;
    fprintf(stderr, "chdir failed, errno %d\n", rc);
    goto error;
  }
  printf("Changed directory to '%s'\n", EXAMPLE_FS_DEVICE);

  /* Make a new directory */
  rc = mkdir(EXAMPLE_DIR, 0);
  if (rc == -1) {
    rc = -errno;
    fprintf(stderr, "mkdir failed, errno %d\n", rc);
    goto error;
  }
  printf("Made new directory '%s'\n", EXAMPLE_DIR);

  /* Change directory to the directory we created */
  rc = chdir(EXAMPLE_DIR);
  if (rc == -1) {
    rc = -errno;
    fprintf(stderr, "chdir failed, errno %d\n", rc);
    goto error;
  }
  printf("Changed directory to '%s'\n", EXAMPLE_DIR);


  /* Get current directory */
  current_dir = getcwd(dirpath, sizeof(dirpath));
  if (current_dir == NULL) {
    rc = -errno;
    fprintf(stderr, "getcwd failed, errno %d\n", rc);
    goto error;
  }
  printf("Current directory is '%s'\n", current_dir);

  /* Open a new file (create it if it does not exist) */
  fd = open(EXAMPLE_FILE_1, O_APPEND | O_RDWR);
  if (fd == -1) {
    rc = -errno;
    fprintf(stderr, "open failed, errno %d\n", rc);
    goto error;
  }
  printf("Opened file '%s'\n", EXAMPLE_FILE_1);

  /* Write 10 example lines */
  for (int i = 1; i <= 5; ++i) {
    snprintf(buf, sizeof(buf), EXAMPLE_LINE_FMT "\n", i);
    size = write(fd, buf, strlen(buf) + 1);
    if (size == -1) {
      rc = -errno;
      fprintf(stderr, "write failed, errno %d\n", rc);
      goto error;
    }
    printf("Written line '" EXAMPLE_LINE_FMT "'\n", i);
  }

  /* Close the file */
  rc = close(fd);
  if (rc == -1) {
    rc = -errno;
    fprintf(stderr, "close failed, errno %d\n", rc);
    goto error;
  }
  printf("Closed file '%s'\n", EXAMPLE_FILE_1);
  /* Mark file descriptor as closed */
  fd = -1;

  /* Rename the file */
  rc = rename(EXAMPLE_FILE_1, EXAMPLE_FILE_2);
  if (rc == -1) {
    rc = -errno;
    fprintf(stderr, "rename failed, errno %d\n", rc);
    goto error;
  }
  printf("Renamed file '%s' to '%s'\n", EXAMPLE_FILE_1, EXAMPLE_FILE_2);

  /* Open the renamed file in read-only mode */
  fd = open(EXAMPLE_FILE_2, O_RDONLY);
  if (fd == -1) {
    rc = -errno;
    fprintf(stderr, "open failed, errno %d\n", rc);
    goto error;
  }
  printf("Opened file '%s'\n", EXAMPLE_FILE_2);

  /* Seek to 2nd line (assuming all lines are of same length) */
  rc = lseek(fd, size, SEEK_SET);
  if (rc == -1) {
    rc = -errno;
    fprintf(stderr, "lseek failed, errno %d\n", rc);
    goto error;
  }
  printf("Seeked to 2nd line written.\n");

  /* Read 2nd line (excluding newline char) */
  size = read(fd, buf, size - 1);
  if (size == -1) {
    rc = -errno;
    fprintf(stderr, "read failed, errno %d\n", rc);
    goto error;
  }
  printf("Read 2nd line:\n%s", buf);

  /* Close the file */
  rc = close(fd);
  if (rc == -1) {
    rc = -errno;
    fprintf(stderr, "close failed, errno %d\n", rc);
    goto error;
  }
  printf("Closed file '%s'\n", EXAMPLE_FILE_2);

  /* Mark file descriptor as closed */
  fd = -1;

  /* Cleanup example resources */
  rc = example_cleanup();
  if (rc == 0) {
    printf("File system example completed successfully!\n");
  }
  return;

error:
  example_cleanup();
}

#else /* __CRATON_ARM */

void craton_user_init(void)
{
}

#endif /* __CRATON_ARM */
```

8.17 craton-threadx/gnss-teseo/gnss-teseo-fw-update-example.c

```
/* Copyright (C) 2015-2016 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/stat.h>

#include <tx_api.h>

#include <atlk/sdk.h>
#include <atlk/gnss.h>
#include <atlk/gnss_teseo.h>
```

```
#include <craton/fs.h>

/*
  CRATON ThreadX Teseo FW Update Example

  This example demonstrates basic usage of Teseo firmware update API for code
  running on top of CRATON processor with ThreadX RTOS.

  Expected firmware image format is a Teseo UPG image. For simplicty, the
  example assumes a valid firmware image is at 'A:/image.bin' (i.e. on microSD).
*/

/* Firmware update thread priority */
#define TESEO_FW_UPDATE_THREAD_PRIORITY 20

/* Firmware update thread */
static TX_THREAD gnss_teseo_fw_update_thread;
static uint8_t gnss_teseo_fw_update_thread_stack[0x8000];
static void gnss_teseo_fw_update_thread_entry(ULONG input);

/* Firmware image path */
#define FW_IMAGE_PATH "A:/image.bin"

/* Firmware image max size */
#define FW_IMAGE_SIZE_MAX 0x100000

/* Firmware image buffer */
static uint8_t fw_image_buffer[FW_IMAGE_SIZE_MAX];

void craton_user_init(void)
{
  /* ThreadX return value */
  ULONG trv = TX_SUCCESS;

  /* Create firmware update thread */
  trv = tx_thread_create(&gnss_teseo_fw_update_thread,
                         "gnss_teseo_fw_update_thread",
                         gnss_teseo_fw_update_thread_entry, 0,
                         gnss_teseo_fw_update_thread_stack,
                         sizeof(gnss_teseo_fw_update_thread_stack),
                         TESEO_FW_UPDATE_THREAD_PRIORITY,
                         TESEO_FW_UPDATE_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  return;
}

static void gnss_teseo_fw_update_thread_entry(ULONG input)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* POSIX return value */
  int rv = 0;
  /* File statistics */
  struct stat st;
  /* Size of firmware image file */
  uint32_t fw_image_size;
  /* File descriptor */
  int fd = -1;
  /* Teseo firmware update parameters */
  gnss_teseo_fw_update_params_t params =
      GNSS_TESEO_FW_UPDATE_PARAMS_INIT;
  /* Not using input */
  (void)input;

  printf("Start Teseo firmware update example...\n");

  /* Enable the usage of file system in this thread */
  rv = fs_thread_enable();
  if (rv == -1) {
    fprintf(stderr, "fs_thread_enable failed, errno: %d\n", errno);
    return;
  }

  /* Get firmware image file statistics */
  rv = stat(FW_IMAGE_PATH, &st);
  if (rv == -1) {
    fprintf(stderr, "stat failed, errno: %d\n", errno);
    return;
  }

  /* Size of firmware image file */
  assert(st.st_size <= INT32_MAX);
  fw_image_size = (uint32_t)st.st_size;

  /* Make sure firmware image size makes sense */
  assert(fw_image_size <= sizeof(fw_image_buffer));

  /* Open firmware image in read-only mode */
  fd = open(FW_IMAGE_PATH, O_RDONLY);
  if (fd == -1) {
    fprintf(stderr, "open failed, errno %d\n", errno);
    return;
  }

  /* Read image into buffer */
  rv = read(fd, fw_image_buffer, fw_image_size);
  if (rv == -1) {
    fprintf(stderr, "read failed, errno %d\n", errno);
    goto exit;
  }
  assert((uint32_t)rv == fw_image_size);

  /* Set Teseo firmware update parameters */
  params.fw_image = fw_image_buffer;
  params.fw_image_size = fw_image_size;
  params.nmea_speed_bps = UART_SPEED_230400_BPS;
  params.download_speed_bps = UART_SPEED_921600_BPS;
  params.erase_nvm_area = 1;
  params.nvm_area_size_kb = 0;
  params.recovery_mode = 0;
  params.sched_params.priority = TESEO_FW_UPDATE_THREAD_PRIORITY;

  /* Update Teseo firmware */
  rc = gnss_teseo_fw_update(&params);
  if (atlk_error(rc)) {
    fprintf(stderr, "gnss_teseo_fw_update failed: %s\n", atlk_rc_to_str(rc));
    goto exit;
  }

  printf("Firmware updated successfully!\n");

exit:
  /* Clean-up resources */
  close(fd);

  return;
}
```

8.18 craton-threadx/gnss-teseo/gnss-teseo-sou-example.c

```
/* Copyright (C) 2015-2016 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <unistd.h>

#include <tx_api.h>

#include <atlk/can.h>
#include <atlk/can_service.h>
#include <atlk/gnss.h>
#include <atlk/gnss_teseo.h>
#include <atlk/mib_service.h>
#include <atlk/mibs/nav-mib.h>

#include "poti-hil.h"

/*
  CRATON ThreadX Teseo SOU Integration Example

  This example demonstrates how to integrate Teseo DR with CAN sensor data
  via Sensor-Over-Uart (SOU) API.

  CAN data format used in this example comes from C2C-CC PoTi Test Fest
  event 2016, testing is done via Hardware In the Loop (HIL) simulation.

  A thread is created in which CAN data is received, parsed and fed to Teseo
  SOU API. Averaging, scaling, SOU message creation and writing is done via
  the SOU module.

  GNSS antenna offset is calculated in the NMEA layer. Published navigation
  position is the vehicle's reference position (in this example it is done
  according to EU standards).

  Vehicle reference position is different in US and EU standards, please
  refer to:

     - US: SAE J2745/1 - section 6.2.3.
     - EU: ETSI EN 302 637-2 - section B.19.

  Fot this example to actually work, Teseo needs to be flashed with DR
  firmware configured to the correct DR operating mode, sensor data units,
  and so on.

  A GNSS simulator with synchronized CAN playback capabilities can be used
  to test in the lab.
*/

/* Example thread priorities */
#define CAN_RECEIVE_THREAD_PRIORITY 40
#define TESEO_SOU_FEEDER_THREAD_PRIORITY 40

/* CAN device ID used in this example */
#define CAN_DEVICE_ID 0

/* GNSS antenna offset relative to vehicle reference point */
#define GNSS_ANTENNA_OFFSET_X_CM (-273L)
#define GNSS_ANTENNA_OFFSET_Y_CM 0L
#define GNSS_ANTENNA_OFFSET_Z_CM (-150L)

#if defined __CRATON_ARM

/* CAN receive thread */
static TX_THREAD can_receive_thread;
static uint8_t can_receive_thread_stack[0x1000];
static void can_receive_thread_entry(ULONG input);

/* CAN service */
static can_service_t *can_service = NULL;

/* CAN socket */
static can_socket_t *can_socket = NULL;

/* CAN ID filter array */
static const can_id_filter_t filter_array[] = {
  { .can_id = 0, .can_id_mask = 0 },
};

/* MIB service */
static mib_service_t *mib_service = NULL;

void craton_user_init(void)
{
  /* ThreadX return value */
  ULONG trv = TX_SUCCESS;
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* CAN socket configuration */
  can_socket_config_t socket_config = CAN_SOCKET_CONFIG_INIT;
  /* Teseo SOU configuration */
  gnss_teseo_sou_config_t sou_config =
      GNSS_TESEO_SOU_CONFIG_INIT;

  /* Get default MIB service instance */
  rc = mib_default_service_get(&mib_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "mib_default_service_get: %s\n", atlk_rc_to_str(rc));
    return;
  }

  /* Set GNSS antenna offset X (positive towards vehicles front) */
  rc = mib_set_navGnssAntennaOffsetX(mib_service, GNSS_ANTENNA_OFFSET_X_CM);
  if (atlk_error(rc)) {
    fprintf(stderr, "mib_set_navGnssAntennaOffsetX: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Set GNSS antenna offset Y (positive towards vehicles right hand side) */
  rc = mib_set_navGnssAntennaOffsetY(mib_service, GNSS_ANTENNA_OFFSET_Y_CM);
  if (atlk_error(rc)) {
    fprintf(stderr, "mib_set_navGnssAntennaOffsetY: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Set GNSS antenna offset Z (positive towards ground) */
  rc = mib_set_navGnssAntennaOffsetZ(mib_service, GNSS_ANTENNA_OFFSET_Z_CM);
  if (atlk_error(rc)) {
    fprintf(stderr, "mib_set_navGnssAntennaOffsetZ: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Get default CAN service instance */
  rc = can_default_service_get(&can_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "can_default_service_get: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Set socket configuration */
  socket_config.filter_array_ptr = filter_array;
  socket_config.filter_array_size = 1;
  socket_config.device_id = CAN_DEVICE_ID;

  /* Create CAN socket */
  rc = can_socket_create(can_service, &can_socket, &socket_config);
  if (atlk_error(rc)) {
    fprintf(stderr, "can_socket_create: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Create CAN receive thread */
  trv = tx_thread_create(&can_receive_thread, "can_receive_thread",
                         can_receive_thread_entry, 0,
                         can_receive_thread_stack,
                         sizeof(can_receive_thread_stack),
                         CAN_RECEIVE_THREAD_PRIORITY,
                         CAN_RECEIVE_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  /* Set Teseo SOU configuration */
  sou_config.operating_mode = GNSS_TESEO_SOU_OPERATING_MODE_20
      ;
  sou_config.gyro_1axis_params_ptr =
    &poti_hil_vehicle_motion_1_gyro_1axis_params;
  sou_config.wheels_speed_params_ptr =
    &poti_hil_vehicle_motion_2_wheel_speed_params;
  sou_config.sched_params.priority = TESEO_SOU_FEEDER_THREAD_PRIORITY;

  /* Initialize Teseo SOU */
  rc = gnss_teseo_sou_init(&sou_config);
  if (atlk_error(rc)) {
    fprintf(stderr, "gnss_teseo_sou_init: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  return;

error:
  /* Release allocated resources */
  can_socket_delete(can_socket);
  can_service_delete(can_service);
  mib_service_delete(mib_service);
}

static void can_receive_thread_entry(ULONG input)
{
  /* Current reverse gear status */
  int reverse_gear_status = 0;
  /* Previous reverse gear status */
  int prev_reverse_gear_status = 0;
  /* Not using input */
  (void)input;

  while (1) {
    /* Autotalks return code */
    atlk_rc_t rc = ATLK_OK;
    /* Received CAN message data */
    uint8_t data[CAN_DATA_SIZE_MAX];
    /* Received CAN message data size */
    size_t data_size = sizeof(data);
    /* Received CAN ID */
    can_id_t can_id;

    /* Receive CAN message */
    rc = can_receive(can_socket, data, &data_size, &can_id,
                     &atlk_wait_forever);
    if (atlk_error(rc)) {
      fprintf(stderr, "can_receive: %s\n", atlk_rc_to_str(rc));
      continue;
    }

    switch (can_id) {
    case POTI_HIL_CAN_ID_VEHICLE_STATE:
    {
      /* Extract reverse gear status from CAN message */
      reverse_gear_status =
        poti_hil_vehicle_state_reverse_gear_status_get(data, data_size);
      if (reverse_gear_status == -1) {
        /* On failure, use previous reverse gear status */
        reverse_gear_status = prev_reverse_gear_status;
      }

      /* Feed reverse gear status to Teseo SOU */
      rc = gnss_teseo_sou_reverse_gear_data_feed(reverse_gear_status);
      if (atlk_error(rc)) {
        /* Ignore failures */
      }
      break;
    }
    case POTI_HIL_CAN_ID_VEHICLE_MOTION_1:
    {
      /* Extract gyro 1-axis status from CAN message */
      sensor_value_t gyro_1axis =
        poti_hil_vehicle_motion_1_gyro_1axis_get(data, data_size);

      /* Feed gyro 1-axis to Teseo SOU */
      rc = gnss_teseo_sou_gyro_1axis_data_feed(gyro_1axis);
      if (atlk_error(rc)) {
        /* Ignore failures */
      }
      break;
    }
    case POTI_HIL_CAN_ID_VEHICLE_MOTION_2:
    {
      /* Extract wheels speed from CAN message */
      sensor_wheels_speed_t wheels_speed =
        poti_hil_vehicle_motion_2_wheels_speed_get(data, data_size);

      /* Feed wheels speed to Teseo SOU */
      rc = gnss_teseo_sou_wheels_speed_data_feed(wheels_speed);
      if (atlk_error(rc)) {
        /* Ignore failures */
      }
      break;
    }
    case POTI_HIL_CAN_ID_LATERAL_STATE:
    default:
      break;
    }

    if (reverse_gear_status != prev_reverse_gear_status) {
      /* If reverse gear is enabled, revese antenna offset X */
      int32_t antenna_offset_x_cm = (reverse_gear_status ?
        GNSS_ANTENNA_OFFSET_X_CM : -GNSS_ANTENNA_OFFSET_X_CM);

      /* Set GNSS antenna offset X (positive towards vehicles front) */
      rc = mib_set_navGnssAntennaOffsetX(mib_service, antenna_offset_x_cm);
      if (atlk_error(rc)) {
        fprintf(stderr, "mib_set_navGnssAntennaOffsetX: %s\n",
                atlk_rc_to_str(rc));
      }

      /* Set previous reverse gear status to current */
      prev_reverse_gear_status = reverse_gear_status;
    }
  }
}

#else /* __CRATON_ARM */

void craton_user_init(void)
{
}

#endif /* __CRATON_ARM */
```

8.19 craton-threadx/gnss-teseo/poti-hil.c

```
/* Copyright (C) 2015-2016 Autotalks Ltd. */
#include <inttypes.h>
#include <string.h>
#include <assert.h>

#include <craton/syslog.h>

#include "poti-hil.h"

int
poti_hil_vehicle_state_reverse_gear_status_get(const uint8_t *data_ptr,
            size_t data_size)
{
  int reverse_gear_status = 0;
  uint8_t gear = 0;

  assert(data_ptr);

  if (atlk_unlikely(data_size != 8)) {
    syslog(LOG_ERR, "Unexpecetd data size: %zu", data_size);
    return -1;
  }
  gear = (data_ptr[3] & 0xf0) >> 4;
  reverse_gear_status = (gear == 1) ? 1 : 0;

  syslog(LOG_DEBUG, "Got reverse gear status: %d", reverse_gear_status);

  return reverse_gear_status;
}

const sensor_value_params_t
poti_hil_vehicle_motion_1_gyro_1axis_params = {
  .units = SENSOR_UNITS_RADPS,
  .min = -2048,                     /* = -4.096 / 0.002 */
  .max = 2048,                      /* = 4.096 / 0.002 */
  .inverse_scaling = 500           /* = 1 / 0.002 */
};

sensor_value_t
poti_hil_vehicle_motion_1_gyro_1axis_get(const uint8_t *data_ptr,
            size_t data_size)
{
  sensor_value_t gyro_1axis = SENSOR_VALUE_NA;

  assert(data_ptr);

  if (atlk_unlikely(data_size != 8)) {
    syslog(LOG_ERR, "Unexpecetd data size: %zu", data_size);
    return gyro_1axis;
  }

  gyro_1axis = ((uint32_t)data_ptr[5] | ((uint32_t)data_ptr[6] << 8)) & 0xfff;
  if (gyro_1axis & (1 << 11)) {
    gyro_1axis |= 0xfffff000;
  }
  gyro_1axis *= -1;

  syslog(LOG_DEBUG, "Got gyro 1-axis: %" PRIi32, gyro_1axis);

  return gyro_1axis;
}

const sensor_value_params_t
poti_hil_vehicle_motion_2_wheel_speed_params = {
  .units = SENSOR_UNITS_MPS,
  .min = -32768,                    /* = -327.68 / 0.01 */
  .max = 32768,                     /* = 327.68 / 0.01 */
  .inverse_scaling = 100           /* = 1 / 0.01 */
};

sensor_wheels_speed_t
poti_hil_vehicle_motion_2_wheels_speed_get(const uint8_t *data_ptr,
            size_t data_size)
{
  sensor_wheels_speed_t wheels_speed =
      SENSOR_WHEELS_SPEED_INIT;
  int16_t speed = 0;

  assert(data_ptr);

  if (atlk_unlikely(data_size != 8)) {
    syslog(LOG_ERR, "Unexpecetd data size: %zu", data_size);
    return wheels_speed;
  }

  memcpy(&speed, &data_ptr[0], sizeof(speed));
  wheels_speed.front_left = speed;

  memcpy(&speed, &data_ptr[2], sizeof(speed));
  wheels_speed.front_right = speed;

  memcpy(&speed, &data_ptr[4], sizeof(speed));
  wheels_speed.rear_left = speed;

  memcpy(&speed, &data_ptr[6], sizeof(speed));
  wheels_speed.rear_right = speed;

  syslog(LOG_DEBUG, "Got wheels speed FL: %" PRIi32 ", FR: %"
         PRIi32 ", RL: %" PRIi32 ", RR: %" PRIi32,
         wheels_speed.front_left, wheels_speed.front_right,
         wheels_speed.rear_left, wheels_speed.rear_right);

  return wheels_speed;
}
```

8.20 craton-threadx/gnss-teseo/poti-hil.h

```
/* Copyright (C) 2015-2016 Autotalks Ltd. */
```

```
#ifndef _POTI_HIL_H
#define _POTI_HIL_H

#include <atlk/sdk.h>
#include <atlk/sensor.h>

#ifdef __cplusplus
extern "C" {
#endif

typedef enum {
  POTI_HIL_CAN_ID_LATERAL_STATE = 0x106,

  POTI_HIL_CAN_ID_VEHICLE_STATE = 0x110,

  POTI_HIL_CAN_ID_VEHICLE_MOTION_1 = 0x120,

  POTI_HIL_CAN_ID_VEHICLE_MOTION_2 = 0x121

} poti_hil_can_id_t;

int
poti_hil_vehicle_state_reverse_gear_status_get(const uint8_t *data_ptr,
            size_t data_size);

extern const sensor_value_params_t
poti_hil_vehicle_motion_1_gyro_1axis_params;

sensor_value_t
poti_hil_vehicle_motion_1_gyro_1axis_get(const uint8_t *data_ptr,
            size_t data_size);

extern const sensor_value_params_t
poti_hil_vehicle_motion_2_wheel_speed_params;

sensor_wheels_speed_t
poti_hil_vehicle_motion_2_wheels_speed_get(const uint8_t *data_ptr,
            size_t data_size);

#ifdef __cplusplus
}
#endif

#endif /* _POTI_HIL_H */
```

8.21 craton-threadx/gnss/gnss-integration-example.c

```
/* Copyright (C) 2015 Autotalks Ltd. */
#include <errno.h>
#include <stdio.h>
#include <assert.h>
#include <unistd.h>
#include <fcntl.h>

#include <tx_api.h>

#include <atlk/nav.h>
#include <atlk/nav_service.h>

#include <atlk/mib_service.h>
#include <atlk/mibs/nav-mib.h>

#include <craton/uart_driver.h>

/*
  CRATON ThreadX GNSS Integration Example

  This example demonstrates how to integrate an arbitrary GNSS device with
  the navigation sub-system of the Autotalks SDK.


  The function gnss_poll() is a "place holder" for user NMEA (or binary data)
  parsing code. The function's implementation is expected to do the following:

    1. Read from passed file descriptor.
    2. Parse standard and relevant proprietary NMEA messages (or proprietary
       binary data when relevant).
    3. Fill passed struct nav_fix_t and return the OK return code.

  In case of errors, user should return appropriate return code indicating
  error (or handle it in whichever way he sees fit).


  The navigation sub-system includes a component called "nav-update". This
  component is a subscriber of navigation fixes and is in charge of the
  following:
    1. Updating system time.
    2. Updating MIB attribute navFixAvailable.
    3. Updating MIB attribute navSysTimeLeapSeconds.

  Updating of the MIB attribute navSysTimeLeapSeconds is especially important
  for proper functionality of system time; specifically, it is vital for
  gettimeofday() functionality.

  To update this MIB attribute, the field leap_seconds_since_2004 is expected
  to be filled in nav_fix_t.

  If this is not possible, it is recommended to get the value of MIB attribute
  navSysTimeLeapSeconds during init and fill this field with this fixed value.
  Note that currently the value returned (which is hard-coded) is only correct
  from July 1st 2015 and until the next leap second event occurs.


  This example assumes the following:

    1. The firmware is initialized in 'gnss' mode.
    2. GNSS device UART is connected to CRATON's 2nd UART device.
    3. GNSS device PPS output is connected to CRATON's PPS input.
*/

/* GNSS thread priority */
#define GNSS_THREAD_PRIORITY 40

#if defined __CRATON_NO_ARC

/* GNSS thread */
static TX_THREAD gnss_thread;
static uint8_t gnss_thread_stack[0x2000];
static void gnss_thread_entry(ULONG input);

/* GNSS UART device ID */
#define UART_DEVICE_ID 1

/* GNSS UART device path */
#define UART_DEVICE_PATH "/dev/uart1"

/* GNSS UART device speed */
#define UART_SPEED_BPS UART_SPEED_230400_BPS

void craton_user_init(void)
{
  /* MIB service */
  mib_service_t *mib_service = NULL;
  /* ThreadX return value */
  ULONG trv = TX_SUCCESS;
  /* Autotalks return code*/
  atlk_rc_t rc = ATLK_OK;

  /* Get MIB service */
  rc = mib_default_service_get(&mib_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "mib_default_service_get: %s\n", atlk_rc_to_str(rc));
    return;
  }

  /* Set navigation data source to 'none' */
  rc = mib_set_navDataSource(mib_service, MIB_navDataSource_none);
  if (atlk_error(rc)) {
    fprintf(stderr, "mib_set_navDataSource: %s\n", atlk_rc_to_str(rc));
    goto out;
  }

  /* Create GNSS thread */
  trv = tx_thread_create(&gnss_thread, "gnss_thread",
                         gnss_thread_entry, 0,
                         gnss_thread_stack,
                         sizeof(gnss_thread_stack),
                         GNSS_THREAD_PRIORITY,
                         GNSS_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

out:
  /* Delete MIB service */
  mib_service_delete(mib_service);

  return;
}

/* GNSS polling function */
static atlk_rc_t atlk_must_check gnss_poll(int fd,
      nav_fix_t *nav_fix)
```

```
{
  /* Avoid 'unused parameter' compilation errors */
  (void)fd;
  (void)nav_fix;

  /*
     Place holder for user code (see description at top of page).

     Sleeping 50ms to simulate the behavior of a GNSS device with update
     rate of 20Hz (as well as to avoid a busy-loop in this example).
  */
  usleep(50 * 1000);

  return ATLK_OK;
}

/* GNSS thread entry */
static void gnss_thread_entry(ULONG input)
{
  /* Not using input */
  (void)input;
  /* Navigation service */
  nav_service_t *nav_service = NULL;
  /* File descriptor */
  int fd = -1;
  /* Autotalks return code*/
  atlk_rc_t rc = ATLK_OK;

  /* Open file descriptor for UART device */
  fd = open(UART_DEVICE_PATH, 0);
  if (fd < 0) {
    fprintf(stderr, "open failed, errno %d\n", errno);
    return;
  }

  /* Set UART device speed */
  rc = uart_speed_set(UART_DEVICE_ID, UART_SPEED_BPS);
  if (atlk_error(rc)) {
    fprintf(stderr, "uart_speed_set: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Sleep 10ms to let UART device speed settle */
  usleep(10 * 1000);

  /* Get default navigation service instance */
  rc = nav_default_service_get(&nav_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "nav_default_service_get: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  while(1) {
    /* Navigation data fix */
    nav_fix_t nav_fix = NAV_FIX_INIT;

    /* Poll GNSS device for next navigation data fix */
    rc = gnss_poll(fd, &nav_fix);
    if (atlk_error(rc)) {
      fprintf(stderr, "gnss_poll: %s\n", atlk_rc_to_str(rc));
      continue;
    }

    /* Publish navigation data fix to all subscribers */
    rc = nav_fix_publish(nav_service, &nav_fix);
    if (atlk_error(rc)) {
      fprintf(stderr, "nav_fix_publish: %s\n", atlk_rc_to_str(rc));
      continue;
    }
  }

error:
  /* Close file descriptor */
  close(fd);

  return;
}

#else /* __CRATON_NO_ARC */

void craton_user_init(void)
{
}

#endif /* __CRATON_NO_ARC */
```

8.22 craton-threadx/i2s/i2s-example.c

```
/* Copyright (C) 2014-2015 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <math.h>

#include <tx_api.h>

#include <craton/i2s_driver.h>
#include <craton/cache.h>

/* Example sample rate */
#define EXAMPLE_SAMPLE_RATE 20e3

/* DMA playback structure */
static i2s_dma_playback_t i2s_dma_playback =
      I2S_DMA_PLAYBACK_INIT;

/* Replay completion semaphore */
static TX_SEMAPHORE semaphore;

/* Audio input from example sine buffer */
static int8_t i2s_buf[1 << 16] cache_line_aligned;

/* Generate sound samples with a sine wave whose frequency changes from
   zero to 1KHz every second.
*/
static void
generate_sine(int8_t *samples, size_t num_samples)
{
  uint32_t i;

  for (i = 0; i < num_samples; i++) {
    double t0 = (double)i / EXAMPLE_SAMPLE_RATE;
    double t1 = sin(t0 * M_PI / 2) * 500;
    samples[i] = (int8_t)(sin((t1 * 2.0 * M_PI)) * INT8_MAX);
  }
}

static void
i2s_playback_done(i2s_dma_playback_t *playback)
{
  /* ThreadX return value */
  ULONG trv = TX_SUCCESS;
  /* Not using parameter */
  (void)playback;

  trv = tx_semaphore_put(&semaphore);
  assert(trv == TX_SUCCESS);
}

void craton_user_init(void)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* ThreadX return value */
  ULONG trv = TX_SUCCESS;

  /* Create a semaphore */
  trv = tx_semaphore_create(&semaphore, "i2s_example_semaphore", 0);
  assert(trv == TX_SUCCESS);

  /* Generate an audio sample buffer */
  generate_sine(i2s_buf, sizeof(i2s_buf));

  /* Set both stereo channels to play the same sample buffer */
  i2s_dma_playback.left_sample_buffer_ptr = i2s_buf;
  i2s_dma_playback.right_sample_buffer_ptr = i2s_buf;
  i2s_dma_playback.sample_buffer_size = sizeof(i2s_buf);

  /* Set i2s_playback_done() as the completion handler */
  i2s_dma_playback.completion_handler = i2s_playback_done;

  /* Start playback */
  rc = i2s_dma_playback_start(&i2s_dma_playback);
  if (atlk_error(rc)) {
    fprintf(stderr, "i2s_dma_playback_start: %s\n", atlk_rc_to_str(rc));
    return;
  }

  /* Wait for the operation to complete */
  trv = tx_semaphore_get(&semaphore, TX_WAIT_FOREVER);
  assert(trv == TX_SUCCESS);

  printf("I2S playback complete, status: %d\n",
         i2s_dma_playback.playback_status);
}
```

8.23 craton-threadx/imq/imq-client.c

```
/* Copyright (C) 2014-2015 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>

#include <tx_api.h>

#include <craton/imq.h>

#include "imq-echo-server.h"

/*
  CRATON ThreadX IMQ Example

  This example demonstrates basic usage of IMQ API for code running on
  top of CRATON processor with ThreadX RTOS.

  This file implements an IMQ client which runs on ARC1 CPU.

  A thread is created in which a socket is connected; IMQ messages are sent
  and then received in a loop. Messages are compared and are expected to be
  identical.
*/

#ifdef __CRATON_ARC1

/* IMQ client thread priority */
#define IMQ_CLIENT_THREAD_PRIORITY 30

/* IMQ client thread */
static TX_THREAD imq_client_thread;
static uint8_t imq_client_thread_stack[0x2000];
static void imq_client_thread_entry(ULONG input);

/* IMQ client data socket */
static imq_socket_t data_socket = IMQ_SOCKET_INIT;

/* Example message format string: Example <seq_num> */
static const char example_msg_fmt[] = "Example %" PRIu32;

/* Cleanup any allocated resources */
static void example_cleanup(void)
{
  imq_close(&data_socket);
}

void craton_user_init(void)
{
  /* ThreadX return value */
  ULONG trv = TX_SUCCESS;

  /* Create IMQ client thread */
  trv = tx_thread_create(&imq_client_thread, "imq_client_thread",
                         imq_client_thread_entry, 0,
                         imq_client_thread_stack,
                         sizeof(imq_client_thread_stack),
                         IMQ_CLIENT_THREAD_PRIORITY,
                         IMQ_CLIENT_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  return;
}

static void imq_client_thread_entry(ULONG input)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* Message counter */
  uint32_t msg_count = 0;
  /* Not using input */
  (void)input;

  /* Connect to IMQ echo server */
  rc = imq_connect(&data_socket, IMQ_ECHO_SERVER_ADDRESS, &
      atlk_wait_forever);
  if (atlk_error(rc)) {
    fprintf(stderr, "imq_connect: %s\n", atlk_rc_to_str(rc));
    goto error;
  }
  while (1) {
    /* Receive buffer */
    char rx_buf[IMQ_ECHO_SERVER_QUEUE_MTU];
    /* Receive buffer size */
    size_t rx_size = IMQ_ECHO_SERVER_QUEUE_MTU;
    /* Send buffer */
    char tx_buf[IMQ_ECHO_SERVER_QUEUE_MTU];
    /* Send buffer size */
    size_t tx_size  = 1 + snprintf(tx_buf, sizeof(tx_buf),
                                   example_msg_fmt, msg_count);

    printf("Sending IMQ message: \"%s\"\n", tx_buf);

    /* Send IMQ message to echo server */
    rc = imq_send(&data_socket, tx_buf, tx_size, NULL);
    if (atlk_error(rc)) {
      fprintf(stderr, "imq_send: %s\n", atlk_rc_to_str(rc));
      goto error;
    }

    /* Receive IMQ message from echo server */
    rc = imq_receive(&data_socket, rx_buf, &rx_size, &
      atlk_wait_forever);
    if (atlk_error(rc)) {
      fprintf(stderr, "imq_receive: %s\n", atlk_rc_to_str(rc));
      goto error;
    }

    /* Compare received and sent messages */
    if (rx_size == tx_size && strncmp(tx_buf, rx_buf, rx_size) == 0) {
      printf("Echo message \"%s\" received successfully.\n", rx_buf);
    }
    else {
      fprintf(stderr, "Error! Messages differ.\n");
      goto error;
    }

    /* Increment message counter */
    msg_count++;

    /* Sleep 100ms between transmissions */
    usleep(100000);
  }

error:
  example_cleanup();
}

#else /* __CRATON_ARC1 */

void craton_user_init(void)
{
}

#endif /* __CRATON_ARC1 */
```

8.24 craton-threadx/imq/imq-echo-server.c

```
/* Copyright (C) 2014-2015 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>

#include <tx_api.h>

#include <craton/imq.h>

#include "imq-echo-server.h"

/*
  CRATON ThreadX IMQ Example

  This example demonstrates basic usage of IMQ API for code running on
  top of CRATON processor with ThreadX RTOS.

  This file implements an IMQ echo server which runs on the ARM CPU.

  An IMQ service socket is bound-to and listened upon. A thread is created in
  which a connection is accepted; IMQ messages are received in a loop and sent
  back to their origin (i.e. echoed back).
*/

/* IMQ echo server thread priority */
#define IMQ_ECHO_SERVER_THREAD_PRIORITY 60
```

```c
/* IMQ echo server thread */
static TX_THREAD imq_echo_server_thread;
static uint8_t imq_echo_server_thread_stack[0x2000];
static void imq_echo_server_thread_entry(ULONG input);

/* IMQ echo server service socket */
static imq_socket_t service_socket = IMQ_SOCKET_INIT;

/* IMQ echo server data socket */
static imq_socket_t data_socket = IMQ_SOCKET_INIT;

/* Cleanup any allocated resources */
static void example_cleanup(void)
{
  imq_close(&data_socket);
  imq_close(&service_socket);
}

void craton_user_init(void)
{
  /* ThreadX return value */
  ULONG trv = TX_SUCCESS;
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* IMQ service configuration */
  imq_service_config_t config = IMQ_SERVICE_CONFIG_INIT;

  /* Bind IMQ echo server socket */
  rc = imq_bind(&service_socket, IMQ_ECHO_SERVER_ADDRESS);
  if (atlk_error(rc)) {
    fprintf(stderr, "imq_bind: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Set socket configuration parameters */
  config.server_to_client_config.queue_mtu = IMQ_ECHO_SERVER_QUEUE_MTU;
  config.server_to_client_config.queue_length =
      IMQ_ECHO_SERVER_QUEUE_LENGTH;
  config.client_to_server_config.queue_mtu = IMQ_ECHO_SERVER_QUEUE_MTU;
  config.client_to_server_config.queue_length =
      IMQ_ECHO_SERVER_QUEUE_LENGTH;

  /* Give the socket a descriptive name */
  config.service_name = "imq_echo_server";

  /* Listen on IMQ echo server socket */
  rc = imq_listen(&service_socket, &config);
  if (atlk_error(rc)) {
    fprintf(stderr, "imq_listen: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Create IMQ echo server thread */
  trv = tx_thread_create(&imq_echo_server_thread, "imq_echo_server_thread",
                         imq_echo_server_thread_entry, 0,
                         imq_echo_server_thread_stack,
                         sizeof(imq_echo_server_thread_stack),
                         IMQ_ECHO_SERVER_THREAD_PRIORITY,
                         IMQ_ECHO_SERVER_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  return;

error:
  example_cleanup();
}

static void imq_echo_server_thread_entry(ULONG input)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* Not using input */
  (void)input;

  /* Accept an IMQ connection */
  rc = imq_accept(&service_socket, &data_socket, &atlk_wait_forever);
  if (atlk_error(rc)) {
    fprintf(stderr, "imq_accept: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  printf("IMQ echo server started...\n");

  while (1) {
    /* Receive/send buffer */
    char buf[IMQ_ECHO_SERVER_QUEUE_MTU];
    /* Receive/send size */
    size_t size = sizeof(buf);

    /* Receive a IMQ message */
    rc = imq_receive(&data_socket, buf, &size, &atlk_wait_forever);
    if (atlk_error(rc)) {
      fprintf(stderr, "imq_receive: %s\n", atlk_rc_to_str(rc));
      goto error;
    }

    /* Send IMQ echo message */
    rc = imq_send(&data_socket, buf, size, NULL);
    if (atlk_error(rc)) {
      fprintf(stderr, "imq_send: %s\n", atlk_rc_to_str(rc));
      goto error;
    }
  }

error:
  example_cleanup();
}
```

8.25 craton-threadx/imq/imq-echo-server.h

```
/* Copyright (C) 2014-2015 Autotalks Ltd. */
#ifndef _IMQ_ECHO_SERVER_H
#define _IMQ_ECHO_SERVER_H

/*
  CRATON ThreadX IMQ Example

  Common declarations.
*/

/* IMQ echo server socket queue MTU */
#define IMQ_ECHO_SERVER_QUEUE_MTU 64

/* IMQ echo server socket queue length */
#define IMQ_ECHO_SERVER_QUEUE_LENGTH 128

/* IMQ echo server address */
#define IMQ_ECHO_SERVER_ADDRESS 7

#endif /* _IMQ_ECHO_SERVER_H */
```

8.26 craton-threadx/mibs/mibs-edca-example.c

```
/* Copyright (C) 2015 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>

#include <tx_api.h>

#include <atlk/mib_service.h>

#include <atlk/mibs/wlan-mib.h>

/*
  CRATON ThreadX EDCA Table MIB Example

  This example demonstrates basic usage of the MIB API for code running on
  top of CRATON processor with ThreadX RTOS.

  @todo: Currently v2x_send_params_t.user_priority is not supported. All
  frames are sent at access category AC_VO. Therefore, changing EDCA parameters
  of other access categories will have no effect.
*/

/* Access category: Best effort */
#define AC_BE 0

/* Access category: Background */
#define AC_BK 1

/* Access category: Video */
#define AC_VI 2

/* Access category: Voice */
#define AC_VO 3

/* Create EDCA table index from interface index and access category */
```

```
#define EDCA_INDEX(if_index, ac) ((if_index) * 4 + (ac) - 3)

void craton_user_init(void)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* MIB service */
  mib_service_t *service = NULL;
  /* EDCA CWmax value */
  uint32_t value;

  /* Get default MIB service instance */
  rc = mib_default_service_get(&service);
  if (atlk_error(rc)) {
    fprintf(stderr, "mib_default_service_get: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Set CWmin of access category AC_BE at interface 1 to 15 */
  rc = mib_set_wlanEdcaCWmin(service, EDCA_INDEX(1, AC_BE), 15);
  if (atlk_error(rc)) {
    fprintf(stderr, "mib_set_wlanEdcaCWmin: %s\n", atlk_rc_to_str(rc));
    goto error;
  }
  printf("CWmin of access category AC_BE at interface 1 set to 15.\n");

  /* Get CWmax of access category AC_VO at interface 2 */
  rc = mib_get_wlanEdcaCWmax(service, EDCA_INDEX(2, AC_VO), &value);
  if (atlk_error(rc)) {
    fprintf(stderr, "mib_get_wlanEdcaCWmax: %s\n", atlk_rc_to_str(rc));
    goto error;
  }
  printf("CWmax of access category AC_VO at interface 2 is %lu\n", value);

  return;

error:
  mib_service_delete(service);
}
```

8.27 craton-threadx/mibs/mibs-example.c

```
/* Copyright (C) 2013-2015 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>

#include <tx_api.h>

#include <atlk/mib_service.h>

#include <atlk/mibs/wlan-mib.h>
#include <atlk/mibs/snmpv2-mib.h>

/*
  CRATON ThreadX MIBs Example

  This example demonstrates basic usage of the MIB API for code running on
  top of CRATON processor with ThreadX RTOS.

  The MIB API mirrors Autotalks proprietary MIBs as well as selected MIB
  attributes from standard MIBs.

  The example demonstrates how to set the frequency of interface 1 to 5880
  MHz using WLAN MIB API (which mirrors AUTOTALKS-WLAN-MIB.mib) and how to
  get the system description via SNMPv2 MIB API (which mirrors the standard
  SNMPv2-MIB.mib).
*/

/* Size of system description string in bytes used in this example */
#define EXAMPLE_SYS_DESCR_SIZE 100

void craton_user_init(void)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* System description string */
  char sys_descr[EXAMPLE_SYS_DESCR_SIZE];
  /* Size of description string in bytes */
  size_t sys_descr_size = sizeof(sys_descr);
  /* MIB service */
  mib_service_t *service = NULL;

  /* Get default MIB service instance */
  rc = mib_default_service_get(&service);
  if (atlk_error(rc)) {
    fprintf(stderr, "mib_default_service_get: %s\n", atlk_rc_to_str(rc));
    goto exit;
  }

  /* Set frequency at interface 1 to 5880 MHz */
  rc = mib_set_wlanFrequency(service, 1, 5880);
  if (atlk_error(rc)) {
    fprintf(stderr, "mib_set_wlanFrequency: %s\n", atlk_rc_to_str(rc));
    goto exit;
  }
  printf("Frequency at interface 1 set to 5880 MHz.\n");

  /* Get system description */
  rc = mib_get_sysDescr(service, sys_descr, &sys_descr_size);
  if (atlk_error(rc)) {
    fprintf(stderr, "mib_get_sysDescr: %s\n", atlk_rc_to_str(rc));
    goto exit;
  }
  printf("System description: %s\n", sys_descr);

exit:
  /* Clean-up resources */
  mib_service_delete(service);

  return;
}
```

8.28 craton-threadx/nav/nav-data-example.c

```
/* Copyright (C) 2016 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>

#include <tx_api.h>

#include <atlk/nav_service.h>

#include "nav-trace.h"

/*
  CRATON ThreadX Navigation Data Example

  This example demonstrates basic usage of Navigation API for code running on
  top of CRATON processor with ThreadX RTOS.

  A single threads is created in which to receive navigation dataes and
  satellite reports. A navigation service is retrieved and a navigation data
  subscriber is created.

  Data is received in a loop with a non-blocking call to nav_data_receive.
  Received data is traced.
*/

/* Example thread priority */
#define EXAMPLE_THREAD_PRIORITY 40

/* Example thread */
static TX_THREAD example_thread;
static uint8_t example_thread_stack[0x1000];
static void example_thread_entry(ULONG input);

/* Navigation service */
static nav_service_t *example_nav_service = NULL;

/* Navigation data subscriber */
static nav_data_subscriber_t *example_nav_data_subscriber = NULL;

/* Cleanup any allocated resources */
static void example_cleanup(void)
{
  nav_data_subscriber_delete(example_nav_data_subscriber);
  nav_service_delete(example_nav_service);
}

void craton_user_init(void)
{
  /* Navigation data subscription mask */
  uint32_t data_mask = 0;
  /* ThreadX return value */
  ULONG trv = TX_SUCCESS;
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* Get default navigation service instance */
  rc = nav_default_service_get(&example_nav_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "nav_default_service_get: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Subscribe to fix and satellite report data frames */
  data_mask = NAV_DATA_TYPE_FIX |
      NAV_DATA_TYPE_SATELLITE_REPORT;

  /* Create a navigation data subscriber */
  rc = nav_data_subscriber_create(example_nav_service,
                                  data_mask,
                                  &example_nav_data_subscriber);
  if (atlk_error(rc)) {
    fprintf(stderr, "nav_data_subscriber_create: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Create example thread */
  trv = tx_thread_create(&example_thread, "example_thread",
                         example_thread_entry, 0,
                         example_thread_stack,
                         sizeof(example_thread_stack),
                         EXAMPLE_THREAD_PRIORITY,
                         EXAMPLE_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  return;

error:
  example_cleanup();
}

void example_thread_entry(ULONG input)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* Navigation data data */
  nav_data_t nav_data = NAV_DATA_INIT;
  /* Not using input */
  (void)input;

  while (1) {
    /* Receive a navigation data */
    rc = nav_data_receive(example_nav_data_subscriber, &nav_data,
                          &atlk_wait_forever);
    if (atlk_error(rc)) {
      /* Unexpected error occurred */
      fprintf(stderr, "nav_data_receive: %s\n", atlk_rc_to_str(rc));
      goto error;
    }

    /* Trace navigation data data content */
    if (nav_data.data_type == NAV_DATA_TYPE_FIX) {
      nav_trace_fix(&nav_data.data.fix);
    }
    else if (nav_data.data_type == NAV_DATA_TYPE_SATELLITE_REPORT) {
      nav_trace_satellite_report(&nav_data.data.satellite_report);
    }
  }

error:
  example_cleanup();
}
```

8.29 craton-threadx/nav/nav-example.c

```
/* Copyright (C) 2014-2015 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>

#include <tx_api.h>

#include <atlk/nav_service.h>

#include "nav-trace.h"

/*
  CRATON ThreadX Navigation Fix Example

  This example demonstrates basic usage of Navigation API for code running on
  top of CRATON processor with ThreadX RTOS.

  A single threads is created in which to receive navigation data fixes. A
  navigation service is retrieved and a navigation fix subscriber is created.

  Fixes are received in a loop with a blocking call to nav_fix_receive.
  Received fixes are traced.
*/

/* Example thread priority */
#define EXAMPLE_THREAD_PRIORITY 40

/* Example thread */
static TX_THREAD example_thread;
static uint8_t example_thread_stack[0x1000];
static void example_thread_entry(ULONG input);

/* Navigation service */
static nav_service_t *example_nav_service = NULL;

/* Navigation fix subscriber */
static nav_fix_subscriber_t *example_nav_fix_subscriber = NULL;

/* Cleanup any allocated resources */
static void example_cleanup(void)
{
  nav_fix_subscriber_delete(example_nav_fix_subscriber);
  nav_service_delete(example_nav_service);
}

void craton_user_init(void)
{
  /* ThreadX return value */
  ULONG trv = TX_SUCCESS;
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;

  /* Get default navigation service instance */
  rc = nav_default_service_get(&example_nav_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "nav_default_service_get: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Create a navigation fix subscriber */
  rc = nav_fix_subscriber_create(example_nav_service,
                                 &example_nav_fix_subscriber);
  if (atlk_error(rc)) {
    fprintf(stderr, "nav_fix_subscriber_create: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Create example thread */
  trv = tx_thread_create(&example_thread, "example_thread",
                         example_thread_entry, 0,
                         example_thread_stack,
                         sizeof(example_thread_stack),
                         EXAMPLE_THREAD_PRIORITY,
                         EXAMPLE_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  return;

error:
  example_cleanup();
}

void example_thread_entry(ULONG input)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* Navigation data fix */
  nav_fix_t nav_fix = NAV_FIX_INIT;
  /* Not using input */
  (void)input;

  while (1) {
    /* Receive a navigation fix */
    rc = nav_fix_receive(example_nav_fix_subscriber, &nav_fix,
                         &atlk_wait_forever);
    if (atlk_error(rc)) {
      /* Unexpected error occurred */
      fprintf(stderr, "nav_fix_receive: %s\n", atlk_rc_to_str(rc));
      goto error;
    }
    /* Trace navigation data fix content */
    nav_trace_fix(&nav_fix);
  }

error:
  example_cleanup();
}
```

8.30 craton-threadx/nav/nav-trace.h

```
/* Copyright (C) 2016 Autotalks Ltd. */
#ifndef _NAV_TRACE_H
#define _NAV_TRACE_H

#include <atlk/sdk.h>
#include <atlk/nav.h>

#include <craton/syslog.h>

/* Trace navigation data fix */
atlk_inline void
nav_trace_fix(const nav_fix_t *fix)
{
  syslog(LOG_INFO, "time: %.1f sec (err: %.4f sec), leap: %d sec%s",
         fix->time.tai_seconds_since_2004, fix->
      error_time_s,
         fix->time.leap_seconds_since_2004,
         fix->time.positive_leap_second ? " (positive leap second)" : "");

  syslog(LOG_INFO, "lat: %.7f deg, lon: %.7f deg, alt: %.1f m (err: %.1f m)",
         fix->position_latitude_deg, fix->
      position_longitude_deg,
         fix->position_altitude_m, fix->
      error_position_altitude_m);

  syslog(LOG_INFO, "err ellipse: hdg: %.1f deg, major len: %.1f m, "
         "minor len: %.1f m",
         fix->error_position_horizontal_major_axis_direction_deg
      ,
         fix->error_position_horizontal_semi_major_axis_length_m
      ,
         fix->error_position_horizontal_semi_minor_axis_length_m
      );

  syslog(LOG_INFO, "heading: %.2f deg (err: %.2f deg)",
         fix->movement_horizontal_direction_deg,
         fix->error_movement_horizontal_direction_deg);

  syslog(LOG_INFO, "speed: %.1f mps (err: %.1f mps), "
         "v-speed: %.1f mps (err: %.1f mps)",
         fix->movement_horizontal_speed_mps,
         fix->error_movement_horizontal_speed_mps,
         fix->movement_vertical_speed_mps,
         fix->error_movement_vertical_speed_mps);

  syslog(LOG_INFO, "mode: %d, data source: 0x%" PRIu32 ", hdop: %.2f",
         fix->mode, fix->data_source, fix->hdop);

  syslog(LOG_INFO, "sat in use: %d, GP in view: %d, GL in view: %d",
         fix->satellites_in_use_num,
         fix->satellites_num[NAV_SATELLITES_GPS],
         fix->satellites_num[NAV_SATELLITES_GLONASS]);
}

/* Trace navigation satellite report */
atlk_inline void
nav_trace_satellite_report(const nav_satellite_report_t *sat)
{
  syslog(LOG_INFO, "time: %.1f sec, leap: %d sec%s",
         sat->time.tai_seconds_since_2004,
         sat->time.leap_seconds_since_2004,
         sat->time.positive_leap_second ? " (positive leap second)" : "");
  for (size_t i = 0; i < sat->satellite_info_array_size; ++i) {
    syslog(LOG_INFO, "[%zd] %s prn: %u, elev: %u deg%s, azimuth %u deg%s, %u db%s",
           i + 1, (sat->satellite_info_array[i].
      satellite_system ==
                 NAV_SATELLITES_GPS) ? "GP" : "GL",
           sat->satellite_info_array[i].prn_num,
           sat->satellite_info_array[i].elevation_deg,
           (sat->satellite_info_array[i].elevation_deg ==
            NAV_SATELLITE_INFO_ELEVATION_DEG_NA) ? " (unknown)" : "",
           sat->satellite_info_array[i].azimuth_deg,
           (sat->satellite_info_array[i].azimuth_deg ==
            NAV_SATELLITE_INFO_AZIMUTH_DEG_NA) ? " (unknown)" : "",
           sat->satellite_info_array[i].cnr_db,
           (sat->satellite_info_array[i].cnr_db ==
            NAV_SATELLITE_INFO_CNR_DB_NA) ? " (not tracked)" : "");
  }
}

#endif /* _NAV_TRACE_H */
```

8.31 craton-threadx/nav/system-time-benchmark.c

```
/* Copyright (C) 2015 Autotalks Ltd. */
#include <stdio.h>
#include <stdint.h>
#include <unistd.h>

#include <time.h>
#include <sys/time.h>

#include <tx_api.h>

#include <atlk/sntp_client.h>
#include <atlk/mib_service.h>
#include <atlk/mibs/nav-mib.h>

/*
  CRATON System Time Benchmark

  This benchmark enables testing CRATON system time accuracy (set via GNSS
  source and synced with a GNSS device's 1PPS output) against a NTP-based
  reference time source using a SNTP client on CRATON.


  SNTP client configuration

  SNTP client is configured to a unicast polling interval of one second.
  Note that this is technically not allowed according to RFC-4330 (minimum
  allowed polling interval is 15 seconds). We do so under the assumption
  that the user is polling a local server (i.e. a server he has set up in his
  own Network). Please don't poll public NTP servers directly at this polling
  interval!

  The client is configured to a minimum server stratum of 3. Given that the
  user is expected to test against his own NTP server, his server is expected
  to  receive updates from a server of stratum 1 or 2 (stratum 2 servers are
  widely available; stratum 1 less so).

  Max root server dispersion is configured to 100000us. Although this seems
  high, most NTP servers tested against seem to have root dispersion values
  of this order. This value represents the maximal observed error between the
  NTP server and the root server; it does not mean that the error is actually
  100ms (disclaimer: this is my current understanding).


  NTP server configuration

  How to set-up a NTP server on Ubuntu Linux desktop is detailed at:

     https://help.ubuntu.com/lts/serverguide/NTP.html

  Tips:

  1. A list of NTP servers in your area is available at: http://www.pool.ntp.org
  2. Configuration:
      * Specifying a server as 'iburst' should speed up NTP time aquision.
      * Make sure to allow clients to synchronize with the server (this is
        disabled by default).
  3. Trouble-shooting:
      * Make sure to restart ntp after editing '/etc/ntp.conf'.
      * Initial NTP time acquisition can take ~minutes, please be patient.
      * Run 'ntpq -p' to see the list of NTP servers and their parameters.
      * Run 'ntpq -c rl' to see additional stats including root dispersion.
      * Make sure a firewall is not blocking NTP traffic.
*/

/* SNTP client thread priority */
#define SNTP_CLIENT_PRIORITY 20

/* IP address of NTP server used in this example */
#define SERVER_ADDRESS (10 << 24 | 10 << 16 | 1 << 8 | 110 << 0)

#ifdef __CRATON_ARM

/* MIB service */
static mib_service_t *mib_service = NULL;

/* Return a textual description for mib_navSysTimeStatus_t */
static const char *
sys_time_status_to_str(mib_navSysTimeStatus_t status)
```

```
{
  switch (status) {
  case MIB_navSysTimeStatus_notSet:
    return "not set";
  case MIB_navSysTimeStatus_set:
    return "set";
  default:
    return "unknown";
  }
}

/* Callback that will be invoked on every NTP update received */
static void
sntp_update_callback(const sntp_info_t *info)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* NTP time */
  double ntp_time = 0.0;
  /* System time */
  struct timeval sys_time;
  /* Whether fix is available */
  int fix_available = 0;
  /* System time status */
  mib_navSysTimeStatus_t sys_time_status = MIB_navSysTimeStatus_notSet;
  /* Difference between system time and NTP time */
  int64_t diff_us = 0;

  /* Get current system time */
  gettimeofday(&sys_time, NULL);

  /* Get whether fix is available */
  rc = mib_get_navFixAvailable(mib_service, &fix_available);
  if (atlk_error(rc)) {
    fprintf(stderr, "mib_get_navFixAvailable: %s\n", atlk_rc_to_str(rc));
    return;
  }

  /* Get system time status */
  rc = mib_get_navSysTimeStatus(mib_service, &sys_time_status);
  if (atlk_error(rc)) {
    fprintf(stderr, "mib_get_navSysTimeStatus: %s\n", atlk_rc_to_str(rc));
    return;
  }

  /* Calculate difference between system time and NTP time */
  ntp_time = sntp_time_to_posix_time(info);
  diff_us = (int64_t)sys_time.tv_sec * 1000000 +
    sys_time.tv_usec - ntp_time * 1000000.0;

  /* Print results */
  printf("Fix is %s, system time is %s, diff from NTP time is %lld us.\n",
         fix_available ? "available" : "not available",
         sys_time_status_to_str(sys_time_status), diff_us);
}

void craton_user_init(void)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* SNTP client configuration parameters */
  sntp_client_config_t config = SNTP_CLIENT_CONFIG_INIT;

  printf("Starting system time benchmark...\n");

  /* Get default MIB service instance */
  rc = mib_default_service_get(&mib_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "mib_default_service_get: %s\n", atlk_rc_to_str(rc));
    return;
  }

  /* Set SNTP client configuration parameters */
  config.sched_params.priority = SNTP_CLIENT_PRIORITY;
  config.update_handler = sntp_update_callback;
  config.ntp_server_address = SERVER_ADDRESS;
  config.type = SNTP_CONNECTION_TYPE_UNICAST;
  config.max_root_dispersion_us = 100000;
  config.min_server_stratum = 3;
  config.unicast_poll_interval_s = 1;

  /* Print filtering parameters used */
  printf("Rejecting NTP updates from NTP server that does not hold to:\n");
  printf("- Max root dispersion: %luus\n", config.max_root_dispersion_us);
  printf("- Min server stratum: %u\n", config.min_server_stratum);

  /* Initialize SNTP client */
  rc = sntp_client_init(&config);
  if (atlk_error(rc)) {
    fprintf(stderr, "sntp_client_init: %s\n", atlk_rc_to_str(rc));
    return;
  }

  printf("Waiting for NTP update...\n");
}

#else /*__CRATON_ARM */

void craton_user_init(void)
{
}

#endif /* __CRATON_ARM */
```

8.32 craton-threadx/net/http-example.c

```
/* Copyright (C) 2014-2015 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>

#include <tx_api.h>

#include <atlk/http_server.h>

/*
  CRATON HTTP Example

  This example demonstrates basic usage of HTTP for code running on
  top of CRATON processor with ThreadX RTOS.

  The example demosntrates registration of HTTP module with couple of
  module functions to be invoked once HTTP POST/GET is issued
*/

#define ARRAY_COUNT(ARR) (sizeof(ARR) / sizeof(ARR[0]))

#define HTTP_STATUS_OK 200

#define HTTP_SERVER_THREAD_PRIORITY 40

#define HTTP_SERVER_THREAD_TIME_SLICE 0

static void
print_info(const char *func_name,
           const char *resource,
           http_request_type_t request_type,
           const void *request_content,
           size_t request_content_size)
{
  printf("%s was called with resource %s\n", func_name, resource);
  printf("HTTP method is %d\n", request_type);
  if (request_content_size > 0) {
    size_t i;
    printf("HTTP request content is:\n");
    for (i = 0; i < request_content_size; i++) {
      printf("%c", ((char *)request_content)[i]);
    }
    printf("\n");
  }
  else {
    printf("No HTTP body content is available\n");
  }
}

atlk_rc_t
my_module_func1(const char *resource,
                http_request_type_t request_type,
                const void *request_content,
                size_t request_content_size,
                void *response_content,
                size_t *response_content_size,
                uint16_t *status_code)
{
  (void)response_content;

  print_info(__func__,
             resource,
             request_type,
             request_content,
             request_content_size);
  *status_code = HTTP_STATUS_OK;

  *response_content_size = 0;

  return ATLK_OK;
}

atlk_rc_t
my_module_func2(const char *resource,
                http_request_type_t request_type,
                const void *request_content,
                size_t request_content_size,
                void *response_content,
                size_t *response_content_size,
                uint16_t *status_code)
{
  (void)response_content;

  print_info(__func__,
             resource,
             request_type,
             request_content,
             request_content_size);

  *status_code = HTTP_STATUS_OK;

  *response_content_size = 0;

  return ATLK_OK;
}

static const http_url_entry_t example_entries[] = {
  {
    .url = "my_module_func1",
    .url_handler = my_module_func1,
  },
  {
    .url = "my_module_func2",
    .url_handler = my_module_func2,
  },
};

void craton_user_init(void)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;

  http_server_config_t config = HTTP_SERVER_CONFIG_INIT;

  config.default_path_prefix = NULL;

  /* Set HTTP server scheduling parameters */
  config.sched_params.priority = HTTP_SERVER_THREAD_PRIORITY;
  config.sched_params.time_slice = HTTP_SERVER_THREAD_TIME_SLICE;

  rc = http_server_init(&config);
  if (atlk_error(rc)) {
    fprintf(stderr, "http_server_module_register: %s\n", atlk_rc_to_str(rc));
    return;
  }

  rc = http_server_module_register("my_module",
                                   example_entries,
                                   ARRAY_COUNT(example_entries));
  if (atlk_error(rc)) {
    fprintf(stderr, "http_server_module_register: %s\n", atlk_rc_to_str(rc));
    return;
  }

  printf("Module my_module is registered to HTTP server\n");

  return;
}
```

8.33 craton-threadx/net/nx-bsd-udp-receive-example.c

```
/* Copyright (C) 2014-2015 Autotalks Ltd. */
#include <stdio.h>

#include <nxd_bsd.h>

/*
  CRATON NX-BSD UDP Recieve Example

  This example demonstrates basic usage of NX-BSD API for code running on top
  of CRATON processor with ThreadX RTOS.
  A UDP socket is opened and bound to an arbitrary port number. UDP packets
  are received in a loop and their length is printed to console.
*/

/* Port number used in this example */
#define EXAMPLE_PORT_NUMBER 2015

/* Maximum message size in octets */
#define EXAMPLE_MAX_MSG_SIZE 1000

void craton_user_init(void)
{
  /* Internet address family socket address */
  struct sockaddr_in sockaddr;
  /* Socket's file descriptor */
  int fd = -1;
  /* POSIX return code */
  int rc = 0;

  printf("NX-BSD UDP receive example.\n");

  /* Create a UDP socket */
  printf("Creating UDP socket...\n");
  fd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
  if (fd == -1) {
    fprintf(stderr, "socket failed: %d\n", errno);
    return;
  }

  /* Prepare socket address struct */
  memset(&sockaddr, 0, sizeof(sockaddr));
  sockaddr.sin_family = AF_INET;
  sockaddr.sin_port = htons(EXAMPLE_PORT_NUMBER);
  sockaddr.sin_addr.s_addr = INADDR_ANY;

  /* Bind the socket */
  printf("Binding UDP socket to port %d...\n", EXAMPLE_PORT_NUMBER);
  rc = bind(fd, (struct sockaddr *)&sockaddr, sizeof(sockaddr));
  if (rc == -1) {
    fprintf(stderr, "bind failed: %d\n", errno);
    return;
  }

  while (1) {
    /* Buffer for received messages */
    char msg[EXAMPLE_MAX_MSG_SIZE];
    /* Length of received message */
    ssize_t len = 0;

    printf("Receiving UDP packet...\n");
    len = recv(fd, msg, sizeof(msg), 0);
    if (len == -1) {
      fprintf(stderr, "recv failed: %d\n", errno);
      continue;
    }

    /* Print received packet length */
    printf("Received UDP packet length: %d bytes.\n", len);
  }

  return;
}
```

8.34 craton-threadx/net/nx-raw-packet-receive-example.c

```
/* Copyright (C) 2014-2015 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>

#include <tx_api.h>
#include <nx_api.h>

#include <craton/net.h>

/*
  CRATON NetX RAW Packet Receive Example

  This example demonstrates how to receive RAW frames over Ethernet using
  NetX-Duo API for code running on top of CRATON processor with ThreadX RTOS.

  A single thread is created in which frames are received in a loop. Frame
  content is dumped to console.
  @todo: This example is not currently supported in multi-core SDK.
*/

/* Example thread priority */
#define EXAMPLE_THREAD_PRIORITY 41

/* Ethernet interface index in trusted IP instance */
#define ETH_IF_INDEX 0

#if defined __CRATON_NO_ARC

/* Example thread */
static TX_THREAD example_thread;
static uint8_t example_thread_stack[0x1000];
static void example_thread_entry(ULONG input);

/* Pointer to trusted IP instance */
static NX_IP *trusted_instance = NULL;

static void
raw_example_print_buffer(const uint8_t *buf, size_t buf_len)
{
  size_t i;

  for (i = 0; i < buf_len; i++) {
    printf("%02x ", buf[i]);
  }
  printf("\n");
}

void craton_user_init(void)
{
  /* ThreadX return value */
  ULONG trv = TX_SUCCESS;
  /* NetX return value */
  ULONG nrv = NX_SUCCESS;
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;

  printf("NetX RAW packet receive example.\n");

  /* Get trusted IP instance */
  rc = net_ip_trusted_instance_get(&trusted_instance);
  if (atlk_error(rc)) {
    fprintf(stderr, "net_ip_trusted_instance_get: %s\n", atlk_rc_to_str(rc));
    return;
  }

  /* Enable RAW packet support on trusted instance. Note that IPv4, ARP and
     IPv6 packets will not be received (they are processed internally by the
     NetX-Duo stack).
   */
  nrv = nx_raw_packet_enable(trusted_instance);
  assert(nrv == NX_SUCCESS);

  /* Create example thread */
  trv = tx_thread_create(&example_thread, "example_thread",
                         example_thread_entry, 0,
                         example_thread_stack,
                         sizeof(example_thread_stack),
                         EXAMPLE_THREAD_PRIORITY,
                         EXAMPLE_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  return;
}

void example_thread_entry(ULONG input)
{
  /* NetX return value */
  ULONG nrv = NX_SUCCESS;
  /* Not using input */
  (void)input;

  while (1) {
    /* NetX packet */
    NX_PACKET *raw_packet = NULL;

    /* Receive a RAW packet (except for IPv4, ARP and IPv6 packets) */
    printf("Receiving RAW packet...\n");
    nrv = nx_raw_packet_receive(trusted_instance, ETH_IF_INDEX,
                                &raw_packet, TX_WAIT_FOREVER);
    assert(nrv == NX_SUCCESS);

    /* Print received packet content */
    printf("Received RAW packet content:\n");
    raw_example_print_buffer(raw_packet->nx_packet_prepend_ptr,
                             raw_packet->nx_packet_length);

    /* Release the packet */
    nx_packet_release(raw_packet);
  }
}

#else /* __CRATON_NO_ARC */

void craton_user_init(void)
{
}

#endif /* __CRATON_NO_ARC */
```

8.35 craton-threadx/net/udp-receive-example.c

```
/* Copyright (C) 2014-2015 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <inttypes.h>

#include <tx_api.h>
#include <nx_api.h>

#include <craton/net.h>

/* UDP port to receive on */
#define UDP_RX_PORT 2009

/* Receiving thread parameters */
#define UDP_RX_THREAD_STACK_SIZE (1 << 12)
#define UDP_RX_THREAD_PRIORITY 40

/* Receiving thread */
static TX_THREAD udp_rx_thread;
static uint8_t udp_rx_thread_stack[UDP_RX_THREAD_STACK_SIZE];
static void udp_rx_thread_entry(ULONG input);

/* Example UDP socket */
static NX_UDP_SOCKET udp_socket;

void craton_user_init(void)
{
  /* API return code */
  atlk_rc_t rc = ATLK_OK;
  /* NetX return value */
  ULONG nrv = NX_SUCCESS;
  /* ThreadX return value */
  ULONG trv = TX_SUCCESS;
  /* NetX trusted IP instance */
  NX_IP *ip_instance = NULL;

  printf("Initializing example UDP server...\n");

  /* Get trusted IP instance */
  rc = net_ip_trusted_instance_get(&ip_instance);
  assert(!atlk_error(rc));

  /* Create a UDP socket */
  nrv = nx_udp_socket_create(ip_instance, &udp_socket, "example_udp_socket",
                             NX_IP_NORMAL, NX_FRAGMENT_OKAY, 0x80, 20);
  assert(nrv == NX_SUCCESS);

  /* Bind the UDP socket to the UDP port */
  nrv = nx_udp_socket_bind(&udp_socket, UDP_RX_PORT, NX_NO_WAIT);
  assert(nrv == NX_SUCCESS);

  /* Create UDP receive thread */
  trv = tx_thread_create(&udp_rx_thread, "example_udp_server_thread",
                         udp_rx_thread_entry, 0,
                         udp_rx_thread_stack, sizeof(udp_rx_thread_stack),
                         UDP_RX_THREAD_PRIORITY, UDP_RX_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  return;
}

void udp_rx_thread_entry(ULONG input)
{
  /* NetX return value */
  ULONG nrv = NX_SUCCESS;
  /* NetX packet */
  NX_PACKET *udp_packet;
  /* Data buffer */
  uint8_t data_buffer[4];
  /* Data buffer size */
  ULONG data_buffer_size;
  /* Not using input */
  (void)input;

  printf("Example UDP server thread receiving on port %d\n", UDP_RX_PORT);

  while(1) {
    /* Receive a UDP packet */
    nrv = nx_udp_socket_receive(&udp_socket, &udp_packet, TX_WAIT_FOREVER);
    assert(nrv == NX_SUCCESS);

    printf("Example UDP server: received packet\n");

    /* Copy (possibly part of) packet data into local buffer */
    data_buffer_size = sizeof(data_buffer);
        nrv = nx_packet_data_retrieve(udp_packet, data_buffer, &data_buffer_size);
    assert(nrv == NX_SUCCESS);

        /* Release packet */
    nrv = nx_packet_release(udp_packet);
    assert(nrv == NX_SUCCESS);

        /* Print some data */
    if (data_buffer_size >= 4) {
      printf("... First 4 bytes are: 0x%02x, 0x%02x, 0x%02x, 0x%02x\n",
             data_buffer[0], data_buffer[1], data_buffer[2], data_buffer[3]);
    }
    else {
      printf("... Payload is shorter than 4 bytes\n");
    }
  }
}
```

8.36 craton-threadx/otp/otp-example.c

```
/* Copyright (C) 2014-2015 Autotalks Ltd. */
#include <stdio.h>
#include <string.h>

#include <craton/nor_flash.h>

/*
  OTP - One Time Programmable module

  This example demonstrates basic usage of OTP API for code running
  on top of CRATON processor with ThreadX RTOS.

  The example reads OTP information and writes to OTP only once.
*/

/* Serial string- length should not exceed 127 */
#define SERIAL_STRING_EXAMPLE    "1h7J9#2a%Hk1D6j8Zz1$P2g6"

/* Max Size of serial buffer */
#define SERIAL_SIZE              128

/* check if partition is empty (all bytes == 0xff)  */
static uint8_t part_is_empty(char *part)
{
  uint16_t i;

  for (i=0 ; i < SERIAL_SIZE ; i++){
    if (part[i] != 0xff)
      return 0;
  }

  return 1;
}

/* Example how to read serial string */
static void serial_read_example(void)
{
  /* Partition table */
  norfl_part_table_t part_table = NORFL_PART_TABLE_INIT;
  /* Serial buffer */
  char serial_buff[SERIAL_SIZE];

  uint8_t part_num;

  atlk_rc_t rc = ATLK_OK;

  /* set buffer - verify that it will be zero terminated */
  memset(&serial_buff[0], 0x00, sizeof(serial_buff));
  /* Read partition table */
  rc = norfl_part_table_read(&part_table);
  if (atlk_error(rc)) {
    printf("ERROR: Failed to read partition table, rc=%d\n",rc);
    goto err;
  };

  /* Search for lockable partition */
  for (part_num = 0 ; part_num < NORFL_NUM_PARTS; part_num++) {
    uint32_t part_type = part_table.part_info[part_num].part_type;
    if ((part_type == (NORFL_PART_TYPE_USER_MIN |
      NORFL_PART_TYPE_F_LOCKABLE)) ||
        (part_type == (NORFL_PART_TYPE_USER_MIN |
      NORFL_PART_TYPE_F_LOCKED))) {
      break;
    }
  }

  /* validate part_num value */
  if (part_num == NORFL_NUM_PARTS) {
    printf("ERROR: Didn't Find Lockable/Locked Partition, rc=%d\n",rc);
    goto err;
  }

  /* read serial from partition */
  rc = norfl_part_read(part_num,
                       0,
                       &serial_buff[0],
                       sizeof(serial_buff));
  if (atlk_error(rc)) {
    printf("ERROR: Failed to read, rc=%d\n",rc);
    goto err;
  };

  if (!part_is_empty(serial_buff)) {
    printf("Serial string: %s\n", serial_buff);
  }
  else {
    printf("Serial Is empty\n");
  }

err:
  return;

}

/* Example how to write serial string */
static void serial_write_example(void)
{
  /* Partition table */
  norfl_part_table_t part_table = NORFL_PART_TABLE_INIT;
  /* Serial buffer */
  char serial_buff[SERIAL_SIZE];

  uint8_t part_num;

  atlk_rc_t rc = ATLK_OK;

  /* set buffer - verify that it will be zero terminated */
  memset(&serial_buff[0], 0x00, sizeof(serial_buff));

  /* Read partition table */
  rc = norfl_part_table_read(&part_table);
  if (atlk_error(rc)) {
    printf("ERROR: Failed to read partition table, rc=%d\n",rc);
    goto err;
  };

  /* Search for lockable partition */
  for (part_num = 0 ; part_num < NORFL_NUM_PARTS; part_num++) {
    uint32_t part_type = part_table.part_info[part_num].part_type;
    if (part_type == (NORFL_PART_TYPE_USER_MIN |
      NORFL_PART_TYPE_F_LOCKABLE)) {
      break;
    }
    else if (part_type == (NORFL_PART_TYPE_USER_MIN |
      NORFL_PART_TYPE_F_LOCKED)) {
      printf("ERROR: Partition is already LOCKED\n");
      goto err;
    }
  }

  /* validate part_num value */
  if (part_num == NORFL_NUM_PARTS) {
    printf("ERROR: Didn't Find Lockable Partition\n");
    goto err;
  }

  /* Set Serial string */
  memcpy(&serial_buff[0],
         SERIAL_STRING_EXAMPLE,
         strlen(SERIAL_STRING_EXAMPLE));

  /* write serial into lockable partition
   * Note : Re-write can be done as many you want until
   *        lock operation
   */
  rc = norfl_part_rewrite(part_num,
                          serial_buff,
                          sizeof(serial_buff));
  if (atlk_error(rc)) {
    printf("ERROR: Failed to re-write, rc=%d\n",rc);
    goto err;
  };
  printf("Re-Write: serial '%s' to partition %d\n",
          serial_buff,
          part_num);

  /* lock Partition */

  /*
     Note - 'Lock' code is under comment to allow execute
            example several times and not lock partition on
            first re-write.
  */

/*  rc = norfl_part_lock(part_num);
  if (atlk_error(rc)) {
    printf("ERROR: Failed to lock partition, rc=%d\n",rc);
    goto err;
  };

  printf("Partition %d is locked \n",
          part_num);
 */

err:
  return;
}


void craton_user_init(void)
{

  /*
     First Run Output: (Serial is Empty)
       read  - Serial Is empty
       write - Re-Write serial 'Hk1D6j8Zz1$P2g6' to partition 15
       read  - Serial string: 'Hk1D6j8Zz1$P2g6'

     if partition is locked:
     Second Run Output: (Serial is locked) -
       read  - Serial string: 'Hk1D6j8Zz1$P2g6'
       write - ERROR: Didn't Find Lockable Partition
       read  - Serial string: 'Hk1D6j8Zz1$P2g6'

     if partition is NOT locked (current implementation):
     Second Run Output: (Serial is Not locked) -
       read  - Serial string: 'Hk1D6j8Zz1$P2g6'
       write - Re-Write serial 'Hk1D6j8Zz1$P2g6' to partition 15
       read  - Serial string: 'Hk1D6j8Zz1$P2g6'
  */

  /* read Serial String */
  serial_read_example();

  /* Write Serial String */
  serial_write_example();

  /* read Serial String */
  serial_read_example();

  return;

}
```

8.37 craton-threadx/posix/posix-example.c

```
#include <unistd.h>
```

```
#include <assert.h>

#include <tx_posix.h>

/*
  POSIX Compliancy Wrapper API Example

  This example demonstrates basic usage of the POSIX Compliancy Wrapper
  API for code running on top of CRATON processor with ThreadX RTOS.

  It includes examples of six pthreads of different priorities, using a
  message queue, semaphore and mutex. Two additional threads are created
  to demonstrate thread control.

  NOTE: This example is based on "posix_demo.c" which is included with
        the POSIX Compliancy Wrapper for ThreadX package.

  @todo: This example is not currently supported in multi-core SDK.
*/

#define     MAX_MESSAGE_SIZE        50
#define     DEMO_BYTE_POOL_SIZE     9120
#define     EXIT_COUNT              50
#define     THREAD0_EXIT_COUNT      (EXIT_COUNT / 10)

/* Set stack size for each thread */
#define EXAMPLE_STACK_SIZE (1 << 14)

/* Define the POSIX pthread object control blocks ... */
pthread_t               pthread_0;
pthread_t               pthread_0_join_0;
pthread_t               pthread_0_join_1;
pthread_t               pthread_1;
pthread_t               pthread_2;
pthread_t               pthread_3;
pthread_t               pthread_4;
pthread_t               pthread_5;
pthread_t               pthread_6;
pthread_t               pthread_7;
pthread_t               pthread_8;

/* Define pthread attributes objects */
pthread_attr_t          ptattr0;
pthread_attr_t          ptattr0_join0;
pthread_attr_t          ptattr0_join1;
pthread_attr_t          ptattr1;
pthread_attr_t          ptattr2;
pthread_attr_t          ptattr3;
pthread_attr_t          ptattr4;
pthread_attr_t          ptattr5;
pthread_attr_t          ptattr6;
pthread_attr_t          ptattr7;
pthread_attr_t          ptattr8;


/* Define the message queue attribute.  */
struct mq_attr          queue_atrr;

/* Define a queue descriptor.          */
mqd_t                   q_des;

/* Define a semaphore.                 */
sem_t                   *sem;

/* Define an unnamed semaphore.       */
sem_t                   unnamed_sem;

/* Define a mutex                     */
pthread_mutex_t         mutex1, mutex2;

/* Define a mutex attributes object   */
pthread_mutexattr_t     mta2;

/* Conditional variable               */
pthread_cond_t          cond;

/* Define the counters used in this demo application...  */
unsigned int     pthread_0_counter;
unsigned int     pthread_0_join_0_counter;
unsigned int     pthread_0_join_1_counter;
unsigned int     pthread_1_counter;
unsigned int     pthread_2_counter;
unsigned int     pthread_3_counter;
unsigned int     pthread_4_counter;
unsigned int     pthread_5_counter;
unsigned int     pthread_6_counter;
unsigned int     pthread_7_counter;
unsigned int      pthread_8_counter;
unsigned int      pthread_1_message_sent;
unsigned int      pthread_2_message_received;


/* Define pthread function prototypes.  */
void    *pthread_0_entry(void *);
void    *pthread_0_join_0_entry(void *);
void    *pthread_0_join_1_entry(void *);
void    *pthread_1_entry(void *);
void    *pthread_2_entry(void *);
void    *pthread_3_entry(void *);
void    *pthread_4_entry(void *);
void    *pthread_5_entry(void *);
void    *pthread_6_entry(void *);
void    *pthread_7_entry(void *);
void    *pthread_8_entry(void *);


/* Message to be sent.  */
char *msg0 = "This is a test message";
char msg0_priority = 3;

/* Memory pool for POSIX internal objects and thread stacks. */
static char pthread_0_stack[EXAMPLE_STACK_SIZE];
static char pthread_1_stack[EXAMPLE_STACK_SIZE];
static char pthread_2_stack[EXAMPLE_STACK_SIZE];
static char pthread_3_stack[EXAMPLE_STACK_SIZE];
static char pthread_4_stack[EXAMPLE_STACK_SIZE];
static char pthread_5_stack[EXAMPLE_STACK_SIZE];
static char pthread_6_stack[EXAMPLE_STACK_SIZE];


/* Define what the initial system looks like.  */
void craton_user_init(void)
{
  struct sched_param  param;

  queue_atrr.mq_maxmsg  = 124;
  queue_atrr.mq_msgsize = MAX_MESSAGE_SIZE;

  /* Put system definition stuff in here, e.g. pthread creates and
   * other assoerted create information. */

  /* Create pthread attributes for pthread 0 to pthread 5 */
  pthread_attr_init(&ptattr0);
  pthread_attr_init(&ptattr1);
  pthread_attr_init(&ptattr2);
  pthread_attr_init(&ptattr3);
  pthread_attr_init(&ptattr4);
  pthread_attr_init(&ptattr5);
  pthread_attr_init(&ptattr6);

  /* Create a sched_param structure */
  memset(&param, 0, sizeof(param));

  /* Now create all pthreads , firstly modify respective ptheread
     attribute with desired priority and stack start
     address and then create the pthread */

  /* Create pthread 0.  */
  param.sched_priority = 10;
  pthread_attr_setschedparam(&ptattr0, &param);
  pthread_attr_setstackaddr(&ptattr0, pthread_0_stack);
  pthread_attr_setstacksize(&ptattr0, sizeof(pthread_0_stack));
  pthread_create(&pthread_0, &ptattr0,pthread_0_entry,NULL);

  /* Create pthread 1.  */
  param.sched_priority = 15;
  pthread_attr_setschedparam(&ptattr1, &param);
  pthread_attr_setstackaddr(&ptattr1, pthread_1_stack);
  pthread_attr_setstacksize(&ptattr1, sizeof(pthread_1_stack));
  pthread_create (&pthread_1, &ptattr1,pthread_1_entry,NULL);

  /* Create pthread 2.  */
  param.sched_priority = 20;
  pthread_attr_setschedparam(&ptattr2, &param);
  pthread_attr_setstackaddr(&ptattr2, pthread_2_stack);
  pthread_attr_setstacksize(&ptattr2, sizeof(pthread_2_stack));
  pthread_create (&pthread_2, &ptattr2,pthread_2_entry,NULL);

  /* Create pthread 3.  */
  param.sched_priority = 25;
  pthread_attr_setschedparam(&ptattr3, &param);
  pthread_attr_setstackaddr(&ptattr3, pthread_3_stack);
  pthread_attr_setstacksize(&ptattr3, sizeof(pthread_3_stack));
  pthread_create (&pthread_3, &ptattr3,pthread_3_entry,NULL);
  /* Create pthread 4.  */
  param.sched_priority = 24;
  pthread_attr_setschedparam(&ptattr4, &param);
  pthread_attr_setstackaddr(&ptattr4, pthread_4_stack);
  pthread_attr_setstacksize(&ptattr4, sizeof(pthread_4_stack));
  pthread_create(&pthread_4, &ptattr4,pthread_4_entry,NULL);

  /* Create pthread 5.  */
  param.sched_priority = 30;
  pthread_attr_setschedparam(&ptattr5, &param);
  pthread_attr_setstackaddr(&ptattr5, pthread_5_stack);
  pthread_attr_setstacksize(&ptattr5, sizeof(pthread_5_stack));
  pthread_create (&pthread_5, &ptattr5,pthread_5_entry,NULL);

  /* Create pthread 6.  */
  param.sched_priority = 5;
  pthread_attr_setschedparam(&ptattr6, &param);
  pthread_attr_setstackaddr(&ptattr6, pthread_6_stack);
  pthread_attr_setstacksize(&ptattr6, sizeof(pthread_6_stack));
  pthread_attr_setdetachstate(&ptattr6, PTHREAD_CREATE_DETACHED);
  pthread_create (&pthread_6, &ptattr6,pthread_6_entry,NULL);

  /* Create pthread 7.  */
  /* Use default values for thread attribute*/
  pthread_attr_init(&ptattr7);
  pthread_attr_setschedpolicy(&ptattr7, SCHED_FIFO);
  pthread_create (&pthread_7, &ptattr7, pthread_7_entry, NULL);

  /* Create pthread 8.  */
  /* Use default values for thread attribute*/
  pthread_create (&pthread_8, NULL, pthread_8_entry, NULL);

  /* Create a Message queue.  */
  q_des = mq_open("Queue",O_CREAT|O_RDWR,0,&queue_atrr);

  /* Create a Semaphore.  */
  sem = sem_open("Sem0", O_CREAT | O_EXCL,0,1);

  /* Create an unnamed Semaphore.  */
  sem_init(&unnamed_sem ,0 ,0);

  /* Create a Mutex */
  pthread_mutex_init(&mutex1, NULL);

  /* Create mutex attribute */
  pthread_mutexattr_init(&mta2);
  pthread_mutexattr_setprotocol(&mta2, PTHREAD_PRIO_INHERIT);
  pthread_mutexattr_setpshared(&mta2, PTHREAD_PROCESS_SHARED);
  pthread_mutexattr_settype(&mta2, PTHREAD_MUTEX_RECURSIVE);

  /* Create a Mutex */
  pthread_mutex_init(&mutex2, &mta2);

  /* Create a Cond var */
  pthread_cond_init(&cond, NULL);
}

/* Global thread status variables.  */
int pt0_status = 0;
int pt0_j0_status = 0;
int pt0_j1_status = 0;
int pt1_status = 0;
int pt2_status = 0;
int pt3_status = 0;
int pt4_status = 0;
int pt5_status = 0;
int pt6_status = 0;
int pt7_status = 0;
int pt8_status = 0;

static int thread_completed_count = 0;

static void increase_thread_complete_count(void)
{

  pthread_mutex_lock(&mutex2);
  thread_completed_count++;
  printf("Thread complete count is %d\n", thread_completed_count);

  if (thread_completed_count == 8) {
    pthread_cond_signal(&cond);
  }
  pthread_mutex_unlock(&mutex2);
}
```

```
static void pthread_example_clean_up(void)
{
  int status;

  /* Destroy the mutex */
  status = pthread_mutex_destroy(&mutex1);
  assert(status == 0);
  status = pthread_mutex_destroy(&mutex2);
  assert(status == 0);

  /* Destroy the mutex attribute */
  status = pthread_mutexattr_destroy(&mta2);
  assert(status == 0);

  /* Unlink message queue*/
  status = mq_unlink("Queue");
  assert(status == 0);

  /* Close message queue*/
  status = mq_close(q_des);
  assert(status == 0);

  /* Unlink the named semaphore */
  status = sem_unlink("Sem0");
  assert(status == 0);

  /* Destroy the semaphore */
  status = sem_close(sem);
  assert(status == 0);

  /* Destroy the semaphore */
  status = sem_destroy(&unnamed_sem);
  assert(status == 0);

}

void    *pthread_0_entry(void *pthread0_input)
{
  int trv;
  (void)pthread0_input;

  /* This pthread simply sits in while-forever-sleep loop */
  printf("Entered %s\n", __func__);

  /* Create pthread attributes for child threads */
  pthread_attr_init(&ptattr0_join0);
  pthread_attr_init(&ptattr0_join1);

  /* Create pthread 0.  */
  pthread_attr_setinheritsched(&ptattr0_join0, PTHREAD_INHERIT_SCHED);
  pthread_create(&pthread_0_join_0, &ptattr0_join0,
                 pthread_0_join_0_entry,NULL);

  /* Create pthread 1.  */
  pthread_attr_setinheritsched(&ptattr0_join1, PTHREAD_INHERIT_SCHED);
  pthread_create(&pthread_0_join_1, &ptattr0_join1,
                 pthread_0_join_1_entry,NULL);

  printf("%s waiting on join\n", __func__);
  trv = pthread_join(pthread_0_join_0, NULL);
  if (trv) {
    printf("pthread_join in %s failed\n", __func__);
  }

  printf("%s is trying to cancel pthread_0_join_1_entry \n", __func__);
  trv =  pthread_cancel(pthread_0_join_1);
  if (trv) {
    printf("pthread_join in %s failed\n", __func__);
  }
  pthread_detach(pthread_0_join_1);

  /* Create pthread attributes for child threads */
  pthread_attr_destroy(&ptattr0_join0);
  pthread_attr_destroy(&ptattr0_join1);

  while(1)
  {
    /* Increment the pthread counter.*/
    pthread_0_counter++;
    printf("%s sleep %u\n", __func__, pthread_0_counter);

    /* sleep for a while  */
    pt0_status=sleep(2);
    if(pt0_status)
      break;

    if (pthread_0_counter == THREAD0_EXIT_COUNT) {
      struct timespec time = {1, 0};

      /* Wait on timeout for condition */
      pthread_mutex_lock(&mutex2);
      pt6_status = pthread_cond_timedwait(&cond, &mutex2, &time);
      pthread_mutex_unlock(&mutex2);

      /* Wait for all threads to complete */
      pthread_mutex_lock(&mutex2);
      printf("Waiting for all threads to complete\n");
      pt6_status = pthread_cond_wait(&cond, &mutex2);
      pthread_mutex_unlock(&mutex2);

      pthread_example_clean_up();

      printf("Example ended\n");

      /* Terminate the thread */
      pthread_exit(&pt0_status);
    }
  }

  return(&pt0_status);
}

void    *pthread_0_join_0_entry(void *pthread0_join0_input)
{
  struct timespec thread_0_join_0_sleep_time={0,0};
  (void)pthread0_join0_input;

  printf("Entered %s\n", __func__);
  while(1)
  {
    /* Increment the pthread counter.*/
    pthread_0_join_0_counter++;
    printf("%s sleep %u\n", __func__, pthread_0_join_0_counter);

    if (pthread_0_join_0_counter == EXIT_COUNT) {
      /* Compare pthread ID's */
      pt0_j0_status = pthread_equal(pthread_self() ,pthread_0_join_0);

      /* Verify that received pthread id is the correct one for this thread */
      if (!pt0_j0_status) {
        printf("\n Incorrect pthread id in pthread_0_join_0 \n");
        assert(0);
      }
      /* Terminate the thread */
      pthread_exit(&pt0_j0_status);
    }

    /* sleep for a while  */
    thread_0_join_0_sleep_time.tv_nsec =  9999999;
    pt0_status=nanosleep(&thread_0_join_0_sleep_time,0);
    if(pt0_j0_status)
      break;
  }
  printf("Completed %s\n", __func__);
  return(&pt0_j0_status);
}

void    *pthread_0_join_1_entry(void *pthread0_join1_input)
{
  int old_state = 0;
  int trv = 0;
  (void)pthread0_join1_input;

  struct timespec thread_0_join_1_sleep_time={0,0};

  printf("Entered %s\n", __func__);
  printf("Cancel state set to ENABLE\n");
  trv = pthread_setcancelstate(PTHREAD_CANCEL_ENABLE,&old_state);
  if (trv) {
    printf("%s was not able to set cancelstate\n", __func__);
  }

  trv = pthread_setcanceltype(PTHREAD_CANCEL_ASYNCHRONOUS,&old_state);
  if (trv) {
    printf("%s was not able to set cancelstate\n", __func__);
  }

  while(1)
  {
    /* Increment the pthread counter.*/
    pthread_0_join_1_counter++;
    printf("%s sleep %u\n", __func__, pthread_0_join_1_counter);

    /* Place a cancel check point to allow parent thread synchronized thread
```

```
      * cancellation  */
     pthread_testcancel();

     /* sleep for a while  */
     thread_0_join_1_sleep_time.tv_nsec =  9999999;
     thread_0_join_1_sleep_time.tv_sec =   0;
     pt0_j1_status=nanosleep(&thread_0_join_1_sleep_time,0);
     if(pt0_j1_status)
       break;
   }
   printf("Completed %s\n", __func__);
   return(&pt0_j1_status);
}


void    *pthread_1_entry(void *pthread1_input)
{
   (void)pthread1_input;

   struct timespec thread_1_sleep_time={0,0};

   printf("Entered %s\n", __func__);
   /* This thread simply sends a messages to a queue shared by pthread 2.  */
   while(1)
   {

     /* Increment the thread counter.  */
     pthread_1_counter++;
     printf("pthread_1_entry send message %u\n", pthread_1_counter);

     /* Send message to queue 0.  */
     pt1_status = mq_send(q_des,msg0,strlen(msg0),msg0_priority);
     /* check status.  */
     if(pt1_status)
       break;

     /* Increment the message sent.  */
     pthread_1_message_sent++;

     /* sleep for a while  */
     thread_1_sleep_time.tv_nsec = 900000000;
     nanosleep(&thread_1_sleep_time,0);

     if (pthread_1_counter == EXIT_COUNT) {
       increase_thread_complete_count();
       /* Terminate thread */
       pthread_exit(&pt1_status);
     }
   }
   assert(0);
   return(&pt1_status);
}

void    *pthread_2_entry(void *pthread2_input)
{

   char    msgr0[MAX_MESSAGE_SIZE];
   unsigned int   priority;
   struct timespec thread_2_sleep_time={0,0};

   (void)pthread2_input;

   printf("Entered %s\n", __func__);
   /* This pthread retrieves messages placed on the queue by pthread 1.  */
   while(1 )
   {
     /* Increment the thread counter.  */
     pthread_2_counter++;
     pt2_status =
         (unsigned int)mq_receive(q_des, msgr0, MAX_MESSAGE_SIZE,
                                  (ULONG *)&priority);
     printf("pthread_2_entry receive message %u\n", pthread_2_counter);

     /* Check received data size */
     if ((unsigned int)pt2_status != strlen(msg0)) {
       printf("ERROR: Message length for received message is incorrect\n");
       break;
     }

     /* Check receive message priority */
     if (priority != msg0_priority) {
       printf("ERROR: Message priority for received message is incorrect\n");
       break;
     }

     if(pt2_status == ERROR)
       break;
```

```
    /* Otherwise, it is OK to increment the received message count.  */
    pthread_2_message_received++;
    /* sleep for a while  */
    thread_2_sleep_time.tv_nsec = 900000000;
    nanosleep(&thread_2_sleep_time,0);

    if (pthread_2_counter == EXIT_COUNT) {
      increase_thread_complete_count();
      pthread_exit(&pt2_status);
    }
  }
  assert(0);
  return(&pt2_status);
}

void    *pthread_3_entry(void *pthread3_input)
{
  struct timespec thread_3_sleep_time={0,0};
  int rv;
  (void)pthread3_input;

  printf("Entered %s\n", __func__);

  /* Wait with timeout for the unnamed semaphore should return with error
   * as semaphore is not released */
  rv = sem_trywait(&unnamed_sem);
  if (rv != -1 && posix_errno != EAGAIN) {
    assert(0);
  }

  /* This function compete for ownership of semaphore_0.  */
  while(1)
  {

    /* Increment the thread counter.  */
    pthread_3_counter++;

    /* Sleep for a while to hold the semaphore.  */
    thread_3_sleep_time.tv_nsec = 900000000;
    nanosleep(&thread_3_sleep_time,0);

    /* Release the semaphore.  */
    pt3_status = sem_post(sem);
    printf("pthread_3_entry release sem\n");

    /* Check status.  */
    if (pt3_status )
      break;

    if (pthread_3_counter == EXIT_COUNT) {
      increase_thread_complete_count();
      pthread_exit(&pt3_status);
    }

  }
  assert(0);
  return(&pt3_status);
}

void    *pthread_4_entry(void *pthread4_input)
{
  ULONG sem_val;
  struct timespec thread_4_sleep_time={0,0};
  (void)pthread4_input;

  printf("Entered %s\n", __func__);

  /* This function compete for ownership of semaphore_0.  */
  while(1)
  {

    /* Increment the thread counter.  */
    pthread_4_counter++;
    printf("pthread_4_entry lock sem %u\n", pthread_4_counter);

    /* Check the semaphore value */
    pt4_status = sem_getvalue(sem, &sem_val);

    /* Check status.  */
    if ((pt4_status) && (!sem_val))
      break;

    /* Get the semaphore with suspension.  */
    pt4_status = sem_wait(sem);

    /* Check status.  */
    if (pt4_status)
      break;

    /* Sleep for a while to hold the semaphore.  */
    thread_4_sleep_time.tv_nsec = 900000000;
    nanosleep(&thread_4_sleep_time,0);

    if (pthread_4_counter == EXIT_COUNT) {
      increase_thread_complete_count();
      pthread_exit(&pt4_status);
    }

  }
  assert(0);
  return(&pt4_status);
}

void    *pthread_5_entry(void *pthread5_input)
{

  struct timespec thread_5_sleep_time={0,0};
  (void)pthread5_input;

  printf("Entered %s\n", __func__);
  while(1)
  {
    /* Increment the thread counter.  */
    pthread_5_counter++;
    printf("pthread_5_entry wait on mutex lock %u\n", pthread_5_counter);

    /* now lock the mutex */
    pt5_status = pthread_mutex_lock(&mutex1);
    if (pt5_status != OK)
      break;

    printf("pthread_5_entry locked mutex %u\n", pthread_5_counter);

    /* sleep for a while  */
    thread_5_sleep_time.tv_nsec = 900000000;
    nanosleep(&thread_5_sleep_time,0);

    pt5_status = pthread_mutex_unlock(&mutex1);
    printf("pthread_5_entry release mutex %u\n", pthread_5_counter);
    if (pt5_status != OK)
      break;

    if (pthread_5_counter == EXIT_COUNT) {
      increase_thread_complete_count();
      pthread_exit(&pt5_status);
    }
  }
  assert(0);
  return(&pt5_status);
}

static void print_once(void)
{
  /* Flag to mark that the function was run only once */
  static int function_count = 0;
  struct timespec time={0, 50000000};
  int rv;

  if (function_count > 1) {
    printf("Error: The print_once function has been called twice \n");
    assert(0);
  }

  function_count++;
  printf("\n");
  printf(" POSIX Exmaple  \n");
  printf(" Example of pthread_once  \n");
  printf(" This should not be printed again  \n");
  printf("\n");

  rv = pthread_mutex_timedlock(&mutex1, &time);
  if (rv != ETIMEDOUT) {
    printf("Got mutex1 although it should be locked (rv = %d)\n", rv);
    assert(0);
  }
}

void    *pthread_6_entry(void *pthread6_input)
{

  unsigned int     try_lock_count = 0;
  struct timespec thread_6_sleep_time={0,0};
  pthread_once_t once_control = PTHREAD_ONCE_INIT;
  (void)pthread6_input;

  printf("Entered %s\n", __func__);
  while(1)
  {
    thread_6_sleep_time.tv_nsec = 50000000;
    nanosleep(&thread_6_sleep_time,0);

    /* Increment the thread counter. */
    printf("pthread_6_entry trylock mutex %u (try #%u)\n",
           pthread_6_counter, try_lock_count);

    if (try_lock_count < 10) {
      /* Try to lock the mutex */
      pt6_status = pthread_mutex_trylock(&mutex1);

      if (pt6_status == EBUSY) {
        /* Mutex is locked by another thread sleep and try again  */
        try_lock_count++;
        continue;
      }
    }
    else {
      /* Insist on locking mutex if lock was not acquired 10 times in a row */
      printf("pthread_6_entry wait to lock mutex %u\n", pthread_6_counter);
      pt6_status = pthread_mutex_lock(&mutex1);

      try_lock_count = 0;
    }

    if (pt6_status != OK)
      break;

    pthread_6_counter++;
    printf("pthread_6_entry locked mutex %u\n", pthread_6_counter);

    /* sleep for a while  */
    thread_6_sleep_time.tv_nsec = 50000000;
    nanosleep(&thread_6_sleep_time,0);
    pt6_status = pthread_mutex_unlock(&mutex1);
    printf("pthread_6_entry release mutex %u\n", pthread_6_counter);
    if (pt6_status != OK)
      break;

    if (pthread_6_counter > 10) {
      /* Call on a message routine  */
      pthread_once(&once_control, print_once);
    }

    if (pthread_6_counter == EXIT_COUNT) {
      increase_thread_complete_count();
      pthread_exit(&pt6_status);
    }
  }
  assert(0);
  return(&pt6_status);
}


void    *pthread_7_entry(void *pthread7_input)
{
  int policy;
  struct sched_param param;
  (void)pthread7_input;

  printf("Entered %s\n", __func__);
  pthread_getschedparam(pthread_self(), &policy, &param);

  while(1)
  {
    /* Increment the thread counter.  */
    pthread_7_counter++;
    printf("pthread_7_entry iteration %u\n", pthread_7_counter);

    if (pthread_7_counter ==  EXIT_COUNT) {
      pthread_yield();
      pthread_setschedparam(pthread_self(), SCHED_RR, &param);
    }

    if (pthread_7_counter == 7 * EXIT_COUNT) {
      param.sched_priority = 1;
      pthread_setschedparam(pthread_self(), SCHED_RR, &param);
    }

    if (pthread_7_counter == 10 * EXIT_COUNT) {
      increase_thread_complete_count();
      pthread_exit(&pt7_status);
    }
  }
  assert(0);
  return(&pt7_status);
}

void    *pthread_8_entry(void *pthread8_input)
{
  (void)pthread8_input;

  printf("Entered %s\n", __func__);
  while(1)
  {
    /* Increment the thread counter.  */
    pthread_8_counter++;
    printf("pthread_8_entry iteration %u\n", pthread_8_counter);

    if (pthread_8_counter == 5 * EXIT_COUNT) {
      int policy;
      struct sched_param param;

      pthread_yield();
      pthread_getschedparam(pthread_self(), &policy, &param);
      pthread_setschedparam(pthread_self(), SCHED_RR, &param);
    }

    if (pthread_8_counter == 10 * EXIT_COUNT) {
      increase_thread_complete_count();
      pthread_exit(&pt8_status);
    }
  }
  assert(0);
  return(&pt8_status);
}
```

8.38 craton-threadx/sntp/sntp-example.c

```
/* Copyright (C) 2015 Autotalks Ltd. */
#include <stdio.h>
#include <stdint.h>
#include <unistd.h>
#include <math.h>

#include <time.h>
#include <sys/time.h>

#include <tx_api.h>

#include <atlk/sntp_client.h>
#include <atlk/mib_service.h>
#include <atlk/mibs/nav-mib.h>

/*
  CRATON SNTP Client Example

  This example demonstrates basic usage of SNTP client API for code running
  on top of CRATON processor with ThreadX RTOS.
*/

/* SNTP client thread priority */
#define SNTP_CLIENT_PRIORITY 20

/* IP address of NTP server used in this example */
#define SERVER_ADDRESS (10 << 24 | 10 << 16 | 1 << 8 | 110 << 0)

#ifdef __CRATON_ARM

/* Example callback that will be invoked on every NTP update received */
static void
sntp_update_callback(const sntp_info_t *info)
{
  /* NTP time */
  double ntp_time = 0.0;
  /* System time */
  struct timeval sys_time;
  /* Buffer for datetime string */
  char buf[64];

  /* Convert NTP time to struct timeval */
  ntp_time = sntp_time_to_posix_time(info);
  sys_time.tv_sec = floor(ntp_time);
  sys_time.tv_usec = ntp_time - (double)sys_time.tv_sec;
  /* Set system time */
  settimeofday(&sys_time, NULL);

  /* Get system time */
  gettimeofday(&sys_time, NULL);

  /* Print system time as datetime string */
  strftime(buf, sizeof(buf), "%b %d %X %Y", localtime(&sys_time.tv_sec));
  printf("Received update! System time set to: %s\n", buf);
}

/* Disable syncing of system time with external 1-PPS */
static atlk_rc_t
disable_pps_sync(void)
{
  /* MIB service */
  mib_service_t *service = NULL;
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;

  /* Get default MIB service instance */
  rc = mib_default_service_get(&service);
  if (atlk_error(rc)) {
    fprintf(stderr, "mib_default_service_get: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Disable syncing of system time with external 1-PPS */
  rc = mib_set_navSysTimePpsSyncEnabled(service, 0);
  if (atlk_error(rc)) {
    fprintf(stderr, "mib_set_navSysTimePpsSyncEnabled: %s\n",
            atlk_rc_to_str(rc));
    goto error;
  }

error:
  /* Cleanup resources */
  mib_service_delete(service);

  return rc;
}

void craton_user_init(void)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* SNTP client configuration parameters */
  sntp_client_config_t config = SNTP_CLIENT_CONFIG_INIT;

  printf("Starting SNTP client example...\n");

  /* Disable syncing of system time with external 1-PPS */
  rc = disable_pps_sync();
  if (atlk_error(rc)) {
    fprintf(stderr, "Failed to disable syncing with 1-PPS\n");
    return;
  }

  /* Set SNTP client configuration parameters */
  config.sched_params.priority = SNTP_CLIENT_PRIORITY;
  config.update_handler = sntp_update_callback;
  config.ntp_server_address = SERVER_ADDRESS;
  config.type = SNTP_CONNECTION_TYPE_UNICAST;
  config.max_root_dispersion_us = 0;
  config.min_server_stratum = 5;
  config.unicast_poll_interval_s = 15;

  /* Initialize SNTP client */
  rc = sntp_client_init(&config);
  if (atlk_error(rc)) {
    fprintf(stderr, "sntp_client_init: %s\n", atlk_rc_to_str(rc));
    return;
  }

  printf("Waiting for NTP update...\n");
}


#else /*__CRATON_ARM */

void craton_user_init(void)
{
}

#endif /* __CRATON_ARM */
```

8.39 craton-threadx/spi/spi-common.h

```
#ifndef _EXAMPLE_SPI_COMMON_H
#define _EXAMPLE_SPI_COMMON_H

/*
  COMMON SPI Example resources, for both SPI master and slave
*/

#define EXAMPLE_SPI_MSG_MASTER_2_SLAVE  "Master message to slave"
#define EXAMPLE_SPI_MSG_SLAVE_2_MASTER "Slave message to master"

#endif /* _EXAMPLE_SPI_COMMON_H */
```

8.40 craton-threadx/spi/spi-master-example.c

```
/* Copyright (C) 2013-2015 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>

#include <tx_api.h>

#include <atlk/sdk.h>
#include <craton/gpio_driver.h>
#include <craton/spi_driver.h>
#include <craton/cache.h>

#include "spi-common.h"

/*
  CRATON ThreadX SPI Master Example

  This example demonstrates basic usage of SPI for code running on
  top of CRATON processor with ThreadX RTOS.

  The example demonstrates how to send and receive data via SPI.

  The flow of the example is:
  1. Master sends a message to slave.
  2. Slave compares the message to expected message, if message is valid
     print success message to console else, prints error message to console.
  3. Master and salve set second transaction parameters,
     master waits for slave for predefined period, and initiate transaction,
     in which master receives from slave.
  4. Master compares the message to expected message, if message is valid
     print success message to console else, print error message to console.

  NOTE: Slave should start before master time out (MASTER_WAIT_FOR_SALVE_USEC)

 */

#define MASTER_WAIT_FOR_SALVE_USEC 5000000
/* SPI master device */
static spi_device_t *spi_master_dev;
/* SPI master sempahore */
static TX_SEMAPHORE spi_master_semaphore;
/* SPI master thread */
static TX_THREAD spi_master_thread;
/* SPI master thread stack */
static uint8_t spi_master_thread_stack[4096];

static void
spi_master_wake_up(void)
{
  ULONG trv = tx_semaphore_put(&spi_master_semaphore);
  assert(trv == TX_SUCCESS);
}

static void
spi_master_sleep(void)
{
  ULONG trv = tx_semaphore_get(&spi_master_semaphore, TX_WAIT_FOREVER);
  assert(trv == TX_SUCCESS);
}

static void
spi_master_init_spi(void)
{
  spi_config_t spi_cfg = SPI_CONFIG_INIT;
  atlk_rc_t rc = ATLK_OK;

  spi_cfg.device_id = 0;
  spi_cfg.device_mode = SPI_MODE_MASTER;
  spi_cfg.data_bits = 8;
  spi_cfg.tx_dma_channel = 1;
  spi_cfg.rx_dma_channel = 0;
  spi_cfg.clock_polarity = SPI_CLOCK_POLARITY_IDLE_LOW;
  spi_cfg.clock_phase = SPI_CLOCK_PHASE_1ST_EDGE;

  rc = spi_device_init(&spi_cfg, &spi_master_dev);
  if (atlk_error(rc)) {
    fprintf(stderr, "spi_device_init: %s\n", atlk_rc_to_str(rc));
  }

  return;
}

static void
spi_master_callback(spi_dma_transfer_t *dev)
{
  (void)dev;

  spi_master_wake_up();
}

static void
spi_master_test(void)
{
  atlk_rc_t rc = ATLK_OK;
  spi_dma_transfer_t transfer = SPI_DMA_TRANSFER_INIT;
  char cache_line_aligned msg_tx[CACHE_LINE_SIZE];
  char cache_line_aligned msg_rx[CACHE_LINE_SIZE];

  transfer.device_ptr = spi_master_dev;

  /* set buffers */
  strncpy(msg_tx, EXAMPLE_SPI_MSG_MASTER_2_SLAVE, sizeof(msg_tx) - 1);
  memset(msg_rx, 0, sizeof(msg_rx));

  /* set transaction parameters */
  transfer.tx_buffer_ptr = msg_tx;
  transfer.tx_buffer_size = sizeof(msg_tx);
  transfer.rx_buffer_ptr = NULL;
  transfer.rx_buffer_size = 0;
  transfer.data_size = sizeof(EXAMPLE_SPI_MSG_MASTER_2_SLAVE);
  transfer.completion_handler = spi_master_callback;

  /* Wait for Slave to set transaction */
  printf("Waiting %d microseconds seconds for slave to be ready\n",
      MASTER_WAIT_FOR_SALVE_USEC);
  usleep(MASTER_WAIT_FOR_SALVE_USEC);

  /* Start transaction */
  rc = spi_dma_transfer_start(&transfer);
  if (atlk_error(rc)) {
    fprintf(stderr, "spi_dma_transfer_start: %s\n", atlk_rc_to_str(rc));
    return;
  }

  /* Wait for DMA to end transfer */
  spi_master_sleep();

  /* set transaction parameters */
  transfer.tx_buffer_ptr = NULL;
  transfer.tx_buffer_size = 0;
  transfer.rx_buffer_ptr = msg_rx;
  transfer.rx_buffer_size = sizeof(msg_rx);
  transfer.data_size = sizeof(EXAMPLE_SPI_MSG_SLAVE_2_MASTER);
  transfer.completion_handler = spi_master_callback;

  /* Wait for Slave to set transaction */
  printf("Waiting %d microseconds for slave to be ready\n",
      MASTER_WAIT_FOR_SALVE_USEC);
  usleep(MASTER_WAIT_FOR_SALVE_USEC);

  /* Start transaction */
  rc = spi_dma_transfer_start(&transfer);
  if (atlk_error(rc)) {
    fprintf(stderr, "spi_dma_transfer_start: %s\n", atlk_rc_to_str(rc));
    return;
  }

  /* Wait for DMA to end transfer */
  spi_master_sleep();

  if (strcmp(msg_rx, EXAMPLE_SPI_MSG_SLAVE_2_MASTER) != 0) {
    printf("Wrong! Message has been received from Slave: %s expected:%s \n",
        msg_rx, EXAMPLE_SPI_MSG_SLAVE_2_MASTER);
  }
  else {
    printf("Message received from Slave: %s\n", msg_rx);
  }

  /* Wait for DMA to end transfer */
  spi_master_sleep();

}

static void
spi_master_thread_entry(ULONG opaque)
{
  (void)opaque;

  spi_master_init_spi();

  spi_master_test();
}

void craton_user_init(void)
{
  ULONG trv;

  trv = tx_semaphore_create(&spi_master_semaphore, "SPI master semaphore", 0);
  assert(trv == TX_SUCCESS);

  trv = tx_thread_create(&spi_master_thread, "SPI master thread",
      spi_master_thread_entry, 0,
      spi_master_thread_stack,
      sizeof(spi_master_thread_stack),
      5, 5,
      TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  return;
}
```

8.41 craton-threadx/spi/spi-slave-example.c

```
/* Copyright (C) 2013-2015 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>

#include <tx_api.h>

#include <atlk/sdk.h>
#include <craton/gpio_driver.h>
#include <craton/spi_driver.h>
#include <craton/cache.h>

#include "spi-common.h"

/*
  CRATON ThreadX SPI Slave Example

  This example demonstrates basic usage of SPI for code running on
  top of CRATON processor with ThreadX RTOS.

  The example demonstrates how to send and receive data via SPI.

  The flow of the example is:
  1. Master sends a message to slave.
  2. Slave compares the message to expected message, if message is valid
     print success message to console else, prints error message to console.
  3. Master and salve set second transaction parameters,
     master waits for slave for predefined period, and initiate transaction,
     in which master receives from slave.
  4. Master compares the message to expected message, if message is valid
     print success message to console else, print error message to console.

  NOTE: Slave should start before master time out.

 */


/* SPI slave device */
spi_device_t *spi_slave_dev;
/* SPI slave sempahore */
TX_SEMAPHORE spi_slave_semaphore;
/* SPI slave thread */
TX_THREAD spi_slave_thread;
/* SPI slave thread stack */
uint8_t spi_slave_thread_stack[4096];

static void
spi_slave_wake_up(void)
```

```
{
  ULONG trv = tx_semaphore_put(&spi_slave_semaphore);
  assert(trv == TX_SUCCESS);
}

static void
spi_slave_sleep(void)
{
  ULONG trv = tx_semaphore_get(&spi_slave_semaphore, TX_WAIT_FOREVER);
  assert(trv == TX_SUCCESS);
}

static void
spi_slave_init_spi(void)
{
  spi_config_t spi_cfg = SPI_CONFIG_INIT;
  atlk_rc_t rc = ATLK_OK;

  spi_cfg.device_id = 0;
  spi_cfg.device_mode = SPI_MODE_SLAVE;
  spi_cfg.data_bits = 8;
  spi_cfg.tx_dma_channel = 1;
  spi_cfg.rx_dma_channel = 0;
  spi_cfg.clock_polarity = SPI_CLOCK_POLARITY_IDLE_LOW;
  spi_cfg.clock_phase = SPI_CLOCK_PHASE_1ST_EDGE;

  rc = spi_device_init(&spi_cfg, &spi_slave_dev);
  if (atlk_error(rc)) {
    fprintf(stderr, "spi_device_init: %s\n", atlk_rc_to_str(rc));
  }

  return;
}

static void
spi_slave_callback(spi_dma_transfer_t *transfer)
{
  (void)transfer;
  spi_slave_wake_up();
}

void
spi_slave_test(void)
{
  atlk_rc_t rc = ATLK_OK;
  spi_dma_transfer_t transfer = SPI_DMA_TRANSFER_INIT;
  char cache_line_aligned msg_tx[CACHE_LINE_SIZE];
  char cache_line_aligned msg_rx[CACHE_LINE_SIZE];

  transfer.device_ptr = spi_slave_dev;

  /* set buffers */
  strncpy(msg_tx, EXAMPLE_SPI_MSG_SLAVE_2_MASTER, sizeof(msg_tx) - 1);
  memset(msg_rx, 0, sizeof(msg_rx));

  /* set transaction parameters */
  transfer.tx_buffer_ptr = NULL;
  transfer.tx_buffer_size = 0;
  transfer.rx_buffer_ptr = msg_rx;
  transfer.rx_buffer_size = sizeof(msg_rx);
  transfer.data_size = sizeof(EXAMPLE_SPI_MSG_MASTER_2_SLAVE);
  transfer.completion_handler = spi_slave_callback;

  /* Start transaction */
  rc = spi_dma_transfer_start(&transfer);
  if (atlk_error(rc)) {
    fprintf(stderr, "spi_dma_transfer_start: %s\n", atlk_rc_to_str(rc));
    return;
  }

  /* Wait for DMA to end transfer */
  spi_slave_sleep();

  if (strcmp(msg_rx, EXAMPLE_SPI_MSG_MASTER_2_SLAVE) != 0) {
    printf("Wrong! Message has been received from Slave: %s expected:%s \n",
        msg_rx, EXAMPLE_SPI_MSG_MASTER_2_SLAVE);
  }
  else {
    printf("Message received from master: %s\n", msg_rx);
  }

  /* set transaction parameters */
  transfer.tx_buffer_ptr = msg_tx;
  transfer.tx_buffer_size = sizeof(msg_tx);
  transfer.rx_buffer_ptr = NULL;
  transfer.rx_buffer_size = 0;
  transfer.data_size = sizeof(EXAMPLE_SPI_MSG_SLAVE_2_MASTER);
  transfer.completion_handler = spi_slave_callback;

  /* Start transaction */
  rc = spi_dma_transfer_start(&transfer);
  if (atlk_error(rc)) {
    fprintf(stderr, "spi_dma_transfer_start: %s\n", atlk_rc_to_str(rc));
    return;
  }

  /* Wait for DMA to end transfer */
  spi_slave_sleep();

}

static void
spi_slave_thread_entry(ULONG opaque)
{
  (void)opaque;

  spi_slave_init_spi();

  spi_slave_test();
}

void
craton_user_init(void)
{
  ULONG trv;

  trv = tx_semaphore_create(&spi_slave_semaphore, "SPI slave semaphore", 0);
  assert(trv == TX_SUCCESS);

  trv = tx_thread_create(&spi_slave_thread, "SPI slave thread",
                         spi_slave_thread_entry, 0,
                         spi_slave_thread_stack, sizeof(spi_slave_thread_stack),
                         5, 5, TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  return;
}
```

8.42 craton-threadx/sys-alarm/sys-alarm-example.c

```
/* Copyright (C) 2016 Autotalks Ltd. */
#include <stdio.h>
#include <inttypes.h>
#include <assert.h>
#include <unistd.h>

#include <tx_api.h>

#include <craton/sys_alarm.h>

/*
  CRATON ThreadX System Alarm Example

  This example demonstrates basic usage of system alarm API.

  A thread is created in which we artificially simulate high CPU utilization,
  followed by high heap memory utilization.

  In each case, an alarm handler is called.
*/

#define EXAMPLE_THREAD_PRIORITY 40

#define EXAMPLE_THREAD_TIME_SLICE 0

#define EXAMPLE_THRESHOLD_PERCENT 80

#define SYS_ALARM_THREAD_PRIORITY 10

#define SYS_ALARM_THREAD_TIME_SLICE 0

#ifdef __CRATON_ARM

/* Whether simulating high CPU utilization is enabled */
static volatile int cpu_alarm_test_enabled = 1;

/* Whether simulating large heap utilization is enabled */
static volatile int heap_alarm_test_enabled = 1;

/* CPU test thread */
static TX_THREAD alarm_test_thread;
static uint8_t alarm_test_thread_stack[0x1000];
static void alarm_test_thread_entry(ULONG input);
```

```
/* System alarm handler */
static void alarm_handler(const sys_alarm_gauges_t *gauges)
{
  if (gauges->cpu_utilization_percent > EXAMPLE_THRESHOLD_PERCENT) {
    printf("System alarm! CPU utilization: %u%%\n\n",
           gauges->cpu_utilization_percent);
    cpu_alarm_test_enabled = 0;
  }

  if (gauges->heap_utilization_percent > EXAMPLE_THRESHOLD_PERCENT) {
    printf("System alarm! Heap utilization: %u%%\n\n",
           gauges->heap_utilization_percent);
    heap_alarm_test_enabled = 0;
  }
}


void craton_user_init(void)
{
  /* System alarm configuration */
  sys_alarm_config_t config = SYS_ALARM_CONFIG_INIT;
  /* ThreadX return value */
  ULONG trv = TX_SUCCESS;
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;

  /* Set system alarm configuration */
  config.alarm_thresholds.cpu_utilization_percent =
      EXAMPLE_THRESHOLD_PERCENT;
  config.alarm_thresholds.heap_utilization_percent =
      EXAMPLE_THRESHOLD_PERCENT;
  config.alarm_handler = alarm_handler;

  /* Set system alarm scheduling parameters */
  config.sched_params.priority = SYS_ALARM_THREAD_PRIORITY;
  config.sched_params.time_slice = SYS_ALARM_THREAD_TIME_SLICE;

  /* Initialize system alarm */
  rc = sys_alarm_init(&config);
  if (atlk_error(rc)) {
    fprintf(stderr, "sys_alarm_init: %s\n", atlk_rc_to_str(rc));
    return;
  }

  /* Create alarm test thread */
  trv = tx_thread_create(&alarm_test_thread, "alarm_test_thread",
                         alarm_test_thread_entry, 0,
                         alarm_test_thread_stack,
                         sizeof(alarm_test_thread_stack),
                         EXAMPLE_THREAD_PRIORITY,
                         EXAMPLE_THREAD_PRIORITY,
                         EXAMPLE_THREAD_TIME_SLICE,
                         TX_AUTO_START);
  assert(trv == TX_SUCCESS);
}

atlk_inline void
countdown(int count)
{
  for (int i = count; i >= 0; --i) {
    printf("%d\n", i);
    usleep(1000000);
  }
}

static void alarm_test_thread_entry(ULONG input)
{
  /* A counter */
  int cnt = 0;
  /* Not using input */
  (void)input;

  printf("Simulating high CPU utilization in ...\n");
  countdown(5);

  /* Artificially create high CPU utilization */
  while (cpu_alarm_test_enabled) {
    if (++cnt % 100000 == 0) {
      usleep(1);
    }
  }

  printf("Simulating large heap utilization in ...\n");
  countdown(5);

  /* Artificially create large heap utilization */
#ifdef __CRATON_NO_ARC
  void *foo = malloc(0x5500000);
#else /* __CRATON_NO_ARC */
  void *foo = malloc(0x2000000);
#endif /* __CRATON_NO_ARC */
  assert(foo);

  free(foo);
}

#else /* __CRATON_ARM */

void craton_user_init(void)
{
}

#endif /* __CRATON_ARM */
```

8.43 craton-threadx/v2x-emulator/v2x-emulator-example-common.h

```
/* Copyright (C) 2013-2015 Autotalks Ltd. */
#ifndef V2X_EMULATOR_EXAMPLE_COMMON_H
#define V2X_EMULATOR_EXAMPLE_COMMON_H

/* Example V2X emulator IMQ address */
#define V2X_EMULATOR_IMQ_ADDRESS 3

/* Example V2X emulator thread priority */
#define V2X_EMULATOR_THREAD_PRIORITY 20

/* Example V2X emulator thread time slice */
#define V2X_EMULATOR_TIME_SLICE 0

#endif /* V2X_EMULATOR_EXAMPLE_COMMON_H */
```

8.44 craton-threadx/v2x-emulator/v2x-emulator-over-udp-example.c

```
/* Copyright (C) 2013-2015 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>

#include <tx_api.h>
#include <nxd_bsd.h>
#include <libcli.h>

#include <atlk/sdk.h>

#include <craton/v2x_emulator_init.h>
#include <craton/net.h>
#include <craton/cli.h>

#include "v2x-emulator-example-common.h"

/*
  CRATON ThreadX V2X Emulator Example


  Notes!
  =========
  1. V2X Emulator is only available on CRATON ARM.
  3. The emulator leaves the choice of external interface to the User glue code.

  Basic emulator flow concept
  -----------------
  Function posfix send/receive are in respect to the v2x_emulator meaning:
  * The stack calls v2x_send to pass data to the V2X emulator.
    The emulator stores the data and waits for a v2x_emulator_receive
    to be called by the User glue code.
    At this point the User glue code can send the data in which ever format and
    no which ever interface.

  * Upon the reception of data by the User glue code on which ever interface
    a v2x_emulator_send should be called to pass data to the emulator.
    The emulator will store the data until v2x_receive is called by the stack.

  See figure below for details.

 ** v2x_send flow **
 ---------               ------------                        ----------
 |  V2X  |  v2x_send     |   V2X    |  v2x_emulator_receive  |  User  | send
 | Stack |------------->>| Emulator |---------------------->>|  Glue  |------->
 |       |               |          |                        |  Code  |
 ---------               ------------                        ---------- (Example
                                                                           UDP
 ** v2x_receive flow **                                              socket)
 ---------                 ------------                         ----------
 |  V2X  |  v2x_receive  |   V2X    |  v2x_emulator_send     |  User  | receive
 | Stack |<<-------------| Emulator |<<----------------------|  Glue  |<-------
 |       |               |          |                        |  Code  |
 ---------                 ------------                         ----------

  User glue can implement a connection with any type of interface such as:
    * UDP socket
    * Raw Ethernet socket
    * UART


  The example
  --------------

  This example demonstrates basic usage of V2X Emulator API for code running on
  top of CRATON processor with ThreadX RTOS.

  A UDP socket is created and used to transmit the v2x frame and other
  v2x attributes over the Ethrenet to a receiving server/board.

  V2X Emulator master_init is called to create and get the emulator which will
  be used to communicate with the external interface.

  V2X Emulator slave_init is called to start the emulator part which
  will be needed to communicate with the V2X stack (for multi core see
  v2x_service_user.c).

  Override of the v2x_default_service_get is required to enable the V2X stack
  to get an instance of v2x emulator in place of v2x hw service - i.e. RF I/F
  (see file v2x_service_user.c).

  Two threads are open the first is for UDP transmit, in it the
  v2x_emulator_receive is called waiting for a v2x_send command. The data is
  then transmitted to the UDP socket.

  The other thread is used to receive in coming UDP frames and switching
  form the received V2X interface to the second interface and then transferring
  them to via v2x_emulator_send to v2x_receive.

  Once image is loaded run cli command
            ate> v2x_emulator_start <IP addr>
      Where:
        - A single board look back test can be run by entering the
          boards IP addr.
        - Board to Board/Host can be performed by entering
          the second Board/Host IP.

*/

/* Example V2X Interface index */
#define V2X_FIRST_IF_INDEX 1
#define V2X_SECOND_IF_INDEX 2

/* Example V2X emulator thread priority */
#define V2X_EMULATOR_SEND_THREAD_PRIORITY 25
#define V2X_EMULATOR_RECEIVE_THREAD_PRIORITY 26

/* Example V2X emulator stack size */
#define V2X_EMULATOR_STACK_SIZE 8192

/* Example V2X emulator max message size */
#define V2X_EMULATOR_MAX_MSG_SIZE  2048

/* Example UDP port */
#define UDP_PORT  2009

/* V2X emulator thread */
static TX_THREAD example_v2x_emulator_send_thread;
static uint8_t example_v2x_emulator_send_thread_stack[V2X_EMULATOR_STACK_SIZE];

/* V2X emulator thread */
static TX_THREAD example_v2x_emulator_receive_thread;
static uint8_t example_v2x_emulator_receive_thread_stack[V2X_EMULATOR_STACK_SIZE];

/* Example V2X emulator */
static v2x_emulator_t *v2x_emulator;

/* Example UDP shared socket */
static int socket_fd;

/* Example V2X emulator UDP payload */
typedef struct {
  v2x_if_index_t if_index;
  v2x_protocol_t protocol;
  eui48_t dest_address;
  v2x_datarate_t datarate;
  v2x_power_dbm8_t power_dbm8;
  size_t data_size;
  uint8_t data[V2X_EMULATOR_MAX_MSG_SIZE];
}  udp_payload_t;

#ifdef __CRATON_NO_ARC
extern atlk_rc_t v2x_init_and_send();
#endif /* __CRATON_NO_ARC */

static atlk_rc_t
open_udp_connection(char *ip_addr_str)
{
  #define INADDR_NONE 0xFFFFFFFF
  uint32_t ip_addr;
  struct sockaddr_in _sockaddr_in;
  int rc;

  /* Convert string ip to uint  */
  ip_addr = inet_addr(ip_addr_str);

  /* Create socket */
  socket_fd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
  if (socket_fd == -1) {
    printf("Failed creating socket");
    return ATLK_E_UNSPECIFIED;
  }

  /* Bind socket */
  memset(&_sockaddr_in, 0, sizeof(_sockaddr_in));
  _sockaddr_in.sin_family = AF_INET;
  _sockaddr_in.sin_port = htons(UDP_PORT);
  _sockaddr_in.sin_addr.s_addr = INADDR_ANY;
  rc = bind(socket_fd,
           (struct sockaddr *)&_sockaddr_in,
            sizeof(_sockaddr_in));
  if (rc == -1) {
    printf("Failed binding socket");
    return ATLK_E_UNSPECIFIED;
  }

  /* Set dest ip and port */
  memset(&_sockaddr_in, 0, sizeof(_sockaddr_in));
  _sockaddr_in.sin_family = AF_INET;
  _sockaddr_in.sin_port = htons(UDP_PORT);
  _sockaddr_in.sin_addr.s_addr = ip_addr;
  rc = connect(socket_fd,
              (struct sockaddr *)&_sockaddr_in,
               sizeof(_sockaddr_in));
  if (rc == -1) {
    printf("Failed connecting socket");
    return ATLK_E_UNSPECIFIED;
  }

  return ATLK_OK;
}


static void
example_v2x_emulator_send_thread_entry(ULONG input)
{
  /* The V2X payload to send */
  udp_payload_t udp_payload;
  /* V2X receive params */
  v2x_receive_params_t receive_params =
      V2X_RECEIVE_PARAMS_INIT;
  /* ATLK return value */
  atlk_rc_t rc;
  int rv;

  (void)input;

  while(1) {

    /* Receive a UDP packet.  */
    rv = recv(socket_fd, &udp_payload, sizeof(udp_payload), 0);

    if (rv == -1) {
      printf("Failed to receive on UDP socket (rv=%d)", rv);
      return;
    }

    /* Extract remaining data */
    receive_params.dest_address    = udp_payload.dest_address;
    receive_params.datarate        = udp_payload.datarate;
    receive_params.power_dbm8      = udp_payload.power_dbm8;
    /* Switch V2X I/F such that if sent on one received on the other */
    if (udp_payload.if_index == V2X_FIRST_IF_INDEX) {
      udp_payload.if_index = V2X_SECOND_IF_INDEX;
    }
    else {
      udp_payload.if_index = V2X_FIRST_IF_INDEX;
    }

    /* Send received data to V2X API - v2x_receive */
    rc = v2x_emulator_send(v2x_emulator,
                           udp_payload.if_index,
                           &udp_payload.protocol,
                           udp_payload.data,
                           udp_payload.data_size,
                           &receive_params,
                           NULL);
    if (atlk_error(rc)) {
      fprintf(stderr, "Failed to send data to V2X emulator (rc=%d)\n", rc);
      return;
    }
  }
}

/* V2X emulator receive buffer */
uint8_t receive_data[V2X_EMULATOR_MAX_MSG_SIZE];

static void
example_v2x_emulator_receive_thread_entry(ULONG input)
{
  /* RX data size */
  size_t data_size;
  /* V2X egress/ingress I/F number */
  v2x_if_index_t egress_if_index;
  /* V2X protocol */
  v2x_protocol_t protocol = V2X_PROTOCOL_INIT;
  /* V2X send params */
  v2x_send_params_t send_params = V2X_SEND_PARAMS_INIT;
 /* The V2X payload to send */
  udp_payload_t udp_payload;
  /* ATLK return value */
  atlk_rc_t rc;
  /*  return value */
  ssize_t rv;

  (void)input;

  while(1) {

    /* Send data size */
    data_size = sizeof(receive_data);

    /* Receive data form V2X API - v2x_send */
    rc = v2x_emulator_receive(v2x_emulator,
                              &egress_if_index,
                              &protocol,
                              receive_data,
                              &data_size,
                              &send_params,
                              &atlk_wait_forever);
    if (atlk_error(rc)) {
      fprintf(stderr, "Failed to receive data from V2X emulator (rc=%d)\n", rc);
      return;
    }

    printf("* v2x_send     - Broadcast TX: I/F - %d, msg - %s\n",
           egress_if_index, receive_data);

    /* Build UDP payload to send */
    udp_payload.if_index    = egress_if_index;
    udp_payload.protocol     = protocol;
    udp_payload.dest_address = send_params.dest_address;
    udp_payload.datarate     = send_params.datarate;
    udp_payload.power_dbm8   = send_params.power_dbm8;

    /* Extract the data */
    udp_payload.data_size = data_size;
    memcpy(udp_payload.data, receive_data, data_size);

    data_size = sizeof(udp_payload_t) -
      V2X_EMULATOR_MAX_MSG_SIZE + udp_payload.data_size;

    rv = send(socket_fd, (CHAR *)&udp_payload, data_size, 0);
    if ((size_t)rv != data_size) {
      printf("Failed to send UDP packet (rv=%d)", rv);
      return;
    }
  }
```

```
}

atlk_rc_t
example_v2x_emulator(char *ip_addr)
{
  /* API return code */
  atlk_rc_t rc = ATLK_OK;
  /* ThreadX return value */
  UINT trv;

  /* Open a UDP connection  */
  rc = open_udp_connection(ip_addr);
  if (atlk_error(rc)) {
    fprintf(stderr, "Failed to create UDP connection (rc=%d)\n", rc);
    return rc;
  }

  /* Create V2X emulator send thread */
  trv = tx_thread_create(&example_v2x_emulator_send_thread,
                         "Example V2X emulator send thread",
                         example_v2x_emulator_send_thread_entry, 0,
                         example_v2x_emulator_send_thread_stack,
                         sizeof(example_v2x_emulator_send_thread_stack),
                         V2X_EMULATOR_SEND_THREAD_PRIORITY,
                         V2X_EMULATOR_SEND_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE,
                         TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  /* Create V2X emulator receive thread */
  trv = tx_thread_create(&example_v2x_emulator_receive_thread,
                         "Example V2X emulator receive thread",
                         example_v2x_emulator_receive_thread_entry, 0,
                         example_v2x_emulator_receive_thread_stack,
                         sizeof(example_v2x_emulator_receive_thread_stack),
                         V2X_EMULATOR_SEND_THREAD_PRIORITY,
                         V2X_EMULATOR_SEND_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE,
                         TX_AUTO_START);
  assert(trv == TX_SUCCESS);

#ifdef __CRATON_NO_ARC

  /* Initiate the V2X API for the test */
  rc = v2x_init_and_send();
  if (atlk_error(rc)) {
    fprintf(stderr, "Failed to init V2X API for example (rc=%d)\n", rc);
    return rc;
  }

#endif /* ! __CRATON_NO_ARC */

  return rc;
}

#define V2X_EMULATOR_CLI_HELP "Start V2X Emulator UDP example\n" \
  "usage: v2x_emulator_start <Destination IP>\n"

int cmd_example_v2x_emulator(struct cli_def *cli,
                             const char *command,
                             char *argv[],
                             int argc)
{
  /* Autotalks return code */
  atlk_rc_t rc;

  (void)command;

  if ('?' == argv[0][0]) {
    cli_print(cli, "%s", V2X_EMULATOR_CLI_HELP);
    return CLI_OK;
  }

  if (argc != 1) {
    cli_print(cli, "Invalid parameters number");
    cli_print(cli, "%s", V2X_EMULATOR_CLI_HELP);
    return CLI_ERROR;
  }

  rc = example_v2x_emulator(argv[0]);
  if (atlk_error(rc)) {
    cli_print(cli, "V2X emulator example failed (rc=%d)", rc);
    return CLI_ERROR;
  }

  return CLI_OK;
}
```

```
void craton_user_init(void)
{
  /* V2X emulator configuration */
  v2x_emulator_config_t emulator_config =
      V2X_EMULATOR_CONFIG_INIT;
  /* CLI instance */
  cli_instance_t *cli = NULL;
  /* CLI command handle */
  struct cli_command *command = NULL;
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;

  printf("\n\n\nExample V2X emulator\n");

  /* Configure v2x_emulator_master */
  emulator_config.sched_params.priority = V2X_EMULATOR_THREAD_PRIORITY;
  emulator_config.sched_params.time_slice = V2X_EMULATOR_TIME_SLICE;

#ifndef __CRATON_NO_ARC

  emulator_config.imq_address = V2X_EMULATOR_IMQ_ADDRESS;

#else

  /* Initiate the slave part of the emulator - emulator<->v2x stack*/
  rc = v2x_emulator_slave_init(&emulator_config);
  if (atlk_error(rc)) {
    fprintf(stderr, "v2x_receive: %s\n", atlk_rc_to_str(rc));
    return;
  }

#endif /* __CRATON_NO_ARC */

  /* Initiate the master part of the emulator - emulator<->external I/F */
  rc = v2x_emulator_master_init(&v2x_emulator, &emulator_config);
  if (atlk_error(rc)) {
    fprintf(stderr, "Failed to initialize V2X Emulator master (rc=%d)\n", rc);
    return;
  }

  /* Get CRATON UART CLI instance */
  rc = cli_instance_get(&cli, CLI_INSTANCE_TYPE_UART);
  if (atlk_error(rc)) {
    fprintf(stderr, "Failed to get cli instance (rc=%d)\n", rc);
    return;
  }

  /* Register example command */
  command = cli_register_command(cli, NULL, "v2x_emulator_start",
                                 cmd_example_v2x_emulator,
                                 PRIVILEGE_UNPRIVILEGED, MODE_ANY,
                                 "Start example of v2x_emulator");
  if (command == NULL) {
    fprintf(stderr, "Failed to register cli command (returned NULL)\n");
    return;
  }

  printf("To run v2x_emulator example >>\n%s \n\n\n", V2X_EMULATOR_CLI_HELP);
}
```

8.45 craton-threadx/v2x-emulator/v2x-service-user.c

```
/* Copyright (C) 2013-2015 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>

#include <tx_api.h>

#include <atlk/sdk.h>
#include <atlk/v2x_service.h>

#include <craton/v2x_emulator_init.h>

#include "v2x-emulator-example-common.h"

/*
  CRATON ThreadX V2X Emulator Example

  Following code is part of the V2X Emulator example.

  For details regarding the V2X Emulator example
  PLEASE see comments @sdk/example/craton-threadx/v2x-emulator
  The actual code in this file is base on the SDK V2X API example.
  See @sdk/example/craton-threadx/v2x/ for more details.

  V2X service is initiated and a socket opened.

  A receive thread is created for V2X receive and V2X send is called to
  broadcast frames.

  Transmit prints the message to broadcast and receive thread the
  message received
*/

/* Interface index for example messages */
#define SEND_IF_INDEX 1
#define RECEIVE_IF_INDEX 2

/* Protocol identifier for example messages */
#define EXAMPLE_PROTO_ID 0x102ULL

/* V2X RX thread priority */
#define V2X_RECEIVE_THREAD_PRIORITY 26

/* V2X RX thread priority */
#define V2X_STACK_SIZE 4096

/* Example message format string: Example <seq_num> */
static const char example_msg_fmt[] = "Example %" PRIu32;

/* Example message string maximum length */
static const size_t example_msg_size_max = sizeof(example_msg_fmt) + 10;

/* RX thread */
static TX_THREAD v2x_receive_thread;
static uint8_t v2x_receive_thread_stack[V2X_STACK_SIZE];

/* Shared V2X service */
static v2x_service_t *v2x_service = NULL;

/* Cleanup any allocated resources */
static void example_cleanup(v2x_socket_t *v2x_socket)
{
  v2x_socket_delete(v2x_socket);
  v2x_service_delete(v2x_service);
}

static void
v2x_receive_thread_entry(ULONG input)
{
  /* V2X socket for sending frames */
  static v2x_socket_t *v2x_socket = NULL;
  /* V2X socket configuration */
  v2x_socket_config_t socket_config = V2X_SOCKET_CONFIG_INIT;
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* Not using input */
  (void)input;

  /* Set socket configuration */
  socket_config.if_index = RECEIVE_IF_INDEX;
  socket_config.protocol.protocol_id = EXAMPLE_PROTO_ID;

  /* Create a V2X socket for frame receive */
  rc = v2x_socket_create(v2x_service,
                         &v2x_socket,
                         &socket_config);
  if (atlk_error(rc)) {
    fprintf(stderr, "v2x_socket_create: %s\n", atlk_rc_to_str(rc));
    goto out;
  }

  while (1) {
    /* RX buffer */
    char buf[example_msg_size_max];
    /* RX size */
    size_t size = sizeof(buf);
    /* Received V2X parameters */
    v2x_receive_params_t receive_params =
      V2X_RECEIVE_PARAMS_INIT;

    /* Receive frame (wait forever until it arrives) */
    rc = v2x_receive(v2x_socket, buf, &size, &receive_params,
                     &atlk_wait_forever);
    if (atlk_error(rc)) {
      fprintf(stderr, "v2x_receive: %s\n", atlk_rc_to_str(rc));
      goto out;
    }
    /* Obtain data as zero-terminated string */
    if (buf[size - 1] != '\0') {
      printf("* Bad message (not zero-terminated)\n");
    }
    else {
      printf("* v2x_receive  - Message RX  : I/F - %d, msg - %s\n",
             socket_config.if_index, buf);
    }
  }

out:
  example_cleanup(v2x_socket);
}

atlk_rc_t
v2x_init_and_send(void)
{
  /* V2X socket for sending frames */
  static v2x_socket_t *v2x_socket = NULL;
  /* V2X socket configuration */
  v2x_socket_config_t socket_config = V2X_SOCKET_CONFIG_INIT;
  /* Message counter */
  uint32_t msg_count = 0;
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* ThreadX return value */
  ULONG trv = TX_SUCCESS;

  /* Get default V2X service instance */
  rc = v2x_default_service_get(&v2x_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "Failed to get V2X default service (rc=%d)\n",rc);
    v2x_service_delete(v2x_service);
    return rc;
  }

  /* Create V2X API receive thread */
  trv = tx_thread_create(&v2x_receive_thread, "Example V2X UPD RX thread",
                         v2x_receive_thread_entry,
                         0,
                         v2x_receive_thread_stack,
                         sizeof(v2x_receive_thread_stack),
                         V2X_RECEIVE_THREAD_PRIORITY,
                         V2X_RECEIVE_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE,
                         TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  /* Set socket configuration */
  socket_config.if_index = SEND_IF_INDEX;
  socket_config.protocol.protocol_id = EXAMPLE_PROTO_ID;

  /* Create a V2X socket for frame send */
  rc = v2x_socket_create(v2x_service,
                         &v2x_socket,
                         &socket_config);
  if (atlk_error(rc)) {
    fprintf(stderr, "v2x_socket_create: %s\n", atlk_rc_to_str(rc));
    goto out;
  }

  while (1) {
    /* Send parameters */
    v2x_send_params_t send_params = V2X_SEND_PARAMS_INIT;
    /* TX buffer */
    char buf[example_msg_size_max];

    /* Print message into buffer (with terminating \0) and update its size */
    size_t size = 1 + snprintf(buf, sizeof(buf), example_msg_fmt, msg_count);
    msg_count++;


    /* Transmit V2X PDU */
    rc = v2x_send(v2x_socket, buf, size, &send_params, NULL);
    if (atlk_error(rc)) {
      fprintf(stderr, "Failed to send frame over v2x_send (rc=%d)\n", rc);
      goto out;
    }

    /* Sleep 10 second between transmissions */
    usleep(10000000);
   }

out:
  example_cleanup(v2x_socket);
  return rc;
```

```
}

#ifdef __CRATON_ARC1
void craton_user_init(void)
{
  /* V2X emulator configuration */
  v2x_emulator_config_t emulator_config =
      V2X_EMULATOR_CONFIG_INIT;
  /* API return code */
  atlk_rc_t rc = ATLK_OK;

  /* Configure v2x_emulator_master */
  emulator_config.sched_params.priority = V2X_EMULATOR_THREAD_PRIORITY;
  emulator_config.sched_params.time_slice = V2X_EMULATOR_TIME_SLICE;
  emulator_config.imq_address = V2X_EMULATOR_IMQ_ADDRESS;


  /* Create emulator */
  rc = v2x_emulator_slave_init(&emulator_config);
  if (atlk_error(rc)) {
    fprintf(stderr, "v2x_receive: %s\n", atlk_rc_to_str(rc));
    return;
  }

  /* Initiate the V2X API for the test */
  rc = v2x_init_and_send();
  if (atlk_error(rc)) {
    fprintf(stderr, "Failed to init V2X API for example (rc=%d)\n", rc);
    return;
  }

  return;
}
#endif /* __CRATON_ARC1 */

/* Override the implementation of v2x_default_service_get */
atlk_rc_t
v2x_default_service_get(v2x_service_t **service_ptr)
{

#if defined(__CRATON_NO_ARC) || defined(__CRATON_ARC1)

  return v2x_emulator_service_get(service_ptr);

#else

  return v2x_imq_service_get(service_ptr);

#endif /* __CRATON_NO_ARC || __CRATON_ARC1 */

}
```

8.46 craton-threadx/v2x/v2x-example.c

```
/* Copyright (C) 2013-2015 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>

#include <tx_api.h>

#include <atlk/v2x.h>
#include <atlk/v2x_service.h>

/*
  CRATON ThreadX V2X Example

  This example demonstrates basic usage of V2X API for code running on top
  of CRATON processor with ThreadX RTOS.

  Two threads are created -- a TX thread and a RX thread. A V2X service is
  retrieved and a V2X socket is created; these are used by both threads.

  The TX thread sends a broadcast frame with protocol ID 0x102. The RX thread
  receives frames with protocol ID 0x102 and prints their content as well as
  receive power.
*/

/* Example threads priorities */
#define TX_THREAD_PRIORITY 40
#define RX_THREAD_PRIORITY 41

#if defined __CRATON_NO_ARC || defined __CRATON_ARC1
```

```
/* TX thread */
static TX_THREAD tx_thread;
static uint8_t tx_thread_stack[0x1000];
static void tx_thread_entry(ULONG input);

/* RX thread */
static TX_THREAD rx_thread;
static uint8_t rx_thread_stack[0x1000];
static void rx_thread_entry(ULONG input);

/* Interface index used in this example */
#define IF_INDEX 1

/* Protocol identifier used in this example */
#define PROTO_ID 0x102ULL

/* Shared V2X socket */
static v2x_socket_t *v2x_socket = NULL;

/* Example message format string: Example <seq_num> */
static const char msg_fmt[] = "Example %" PRIu32;

/* Example message string maximum length */
static const size_t msg_size_max = sizeof(msg_fmt) + 10;

void craton_user_init(void)
{
  /* ThreadX return value */
  ULONG trv = TX_SUCCESS;
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* V2X socket configuration */
  v2x_socket_config_t socket_config = V2X_SOCKET_CONFIG_INIT;
  /* V2X service */
  v2x_service_t *v2x_service = NULL;

  /* Get default V2X service instance */
  rc = v2x_default_service_get(&v2x_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "v2x_default_service_get: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Set socket configuration */
  socket_config.if_index = IF_INDEX;
  socket_config.protocol.protocol_id = PROTO_ID;

  /* Create a V2X socket */
  rc = v2x_socket_create(v2x_service, &v2x_socket, &socket_config);
  if (atlk_error(rc)) {
    fprintf(stderr, "v2x_socket_create: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Create TX thread */
  trv = tx_thread_create(&tx_thread, "tx_thread",
                         tx_thread_entry, 0,
                         tx_thread_stack,
                         sizeof(tx_thread_stack),
                         TX_THREAD_PRIORITY,
                         TX_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  /* Create RX thread */
  trv = tx_thread_create(&rx_thread, "rx_thread",
                         rx_thread_entry, 0,
                         rx_thread_stack,
                         sizeof(rx_thread_stack),
                         RX_THREAD_PRIORITY,
                         RX_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  return;

error:
  /* Clean-up resources */
  v2x_socket_delete(v2x_socket);
  v2x_service_delete(v2x_service);

  return;
}

void tx_thread_entry(ULONG input)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* Send parameters */
  v2x_send_params_t send_params = V2X_SEND_PARAMS_INIT;
  /* Message counter */
  uint32_t msg_count = 0;
  /* Not using input */
  (void)input;

  /* Set transmit power to -10 dBm */
  send_params.power_dbm8 = -80;

  while (1) {
    /* TX buffer */
    char buf[msg_size_max];

    /* Print message into buffer (with terminating \0) and update its size */
    size_t size = 1 + snprintf(buf, sizeof(buf), msg_fmt, msg_count);
    msg_count++;

    printf("Example: Broadcast TX: \"%s\"\n", buf);

    /* Transmit V2X PDU */
    rc = v2x_send(v2x_socket, buf, size, &send_params, NULL);
    if (atlk_error(rc)) {
      fprintf(stderr, "v2x_send: %s\n", atlk_rc_to_str(rc));
      return;
    }

    /* Sleep 1 second between transmissions */
    usleep(1000000);
  }
}

void rx_thread_entry(ULONG input)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* Not using input */
  (void)input;

  while (1) {
    /* RX buffer */
    char buf[msg_size_max];
    /* RX size */
    size_t size = sizeof(buf);
    /* Received V2X parameters */
    v2x_receive_params_t receive_params =
      V2X_RECEIVE_PARAMS_INIT;

    /* Receive frame (wait forever until it arrives) */
    rc = v2x_receive(v2x_socket, buf, &size, &receive_params,
                     &atlk_wait_forever);
    if (atlk_error(rc)) {
      fprintf(stderr, "v2x_receive: %s\n", atlk_rc_to_str(rc));
      return;
    }

    /* Print source address of received frame */
    printf("Example: RX from %02x:%02x:%02x:%02x:%02x:%02x\n",
           receive_params.source_address.octets[0],
           receive_params.source_address.octets[1],
           receive_params.source_address.octets[2],
           receive_params.source_address.octets[3],
           receive_params.source_address.octets[4],
           receive_params.source_address.octets[5]);

    /* Obtain data as zero-terminated string */
    if (buf[size - 1] != '\0') {
      printf("* Bad message (not zero-terminated)\n");
    }
    else {
      printf("* Message: \"%s\"\n", buf);
    }

    /* Print RX power */
    if (receive_params.power_dbm8 != V2X_POWER_DBM8_NA) {
      printf("* RX power: %.2f dBm\n",
             (double)receive_params.power_dbm8 /
      V2X_POWER_DBM8_PER_DBM);
    }
  }
}

#else /* __CRATON_NO_ARC || __CRATON_ARC1 */

void craton_user_init(void)
```

```
{
}

#endif /* __CRATON_NO_ARC || __CRATON_ARC1 */
```

8.47 craton-threadx/wave-ipv6/wave-ipv6-client-example.c

```
/* Copyright (C) 2016 Autotalks Ltd. */
#include <assert.h>
#include <stdio.h>
#include <unistd.h>

#include <tx_api.h>
#include <nx_api.h>

#include <atlk/v2x_service.h>
#include <atlk/v2x.h>

#include <craton/net.h>
#include <craton/wave_ipv6.h>

#include "wave_ipv6_common.h"

/*
  CRATON ThreadX IPv6 TCP Client example

  This example demonstrates basic usage of TCP protocol over IPv6 for code
  running on top of CRATON processor with ThreadX RTOS.

  Client thread is created. A TCP socket is created and connected to
  any port (bind) and after it connected to server.

  The Client thread sends TCP frames to server "Hello from client" and waits
  for response from server.

  The client side should be started after server side.
*/

/* Pointer to `untrusted` network IP instance */
static NX_IP *untrusted_instance = NULL;
/* NetX TCP socket */
static NX_TCP_SOCKET client_socket;
/* client thread */
static TX_THREAD client_thread;
static void thread_client_entry(ULONG thread_input);
/* thread stack */
static uint8_t example_client_thread_stack[0x1000];
/* Example thread priorities */
#define EXAMPLE_CLIENT_THREAD_PRIORITY 40

/* Example message format string */
#define EXAMPLE_CLIENT_MESSAGE "Hello from client"

void craton_user_init(void)
{
  /* NetX return value */
  UINT nrc;
  /* ThreadX return value */
  ULONG trc = TX_SUCCESS;
  /* Auto-talks return code */
  atlk_rc_t rc;
  /* IPv6 address */
  NXD_ADDRESS ip_address;

  v2x_service_t *v2x_service = NULL;

  /* Get default V2X service instance */
  rc = v2x_default_service_get(&v2x_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "v2x_default_service_get: %s\n", atlk_rc_to_str(rc));
    return;
  }

  /* Set V2X access profile of untrusted network interface #0 */
  v2x_netif_profile_t netif_profile = {
    .if_index = EXAMPLE_IF_INDEX,
    .channel_id = V2X_CHANNEL_ID_INIT,
    .datarate = V2X_DATARATE_6MBPS,
    .power_dbm8 = -80
  };

  rc = v2x_netif_profile_set(v2x_service, 0, &netif_profile);
  if (atlk_error(rc)) {
    fprintf(stderr, "v2x_netif_profile_set: %s\n", atlk_rc_to_str(rc));
    return;
  }
  /* Get `untrusted` IP instance */
  rc = net_ip_untrusted_instance_get(&untrusted_instance);
  assert(rc == ATLK_OK);

  /* Enable IPv6 */
  nrc = nxd_ipv6_enable(untrusted_instance);
  assert(nrc == NX_SUCCESS);

  /* Enable both ICMPv4 and ICMPv6 */
  nrc = nxd_icmp_enable(untrusted_instance);
  assert(nrc == NX_SUCCESS);

  ip_address.nxd_ip_version = NX_IP_VERSION_V6;
  ip_address.nxd_ip_address.v6[0] = 0x20010000;
  ip_address.nxd_ip_address.v6[1] = 0;
  ip_address.nxd_ip_address.v6[2] = 0;
  ip_address.nxd_ip_address.v6[3] = 0x11;

  nrc = nxd_ipv6_global_address_set(untrusted_instance,
                                    &ip_address,
                                    64);
  assert(nrc == NX_SUCCESS);

  /* Create the main thread.  */
  trc = tx_thread_create(&client_thread,
                         "thread client",
                          thread_client_entry,
                          0,
                          example_client_thread_stack,
                          sizeof(example_client_thread_stack),
                          EXAMPLE_CLIENT_THREAD_PRIORITY,
                          EXAMPLE_CLIENT_THREAD_PRIORITY,
                          TX_NO_TIME_SLICE,
                          TX_AUTO_START);
  assert(trc == TX_SUCCESS);
}

/* Define the test threads.  */
void thread_client_entry(ULONG thread_input)
{
  /* Not using input */
  (void)thread_input;
  /* NetX return value */
  UINT nrc;
  /* NetX TCP packet pointer for TX and RX */
  NX_PACKET *tx_packet;
  NX_PACKET *rx_packet;
  /* packet length */
  ULONG length;
  /* IPv6 server address */
  NXD_ADDRESS server_ipv6_address;
  /* NetX packet pool pointer */
  NX_PACKET_POOL *packet_pool = NULL;

  /* Wait 5 seconds for the IP thread to finish its initialization and
     for the IPv6 stack to finish DAD process. */
  usleep(5000000);

  printf("\n***Starting TCP Client***\n");

  server_ipv6_address.nxd_ip_version = NX_IP_VERSION_V6;
  server_ipv6_address.nxd_ip_address.v6[0] = 0x20010000;
  server_ipv6_address.nxd_ip_address.v6[1] = 0;
  server_ipv6_address.nxd_ip_address.v6[2] = 0;
  server_ipv6_address.nxd_ip_address.v6[3] = SERVER_ADDRESS;

  packet_pool = untrusted_instance->nx_ip_default_packet_pool;
  assert(packet_pool);

  /* Create a socket.  */
  nrc = nx_tcp_socket_create(untrusted_instance,
                             &client_socket,
                             "Client Socket",
                             NX_IP_NORMAL,
                             NX_FRAGMENT_OKAY,
                             NX_IP_TIME_TO_LIVE,
                             200,
                             NX_NULL, NX_NULL);
  assert(nrc == NX_SUCCESS);

  nrc =  nx_tcp_client_socket_bind(&client_socket,
                                   NX_ANY_PORT,
                                   NX_WAIT_FOREVER);
  assert(nrc == NX_SUCCESS);

  /* Attempt to connect the socket.  */
  nrc = nxd_tcp_client_socket_connect(&client_socket,
                                    &server_ipv6_address,
                                    EXAMPLE_SERVER_PORT,
                                    NX_WAIT_FOREVER);
  assert(nrc == NX_SUCCESS);

  printf("Connection established\n");

  /* Loop to send and receive packets  */
  while (1) {
    /* Allocate a packet.  */
    nrc = nx_packet_allocate(packet_pool,
                             &tx_packet,
                             NX_TCP_PACKET,
                             NX_WAIT_FOREVER);
    assert(nrc == NX_SUCCESS);

    /* Write ABCs into the packet payload!  */
    nx_packet_data_append(tx_packet,
                       EXAMPLE_CLIENT_MESSAGE,
                       sizeof(EXAMPLE_CLIENT_MESSAGE),
                       packet_pool,
                       TX_WAIT_FOREVER);

    nrc = nx_packet_length_get(tx_packet, &length);
    if ((nrc) || (length != sizeof(EXAMPLE_CLIENT_MESSAGE))) {
      assert(0);
    }

    /* Send the packet out!  */
    nrc = nx_tcp_socket_send(&client_socket, tx_packet, TX_WAIT_FOREVER);
    assert(nrc == NX_SUCCESS);

    /* Wait for packet from server  */
    nrc = nx_tcp_socket_receive(&client_socket,
                                &rx_packet,
                                NX_WAIT_FOREVER);
    assert(nrc == NX_SUCCESS);

    printf("Client received: \"%s\"\n", rx_packet->nx_packet_prepend_ptr);
    nx_packet_release(rx_packet);

    /* wait 1 second */
    usleep(1000000);
  }
}
```

8.48 craton-threadx/wave-ipv6/wave-ipv6-example.c

```
/* Copyright (C) 2014-2015 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>

#include <tx_api.h>
#include <nx_api.h>
#include <craton/net.h>

#include <atlk/v2x_service.h>
#include <atlk/v2x.h>
#include <craton/wave_ipv6.h>

/*
  CRATON ThreadX WAVE IPv6 Example

  This example demonstrates basic usage of WAVE IPv6 for code running on
  top of CRATON processor with ThreadX RTOS.

  Two threads are created -- a TX thread and a RX thread. A UDP socket is
  created per thread.

  The TX thread sends multicast UDP frames while the RX thread receives UDP
  frames and prints their content.

  @todo: This example is not currently supported in multi-core SDK.
*/

#define EXAMPLE_IF_INDEX 1

/* UDP server and client ports */
#define EXAMPLE_SERVER_PORT 6666
#define EXAMPLE_CLIENT_PORT 6667

/* Example thread priorities */
#define EXAMPLE_TX_THREAD_PRIORITY 40
#define EXAMPLE_RX_THREAD_PRIORITY 41
```

```
/* TX thread */
static TX_THREAD example_tx_thread;
static uint8_t example_tx_thread_stack[0x1000];
static void example_tx_thread_entry(ULONG input);

/* RX thread */
static TX_THREAD example_rx_thread;
static uint8_t example_rx_thread_stack[0x1000];
static void example_rx_thread_entry(ULONG input);

/* Pointer to 'untrusted' network IP instance */
static NX_IP *untrusted_instance = NULL;

/* Example message format string: Example <seq_num> */
static const char example_msg_fmt[] = "Example %" PRIu32;

/* Example message string maximum length */
static const size_t example_msg_size_max = sizeof(example_msg_fmt) + 10;


void craton_user_init(void)
{
  /* ThreadX return value */
  ULONG trv = TX_SUCCESS;
  /* NetX return value */
  ULONG nrv = NX_SUCCESS;
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* IPv6 address */
  NXD_ADDRESS ipv6_address;
  /* IPv4 address */
  ULONG ipv4_address;
  /* V2X service */
  v2x_service_t *v2x_service = NULL;

  /* Get default V2X service instance */
  rc = v2x_default_service_get(&v2x_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "v2x_default_service_get: %s\n", atlk_rc_to_str(rc));
    return;
  }

  /* Set V2X access profile of untrusted network interface #0 */
  v2x_netif_profile_t netif_profile = {
    .if_index = EXAMPLE_IF_INDEX,
    .channel_id = V2X_CHANNEL_ID_INIT,
    .datarate = V2X_DATARATE_6MBPS,
    .power_dbm8 = -80
  };

  rc = v2x_netif_profile_set(v2x_service, 0, &netif_profile);
  if (atlk_error(rc)) {
    fprintf(stderr, "v2x_netif_profile_set: %s\n", atlk_rc_to_str(rc));
    return;
  }

  /* Get 'untrusted' IP instance */
  rc = net_ip_untrusted_instance_get(&untrusted_instance);
  if (atlk_error(rc)) {
    fprintf(stderr, "net_ip_untrusted_instance_get: %s\n", atlk_rc_to_str(rc));
    return;
  }

  /* Enable IPv6 on the 'untrusted' IP instance */
  nrv = nxd_ipv6_enable(untrusted_instance);
  assert(nrv == NX_SUCCESS);

  /* Get IPv4 address of the 'untrusted' IP instance */
  nrv = nx_ip_interface_info_get(untrusted_instance, 0, NULL, &ipv4_address,
                                 NULL, NULL, NULL, NULL);
  assert(nrv == NX_SUCCESS);

  /* Set IPv6 address based on retrieved IPv4 address */
  ipv6_address.nxd_ip_version = NX_IP_VERSION_V6;
  ipv6_address.nxd_ip_address.v6[0] = 0x20010000;
  ipv6_address.nxd_ip_address.v6[1] = 0x0;
  ipv6_address.nxd_ip_address.v6[2] = 0x0;
  ipv6_address.nxd_ip_address.v6[3] = ipv4_address;

  /* Set IPv6 address of 'untrusted' IP instance */
  nrv = nxd_ipv6_global_address_set(untrusted_instance, &ipv6_address, 64);
  assert(nrv == NX_SUCCESS);

  /* Create TX thread */
  trv = tx_thread_create(&example_tx_thread, "example_tx_thread",
                         example_tx_thread_entry, 0,
                         example_tx_thread_stack,
                         sizeof(example_tx_thread_stack),
                         EXAMPLE_TX_THREAD_PRIORITY,
                         EXAMPLE_TX_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  /* Create RX thread */
  trv = tx_thread_create(&example_rx_thread, "example_rx_thread",
                         example_rx_thread_entry, 0,
                         example_rx_thread_stack,
                         sizeof(example_rx_thread_stack),
                         EXAMPLE_RX_THREAD_PRIORITY,
                         EXAMPLE_RX_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  return;
}

void example_tx_thread_entry(ULONG input)
{
  /* NetX return value */
  ULONG nrv = NX_SUCCESS;
  /* NetX UDP socket */
  NX_UDP_SOCKET udp_socket;
  /* NetX UDP packet pointer */
  NX_PACKET *udp_packet = NULL;
  /* NetX packet pool pointer */
  NX_PACKET_POOL *packet_pool = NULL;
  /* IPv6 multicast address */
  NXD_ADDRESS ipv6_multicast_address;
  /* Message counter */
  uint32_t msg_count = 0;
  /* Not using input */
  (void)input;

  /* Point to the `untrusted` IP instance packet pool */
  packet_pool = untrusted_instance->nx_ip_default_packet_pool;

  /* Arbitrary multicast IPv6 address */
  ipv6_multicast_address.nxd_ip_version = NX_IP_VERSION_V6;
  ipv6_multicast_address.nxd_ip_address.v6[0] = 0xFF020000;
  ipv6_multicast_address.nxd_ip_address.v6[1] = 0x0;
  ipv6_multicast_address.nxd_ip_address.v6[2] = 0x0;
  ipv6_multicast_address.nxd_ip_address.v6[3] = 0x1;

  /* Create a UDP socket for sending UDP packets */
  nrv = nx_udp_socket_create(untrusted_instance, &udp_socket, "udp_tx_socket",
                             NX_IP_NORMAL, NX_FRAGMENT_OKAY, 0x80, 5);
  assert(nrv == NX_SUCCESS);

  /* Bind UDP socket to example client port */
  nrv = nx_udp_socket_bind(&udp_socket, EXAMPLE_CLIENT_PORT, TX_WAIT_FOREVER);
  assert(nrv == NX_SUCCESS);

  while (1) {
    /* TX buffer */
    char buf[example_msg_size_max];

    /* Print message into buffer (with terminating \0) and update its size */
    size_t size = 1 + snprintf(buf, sizeof(buf), example_msg_fmt, msg_count);
    msg_count++;

    /* Allocate a packet from the packet pool */
    nrv = nx_packet_allocate(packet_pool, &udp_packet, NX_UDP_PACKET,
                             TX_WAIT_FOREVER);
    assert(nrv == NX_SUCCESS);

    /* Write an arbitrary message */
    nx_packet_data_append(udp_packet, buf, size, packet_pool, TX_WAIT_FOREVER);

    /* Send packet to the server */
    nrv = nxd_udp_socket_send(&udp_socket, udp_packet, &ipv6_multicast_address,
                              EXAMPLE_SERVER_PORT);
    assert(nrv == NX_SUCCESS);

    /* Sleep 1 second between transmissions */
    usleep(1000000);
  }
}

void example_rx_thread_entry(ULONG input)
{
  /* NetX return value */
  ULONG nrv = NX_SUCCESS;
  /* NetX UDP socket */
  NX_UDP_SOCKET udp_socket;
  /* NetX UDP packet */
  NX_PACKET *udp_packet = NULL;
  /* Not using input */
  (void)input;

  /* Create a UDP socket for receiving UDP packets */
  nrv = nx_udp_socket_create(untrusted_instance, &udp_socket, "udp_rx_socket",
                             NX_IP_NORMAL, NX_FRAGMENT_OKAY, 0x80, 5);
  assert(nrv == NX_SUCCESS);

  /* Bind UDP socket to example server port */
  nrv = nx_udp_socket_bind(&udp_socket, EXAMPLE_SERVER_PORT, TX_WAIT_FOREVER);
  assert(nrv == NX_SUCCESS);

  while (1) {
    /* Receive a UDP packet */
    nrv = nx_udp_socket_receive(&udp_socket, &udp_packet, TX_WAIT_FOREVER);
    assert(nrv == NX_SUCCESS);

    /* Print length and first bytes of received packet */
    if (udp_packet->nx_packet_prepend_ptr[udp_packet->nx_packet_length - 1]
        != '\0') {
      printf("Received a bad message (not zero-terminated)\n");
    }
    else {
      printf("Received message: \"%s\"\n", udp_packet->nx_packet_prepend_ptr);
    }

    /* Release the packet */
    nx_packet_release(udp_packet);
  }
}
```

8.49 craton-threadx/wave-ipv6/wave-ipv6-server-example.c

```
/* Copyright (C) 2016 Autotalks Ltd. */
#include <assert.h>
#include <stdio.h>
#include <unistd.h>

#include <tx_api.h>
#include <nx_api.h>

#include <atlk/v2x_service.h>
#include <atlk/v2x.h>

#include <craton/net.h>
#include <craton/wave_ipv6.h>

#include "wave_ipv6_common.h"

/*
  CRATON ThreadX IPv6 TCP Server example

  This example demonstrates basic usage of TCP protocol over IPv6 for code
  running on top of CRATON processor with ThreadX RTOS.

  Server thread is created. A TCP socket is created and listen to client port,
  after client connection server accept connection.

  The Server thread sends back TCP frames to client "Hello from server".
*/

/* Pointer to `untrusted` network IP instance */
static NX_IP *untrusted_instance = NULL;
/* NetX TCP socket */
static NX_TCP_SOCKET  server_socket;
/* server thread */
static TX_THREAD server_thread;
static void thread_server_entry(ULONG thread_input);
/* thread stack */
static uint8_t example_server_thread_stack[0x1000];
/* Example thread priorities */
#define EXAMPLE_SERVER_THREAD_PRIORITY 40

static void thread_server_connect_received(NX_TCP_SOCKET *server_socket, UINT port);
static void thread_server_disconnect_received(NX_TCP_SOCKET *server_socket);

/* Example message format string */
#define EXAMPLE_SERVER_MESSAGE "Hello from server"

void craton_user_init(void)
{
  /* NetX return value */
  UINT nrc;
  /* ThreadX return value */
  ULONG trc = TX_SUCCESS;
  /* Auto-talks return code */
  atlk_rc_t rc;
  /* IPv6 address */
  NXD_ADDRESS ip_address;

  v2x_service_t *v2x_service = NULL;

  /* Get default V2X service instance */
  rc = v2x_default_service_get(&v2x_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "v2x_default_service_get: %s\n", atlk_rc_to_str(rc));
    return;
  }

  /* Set V2X access profile of untrusted network interface #0 */
  v2x_netif_profile_t netif_profile = {
    .if_index = EXAMPLE_IF_INDEX,
    .channel_id = V2X_CHANNEL_ID_INIT,
    .datarate = V2X_DATARATE_6MBPS,
    .power_dbm8 = -80
  };

  rc = v2x_netif_profile_set(v2x_service, 0, &netif_profile);
  if (atlk_error(rc)) {
    fprintf(stderr, "v2x_netif_profile_set: %s\n", atlk_rc_to_str(rc));
    return;
  }
  /* Get `untrusted` IP instance */
  rc = net_ip_untrusted_instance_get(&untrusted_instance);
  assert(rc == ATLK_OK);

    /* Enable IPv6 */
  nrc = nxd_ipv6_enable(untrusted_instance);
  assert(nrc == NX_SUCCESS);

  /* Enable both ICMPv4 and ICMPv6 */
  nrc = nxd_icmp_enable(untrusted_instance);
  assert(nrc == NX_SUCCESS);

  ip_address.nxd_ip_version = NX_IP_VERSION_V6;
  ip_address.nxd_ip_address.v6[0] = 0x20010000;
  ip_address.nxd_ip_address.v6[1] = 0;
  ip_address.nxd_ip_address.v6[2] = 0;
  ip_address.nxd_ip_address.v6[3] = SERVER_ADDRESS;

  nrc = nxd_ipv6_global_address_set(untrusted_instance,
                                    &ip_address,
                                    64);
  assert(nrc == NX_SUCCESS);

  /* Create the main thread.  */
  trc = tx_thread_create(&server_thread,
                         "thread server",
                         thread_server_entry,
                         0,
                         example_server_thread_stack,
                         sizeof(example_server_thread_stack),
                         EXAMPLE_SERVER_THREAD_PRIORITY,
                         EXAMPLE_SERVER_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE,
                         TX_AUTO_START);
  assert(trc == TX_SUCCESS);
}

void thread_server_entry(ULONG thread_input)
{
  /* Not using input */
  (void)thread_input;
  /* NetX return value */
  UINT nrc;
  /* NetX TCP packet pointer for TX and RX */
  NX_PACKET *tx_packet;
  NX_PACKET *rx_packet;
  /* status of IP instance */
  ULONG actual_status;
  /* packet length */
  ULONG length;
  /* NetX packet pool pointer */
  NX_PACKET_POOL *packet_pool = NULL;

  /* Wait 5 seconds for the IP thread to finish its initialization and
     for the IPv6 stack to finish DAD process. */
  usleep(5000000);

  printf("\n***Starting TCP Server***\n");
  /* Ensure the IP instance has been initialized.  */
  nrc = nx_ip_status_check(untrusted_instance,
                           NX_IP_INITIALIZE_DONE,
                           &actual_status,
                           100);
  assert(nrc == NX_SUCCESS);

  packet_pool = untrusted_instance->nx_ip_default_packet_pool;
  assert(packet_pool);

  /* Create a socket.  */
  nrc =  nx_tcp_socket_create(untrusted_instance,
                           &server_socket,
                           "Server Socket",
                           NX_IP_NORMAL,
                           NX_FRAGMENT_OKAY,
                           NX_IP_TIME_TO_LIVE,
                           100,
                           NX_NULL,
                           thread_server_disconnect_received);
  assert(nrc == NX_SUCCESS);

  /* Setup this thread to listen.  */
  nrc = nx_tcp_server_socket_listen(untrusted_instance,
                                 EXAMPLE_SERVER_PORT,
                                 &server_socket,
                                 5,
                                 thread_server_connect_received);
  assert(nrc == NX_SUCCESS);

  /* Accept a client socket connection.  */
  nrc = nx_tcp_server_socket_accept(&server_socket,
                                    NX_WAIT_FOREVER);
  assert(nrc == NX_SUCCESS);
  printf("Someone was connected\n");

  /* Loop to receive and send packets  */
  while(1) {
    /* Receive a TCP message from the socket.  */
    nrc = nx_tcp_socket_receive(&server_socket,
                             &rx_packet,
                             NX_WAIT_FOREVER);
    assert(nrc == NX_SUCCESS);

    printf("Server received: \"%s\"\n", rx_packet->nx_packet_prepend_ptr);
    nx_packet_release(rx_packet);

    /* Allocate a packet.  */
    nrc = nx_packet_allocate(packet_pool,
                          &tx_packet,
                          NX_TCP_PACKET,
                          NX_WAIT_FOREVER);
    assert(nrc == NX_SUCCESS);

    /* Write ABCs into the packet pay load!  */
    nx_packet_data_append(tx_packet,
                       EXAMPLE_SERVER_MESSAGE,
                       sizeof(EXAMPLE_SERVER_MESSAGE),
                       packet_pool,
                       TX_WAIT_FOREVER);

    nrc = nx_packet_length_get(tx_packet, &length);
    if ((nrc) || (length != sizeof(EXAMPLE_SERVER_MESSAGE))) {
      assert(0);
    }
    nrc = nx_tcp_socket_send(&server_socket,
                          tx_packet,
                          NX_WAIT_FOREVER);
    assert(nrc == NX_SUCCESS);
  }
}

void  thread_server_connect_received(NX_TCP_SOCKET *socket_ptr, UINT port)
{
  /* Not using input */
  (void)socket_ptr;
  (void)port;
  printf("Connection received\n");
}


void  thread_server_disconnect_received(NX_TCP_SOCKET *socket)
{
  /* Not using input */
  (void)socket;
  printf("Disconnection received\n");
```

```
}
```

8.50 craton-threadx/wlan-driver/traffic-monitor-example.c

```
/* Copyright (C) 2015 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>

#include <tx_api.h>

#include <craton/syslog.h>
#include <craton/wlan_driver.h>

#include <atlk/v2x.h>
#include <atlk/v2x_service.h>

/*
  CRATON WLAN Driver Traffic Monitor Example

  This example demonstrates basic usage of WLAN driver traffic monitor API.

  Two traffic monitor callbacks are registered: a RX callback and a TX
  callback. Both callbacks are registered on the first WLAN device.

  A single thread is spawned in which broadcast frames with protocol ID 0x102
  are sent on interface index 1. Received frames are discarded (to avoid
  running out of memory due to full RX queue).

  Example output is sent to Syslog at log level INFO. It is important to set
  Syslog sink to UDP so as traffic monitor callbacks will return quickly
  (printing to console takes a lot of time and should not be done within
  callbacks).
*/

/* Example threads priorities */
#define TX_THREAD_PRIORITY 40
#define RX_THREAD_PRIORITY 41

#if defined __CRATON_NO_ARC || defined __CRATON_ARC1

/* Transmitting thread */
static TX_THREAD thread;
static uint8_t thread_stack[0x1000];
static void thread_entry(ULONG input);

/* Shared V2X service */
static v2x_service_t *v2x_service = NULL;

/* Shared V2X socket */
static v2x_socket_t *v2x_socket = NULL;

/* Interface index for example messages */
#define IF_INDEX 1

/* Protocol identifier for example messages */
#define PROTO_ID 0x102ULL

/* Length of payload used in this example */
#define PAYLOAD_LENGTH 45

/* Cleanup any allocated resources */
static void cleanup(void)
{
  v2x_socket_delete(v2x_socket);
  v2x_service_delete(v2x_service);
}

/* WLAN device ID used by traffic monitor */
#define WLAN_DEV_ID 0

/* Callback called on each received frame */
static void
rx_callback(const wlan_frame_t *frame, const wlan_rx_frame_info_t *info);

/* Callback called on each sent frame */
static void
tx_callback(const wlan_frame_t *frame, const wlan_tx_frame_info_t *info);

void craton_user_init(void)
{
  /* ThreadX return value */
  ULONG trv = TX_SUCCESS;
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* V2X socket configuration */
  v2x_socket_config_t socket_config = V2X_SOCKET_CONFIG_INIT;

  /* Get default V2X service instance */
  rc = v2x_default_service_get(&v2x_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "v2x_default_service_get: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Set socket configuration */
  socket_config.if_index = IF_INDEX;
  socket_config.protocol.protocol_id = PROTO_ID;

  /* Create a V2X socket */
  rc = v2x_socket_create(v2x_service, &v2x_socket,
                         &socket_config);
  if (atlk_error(rc)) {
    fprintf(stderr, "v2x_socket_create: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Register traffic monitor TX callback */
  rc = wlan_tx_traffic_monitor_set(WLAN_DEV_ID, tx_callback);
  if (atlk_error(rc)) {
    fprintf(stderr, "wlan_tx_traffic_monitor_set: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Register traffic monitor RX callback */
  rc = wlan_rx_traffic_monitor_set(WLAN_DEV_ID, rx_callback);
  if (atlk_error(rc)) {
    fprintf(stderr, "wlan_rx_traffic_monitor_set: %s\n", atlk_rc_to_str(rc));
    goto error;
  }

  /* Create TX thread */
  trv = tx_thread_create(&thread, "thread",
                         thread_entry, 0,
                         thread_stack,
                         sizeof(thread_stack),
                         TX_THREAD_PRIORITY,
                         TX_THREAD_PRIORITY,
                         TX_NO_TIME_SLICE, TX_AUTO_START);
  assert(trv == TX_SUCCESS);

  return;

error:
  cleanup();
}

void thread_entry(ULONG input)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* Send parameters */
  v2x_send_params_t send_params = V2X_SEND_PARAMS_INIT;
  /* Receive parameters */
  v2x_receive_params_t receive_params =
      V2X_RECEIVE_PARAMS_INIT;
  /* Not using input */
  (void)input;

  /* Set transmit power to -10 dBm */
  send_params.power_dbm8 = -80;

  while (1) {
    /* Payload of all-zeros */
    char buf[PAYLOAD_LENGTH] = { 0 };
    /* Size of payload */
    size_t size = sizeof(buf);

    /* Transmit V2X PDU */
    rc = v2x_send(v2x_socket, buf, size, &send_params, NULL);
    if (atlk_error(rc)) {
      fprintf(stderr, "v2x_send: %s\n", atlk_rc_to_str(rc));
      goto error;
    }

    /* Recieve and discard frame - to avoid a full queue */
    size = sizeof(buf);
    rc = v2x_receive(v2x_socket, buf, &size, &receive_params, NULL);
    if (atlk_error(rc) && rc != ATLK_E_NOT_READY) {
      fprintf(stderr, "v2x_receive: %s\n", atlk_rc_to_str(rc));
      goto error;
    }
    /* Sleep 1 second between transmissions */
    usleep(1000000);
  }

error:
  cleanup();
}

/* Callback called on each received frame */
static void
rx_callback(const wlan_frame_t *frame, const wlan_rx_frame_info_t *info)
{
  syslog(LOG_INFO, "Received frame on device ID %u", info->
      device_id);
  syslog(LOG_INFO, "Total frame length %d bytes, data rate %.1f Mbps",
         (int)frame->frame_header_size + (int)frame->
      frame_body_size,
         (float)info->datarate / 2.0);
  syslog(LOG_INFO, "Frame was received in MAC HW at: %llu us",
         info->rx_time_us);
  syslog(LOG_INFO, "RX complete ISR occured at: %llu us",
         info->rx_isr_time_us);
}

/* Callback called on each sent frame */
static void
tx_callback(const wlan_frame_t *frame, const wlan_tx_frame_info_t *info)
{
  syslog(LOG_INFO, "Sent frame on device ID %u", info->device_id);
  syslog(LOG_INFO, "Total frame length %d bytes, data rate %.1f Mbps",
         (int)frame->frame_header_size + (int)frame->
      frame_body_size,
         (float)info->datarate / 2.0);
  syslog(LOG_INFO, "Frame was queued in TX queue at: %llu us",
         info->tx_queue_time_us);
  syslog(LOG_INFO, "TX complete ISR occured at: %llu us",
         info->tx_isr_time_us);
}

#else /* __CRATON_NO_ARC || __CRATON_ARC1 */

void craton_user_init(void)
{
}

#endif  /* __CRATON_NO_ARC || __CRATON_ARC1 */
```

8.51 remote-posix/crypto/aes-example.c

```
/* Copyright (C) 2014-2015 Autotalks Ltd. */
#include <stdio.h>
#include <string.h>
#include <inttypes.h>

#include <atlk/sdk.h>
#include <atlk/aes.h>

/*
  CRATON Remote AES Example

  This example demonstrates the usage of the AES-CBC encryption/decryption and
  AES-CMAC generation APIs for code running on top of CRATON processor
  with ThreadX RTOS.
*/

/* Helper function for converting buffer to hex */
static void
buffer_to_line(const void *buf, size_t buf_len, char *line)
{
  const uint8_t *ptr = buf;
  char *pos = &line[0];
  size_t i;

  if (!buf_len) {
    *pos = '\0';
    return;
  }

  if (buf_len > 16) {
    buf_len = 16;
  }

  for (i = 0; i < buf_len - 1; i++) {
    pos += sprintf(pos, "%02x ", ptr[i]);
  }
  pos += sprintf(pos, "%02x", ptr[i]);
}

/* Print buffer to standard output */
static void
buffer_print(const void *buf, size_t len)
{
  const uint8_t *ptr = buf;
  size_t i, line_len, remaining = len;
  char line[80];

  for (i = 0; i < len; i += 16) {
    line_len = remaining < 16 ? remaining : 16;
    remaining -= 16;

    buffer_to_line(ptr + i, line_len, line);
    printf("  %.8lx: %s\n", (unsigned long)i, line);
  }
}

/*
 * AES-CBC example test vectors were taken from:
 *   NIST Special Publication 800-38A:
 *   Recommendation for Block Cipher Modes of Operation:
 *   Methods and Techniques,
 *   Appendix F.2
 */

/* Example AES key used for AES-CBC encryption/decryption */
static const aes_key_t aes_cbc_key = {
  { 0x2b, 0x7e, 0x15, 0x16, 0x28, 0xae, 0xd2, 0xa6,
    0xab, 0xf7, 0x15, 0x88, 0x09, 0xcf, 0x4f, 0x3c }
};

/* Example initialization vector used for AES-CBC encryption/decryption */
static const aes_cbc_iv_t aes_cbc_iv = {
  { 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
    0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f }
};

/* Example plaintext used for AES-CBC encryption */
static const uint8_t aes_cbc_plaintext[] = {
  0x6b, 0xc1, 0xbe, 0xe2, 0x2e, 0x40, 0x9f, 0x96,
  0xe9, 0x3d, 0x7e, 0x11, 0x73, 0x93, 0x17, 0x2a,
  0xae, 0x2d, 0x8a, 0x57, 0x1e, 0x03, 0xac, 0x9c,
  0x9e, 0xb7, 0x6f, 0xac, 0x45, 0xaf, 0x8e, 0x51,
  0x30, 0xc8, 0x1c, 0x46, 0xa3, 0x5c, 0xe4, 0x11,
  0xe5, 0xfb, 0xc1, 0x19, 0x1a, 0x0a, 0x52, 0xef,
  0xf6, 0x9f, 0x24, 0x45, 0xdf, 0x4f, 0x9b, 0x17,
  0xad, 0x2b, 0x41, 0x7b, 0xe6, 0x6c, 0x37, 0x10
};

/* Example ciphertext used for AES-CBC decryption */
static const uint8_t aes_cbc_ciphertext[] = {
  0x76, 0x49, 0xab, 0xac, 0x81, 0x19, 0xb2, 0x46,
  0xce, 0xe9, 0x8e, 0x9b, 0x12, 0xe9, 0x19, 0x7d,
  0x50, 0x86, 0xcb, 0x9b, 0x50, 0x72, 0x19, 0xee,
  0x95, 0xdb, 0x11, 0x3a, 0x91, 0x76, 0x78, 0xb2,
  0x73, 0xbe, 0xd6, 0xb8, 0xe3, 0xc1, 0x74, 0x3b,
  0x71, 0x16, 0xe6, 0x9e, 0x22, 0x22, 0x95, 0x16,
  0x3f, 0xf1, 0xca, 0xa1, 0x68, 0x1f, 0xac, 0x09,
  0x12, 0x0e, 0xca, 0x30, 0x75, 0x86, 0xe1, 0xa7
};

static void
aes_cbc_encrypt_example(void)
{
  /* Buffer for storing ciphertext */
  uint8_t ciphertext[sizeof(aes_cbc_plaintext)];
  /* Size of ciphertext */
  size_t ciphertext_size = sizeof(ciphertext);
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;

  printf("AES-CBC encryption example\n");

  printf("Encryption key:\n");
  buffer_print(&aes_cbc_key, sizeof(aes_cbc_key));

  printf("Initialization vector:\n");
  buffer_print(&aes_cbc_iv, sizeof(aes_cbc_iv));

  printf("Plaintext:\n");
  buffer_print(aes_cbc_plaintext, sizeof(aes_cbc_plaintext));

  /* Encrypt plaintext with AES-CBC */
  rc = aes_cbc_encrypt(&aes_cbc_key,
                         &aes_cbc_iv,
                         aes_cbc_plaintext, sizeof(aes_cbc_plaintext),
                         ciphertext, &ciphertext_size);
  if (atlk_error(rc)) {
    fprintf(stderr, "aes_cbc_encrypt: %s\n", atlk_rc_to_str(rc));
    return;
  }

  printf("Ciphertext:\n");
  buffer_print(ciphertext, ciphertext_size);

  /* Make sure ciphertext is correct */
  if (memcmp(aes_cbc_ciphertext, ciphertext, ciphertext_size) == 0) {
    printf("AES-CBC encryption succeeded\n");
  }
  else {
    printf("AES-CBC encryption failed\n");
  }
  printf("\n");
}

static void
aes_cbc_decrypt_example(void)
{
  /* Buffer for storing plaintext */
  uint8_t plaintext[sizeof(aes_cbc_ciphertext)];
  /* Size of plaintext */
  size_t plaintext_size = sizeof(plaintext);
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;

  printf("AES-CBC decryption example\n");

  printf("Decryption key:\n");
  buffer_print(&aes_cbc_key, sizeof(aes_cbc_key));

  printf("Initialization vector:\n");
  buffer_print(&aes_cbc_iv, sizeof(aes_cbc_iv));

  printf("Ciphertext:\n");
  buffer_print(aes_cbc_ciphertext, sizeof(aes_cbc_ciphertext));

  /* Decrypt ciphertext with AES-CBC */
  rc = aes_cbc_decrypt(&aes_cbc_key,
                       &aes_cbc_iv,
                       aes_cbc_ciphertext, sizeof(aes_cbc_ciphertext),
                       plaintext, &plaintext_size);
  if (atlk_error(rc)) {
    fprintf(stderr, "aes_cbc_decrypt: %s\n", atlk_rc_to_str(rc));
    return;
  }

  printf("Plaintext:\n");
  buffer_print(plaintext, plaintext_size);

  /* Make sure plaintext is correct */
  if (memcmp(aes_cbc_plaintext, plaintext, plaintext_size) == 0) {
    printf("AES-CBC decryption succeeded\n");
  }
  else {
    printf("AES-CBC decryption failed\n");
  }
  printf("\n");
}

/*
 * AES-CMAC example test vectors were taken from:
 *   NIST Special Publication 800-38B:
 *   Recommendation for Block Cipher Modes of Operation:
 *   The CMAC Mode for Authentication,
 *   Appendix D.1
 */

/* Example AES key used for AES-CMAC tag generation */
static const aes_key_t aes_cmac_key = {
  { 0x2b, 0x7e, 0x15 ,0x16, 0x28, 0xae, 0xd2, 0xa6,
    0xab, 0xf7, 0x15, 0x88, 0x09, 0xcf, 0x4f, 0x3c }
};

/* Example message for AES-CMAC tag generation */
static const uint8_t aes_cmac_msg[] = {
  0x6b, 0xc1, 0xbe, 0xe2, 0x2e, 0x40, 0x9f, 0x96,
  0xe9, 0x3d, 0x7e, 0x11, 0x73, 0x93, 0x17, 0x2a
};

/* Expected AES-CMAC tag */
static const aes_cmac_tag_t aes_cmac_tag = {
  { 0x07, 0x0a, 0x16, 0xb4, 0x6b, 0x4d, 0x41, 0x44,
    0xf7, 0x9b, 0xdd, 0x9d, 0xd0, 0x4a, 0x28, 0x7c }
};

static void
aes_cmac_example(void)
{
  /* AES-CMAC tag */
  aes_cmac_tag_t tag = AES_CMAC_TAG_INIT;
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;

  printf("AES-CMAC example:\n");

  printf("Key:\n");
  buffer_print(&aes_cmac_key, sizeof(aes_cmac_key));

  printf("Message:\n");
  buffer_print(aes_cmac_msg, sizeof(aes_cmac_msg));

  /* Compute AES-CMAC tag */
  rc = aes_cmac_compute(&aes_cmac_key, aes_cmac_msg, sizeof(aes_cmac_msg), &tag);
  if (atlk_error(rc)) {
    fprintf(stderr, "aes_cmac_compute: %s\n", atlk_rc_to_str(rc));
    return;
  }

  printf("AES-CMAC Tag:\n");
  buffer_print(&tag, sizeof(tag));

  /* Make sure tag is correct */
  if (memcmp(&aes_cmac_tag, &tag, sizeof(tag)) == 0) {
    printf("AES-CMAC generation succeeded\n");
  }
  else {
    printf("AES-CBC generation failed\n");
  }
  printf("\n");
}

int main(int argc, char *argv[])
{
  (void)argc;
  (void)argv;

  /* AES-CBC encryption example */
  aes_cbc_encrypt_example();

  /* AES-CBC decryption example */
  aes_cbc_decrypt_example();

  /* AES-CMAC example */
  aes_cmac_example();

  return 0;
}
```

8.52 remote-posix/crypto/ecdsa-benchmark.c

```
/* Copyright (C) 2013-2016 Autotalks Ltd. */
#include <stdio.h>
#include <assert.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
#include <string.h>
#include <pthread.h>
#include <arpa/inet.h>

#include <atlk/sdk.h>
#include <atlk/remote.h>
#include <atlk/ecc_remote.h>
#include <atlk/ecc_service.h>

/*
  CRATON Remote ECDSA Benchmark

  This program benchmarks CRATON ECDSA verification API with some of the
  elliptic curves supported by the API. The same verification request is
  used over and over for each curve. All verification are intended to result
  in success. This is checked using assert().

  To take advantage of CRATON HW parallelism the average number of
  ongoing requests (i.e. started but not completed) should be more than 1.
  Having more than 3 ongoing requests will not result in any throughput gains
  but is supported up to an implementation-defined upper bound. For reference,
  in SDK 4.5 this upper bound is 256.
*/

/* Compile-time number of elements in array */
#define ARRAY_SIZE(arr) (sizeof(arr) / sizeof((arr)[0]))

/* Total number of requests per benchmark */
#define NUM_REQUESTS 10000

/* Max round trip time in usec */
#define EXAMPLE_MAX_RTT_USEC 100000

/*
  Number of incomplete requests at any point in time.
  CRATON ECDSA HW will provide less than maximum throughput
  if this number is less than 3.
*/
#define NUM_ONGOING 8

/* Is valid response? */
#define VALID_RESPONSE(last_received, last_sent) (last_received <= last_sent)

/* Benchmark descriptor */
struct benchmark {
  const char *curve_name;
  ecc_request_t request;
};

/* Remote transport object */
remote_transport_t *transport = NULL;

/* Array of benchmark descriptors (one per curve) */
static const struct benchmark benchmarks[] = {
  {
    .curve_name = "NIST P256",
    .request =  {
      .context = {
        .request_id = 0,
        .request_type = ECC_REQUEST_TYPE_VERIFY,
        .curve = ECC_CURVE_NIST_P256
      },
      .params.verify_params = {
        .public_key = {
                .point_type = ECC_POINT_UNCOMPRESSED,
                .x_coordinate = {
                  .value = {
                    0xbc3fdd5d, 0x620d0a14, 0x5d867d8b, 0x286867ec,
                    0x92c47d90, 0x8a772d43, 0x44eb3895, 0x26f3751e
                  }
                },
                .y_coordinate = {
                  .value = {
                    0x96fc56f1, 0xf79baeaa, 0xff5b3542, 0xb7ffb678,
                    0xc22d9ddb, 0x3dc0cb4d, 0xf0e24af5, 0x1606db3b
                  }
              }
        },
        .digest = {
                .value = {
                  0xb9, 0x3d, 0x12, 0xb2, 0xc6, 0x02, 0x7b, 0x0b,
                  0xa4, 0xd4, 0xd8, 0xc2, 0xbc, 0x20, 0xda, 0x88,
                  0x8b, 0xe2, 0x42, 0x2f, 0x08, 0x9b, 0xe3, 0x24,
                  0x3a, 0x6c, 0x44, 0xe5, 0x0d, 0xde, 0xf0, 0xcb
                },
                .value_size = 32
        },
        .signature = {
                .r_scalar = {
                  .value = {
                    0x4e3a775c, 0x71a5c259, 0xfad57a8e, 0xd1e45591,
                    0x030fbb65, 0x94d2300b, 0x7ceccd7d, 0xbc70ad36
                  }
                },
                .s_scalar = {
                  .value = {
                    0xbc05d39c, 0xd2c5f32b, 0xf10502c6, 0xb91de10c,
                    0x8599d089, 0x0873e8ae, 0x7b137225, 0xd51dd454
                  }
                }
        }
      }
    }
  },

  {
    .curve_name = "Brainpool P256t1",
    .request = {
      .context = {
        .request_id = 0,
        .request_type = ECC_REQUEST_TYPE_VERIFY,
        .curve = ECC_CURVE_BRAINPOOL_P256t1
      },
      .params.verify_params = {
        .public_key = {
                .point_type = ECC_POINT_UNCOMPRESSED,
                .x_coordinate = {
                  .value = {
                    0xfc7fc794, 0x3a85ed28, 0xc3ebcaaf, 0x2a326938,
                    0xef9ed9c7, 0x779df5c2, 0x6e220a95, 0x6344dff5
                  }
                },
                .y_coordinate = {
                  .value = {
                    0x365babff, 0xf6cfc69f, 0xc2a9f394, 0x94cd22bf,
                    0x46cbf110, 0x273452ae, 0xf55a41f3, 0x2e2e94a8
                  }
                }
        },
        .digest = {
                .value = {
                  0x8e, 0x89, 0x03, 0x45, 0x87, 0x5b, 0xef, 0x0b,
                  0xaa, 0xa0, 0xe0, 0x98, 0xbf, 0xf2, 0x78, 0xdd,
                  0xbf, 0x00, 0xee, 0x06, 0xcc, 0x08, 0x07, 0xa9,
                  0xd8, 0xf6, 0x4c, 0x93, 0x29, 0xb0, 0xd2, 0x2d
                },
                .value_size = 32
        },
        .signature = {
                .r_scalar = {
                  .value = {
                    0x15a73647, 0xb0ed3efa, 0x6f44c325, 0x7607b1a5,
                    0xa06cf2a1, 0xc5f298a9, 0x13c2c3bc, 0x9168331f
                  }
                },
                .s_scalar = {
                  .value = {
                    0xec7d28a2, 0x396dbb17, 0xbfc33ae6, 0xf0832dd6,
                    0x2adf90bb, 0x4b422130, 0x46ad044f, 0x353f89ca
                  }
                }
        }
      }
    }
  }
};

static ecc_service_t *ecc_benchmark_service = NULL;
static ecc_socket_t *ecc_benchmark_socket = NULL;

static void
run_benchmark(const struct benchmark *benchmark);

void ecdsa_benchmark_start(unsigned int input);
void ecdsa_latency_benchmark_start(unsigned int input);

int main(int argc, char *argv[])
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* Remote IP transport configuration parameters */
  remote_ip_transport_config_t config =
      REMOTE_IP_TRANSPORT_CONFIG_INIT;
  /* Local IPv4 address */
  uint32_t local_ipv4_addr;
  /* Server IPv4 address */
  uint32_t server_ipv4_addr;

  if ((argc != 3) || ((server_ipv4_addr = inet_addr(argv[1])) == INADDR_NONE)) {
    fprintf(stderr, "Usage: %s SERVER-IP4-ADDR LOCAL-IF-NAME\n", argv[0]);
    return EXIT_FAILURE;
  }

  /* Get local IPv4 address */
  rc = remote_util_local_ipv4_address_get(argv[2], &local_ipv4_addr);
  if (atlk_error(rc)) {
    fprintf(stderr, "remote_util_local_ipv4_address_get: %s\n",
            atlk_rc_to_str(rc));
    goto exit;
  }

  /* Set remote IP transport configuration parameters */
  config.local_ipv4_address = local_ipv4_addr;
  config.remote_ipv4_address = server_ipv4_addr;
  config.max_rtt_ms = 20;

  /* Create remote IP transport object */
  rc = remote_ip_transport_create(&config, &transport);
  if (atlk_error(rc)) {
    fprintf(stderr, "remote_ip_transport_create: %s\n", atlk_rc_to_str(rc));
    goto exit;
  }

  /* Create the ECC service */
  rc = ecc_remote_service_create(transport, NULL, &ecc_benchmark_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "ecc_remote_service_create: %s\n", atlk_rc_to_str(rc));
    goto exit;
  }

  /* Create socket */
  rc = ecc_socket_create(ecc_benchmark_service, &ecc_benchmark_socket);
  assert(!atlk_error(rc));

  /* start throughput test */
  ecdsa_benchmark_start(0);

  /* Start latency test */
  ecdsa_latency_benchmark_start(0);

exit:
  /* Clean-up resources */
  ecc_socket_delete(ecc_benchmark_socket);
  ecc_service_delete(ecc_benchmark_service);
  remote_transport_delete(transport);

  return atlk_error(rc);
}

static void
timestamp_get(struct timespec *ts)
{
  int rv;

  rv = clock_gettime(CLOCK_MONOTONIC, ts);
  if (rv != 0) {
    printf("Error getting the current time: rv = %d", rv);
  }
}

static void
timestamp_delta_calc(const struct timespec *new,
                     const struct timespec *old,
                     int32_t *delta_ms)
{
  if ((new->tv_nsec - old->tv_nsec) < 0) {
    *delta_ms = (new->tv_sec - old->tv_sec - 1) * 1000;;
    *delta_ms += (1000000000 + new->tv_nsec - old->tv_nsec) / 1000000;
  }
  else {
     *delta_ms = (new->tv_sec - old->tv_sec) * 1000;
     *delta_ms += (new->tv_nsec - old->tv_nsec) / 1000000;
  }
}

static void
send_fake_request(const struct benchmark *benchmark)
{
  atlk_rc_t rc;
  ecc_request_t request;
  ecc_response_t response;
  atlk_wait_t wait = ATLK_WAIT_INIT;

  /* Send 1 fake request in order to update the Host ARP table */
  /* Setup a long wait */
  wait.wait_type = ATLK_WAIT_INTERVAL;
  wait.wait_usec = 1000000;
  request = benchmark->request;
  /* Set a high request_id for the fake packet */
  request.context.request_id = NUM_REQUESTS - 1;
  rc = ecc_request_send(ecc_benchmark_socket, &request, NULL);
  assert(!atlk_error(rc));
  /* Wait for the response from the target, for 1 second */
  usleep(100000);
  /* Read the response */
  rc = ecc_response_receive(ecc_benchmark_socket, &response, &wait);
  if (rc != ATLK_E_TIMEOUT) {
    assert(!atlk_error(rc));
  }
}
```

```
/* Buffer for sent timestamps */
static struct timespec packet_sent_timestamp[NUM_REQUESTS];

/* Buffer for storing all round trip times for each request */
static int32_t packet_rtt_ms[NUM_REQUESTS];

static int request_id_received[NUM_REQUESTS];

static void
run_latency_benchmark(const struct benchmark *benchmark)
{
  atlk_rc_t rc;
  ecc_response_t response = ECC_RESPONSE_INIT;
  struct timespec ts_end;
  int32_t num_received;
  int32_t num_timeouts;
  int32_t num_sent;
  int32_t average_packet_rtt_ms;
  int32_t max_packet_rtt_ms;
  ecc_request_t request = ECC_REQUEST_INIT;
  struct timespec ts_now;
  int i;
  atlk_wait_t wait = ATLK_WAIT_INIT;

  memset(request_id_received, 0x00, sizeof(request_id_received));
  memset(packet_rtt_ms, 0x00, sizeof(packet_rtt_ms));
  memset(packet_sent_timestamp, 0x00, sizeof(packet_sent_timestamp));

  /* Start benchmark */
  printf("Benchmarking latency of ECDSA verification with curve \"%s\"...\n",
         benchmark->curve_name);

  /* Setup wait, not forever */
  wait.wait_type = ATLK_WAIT_INTERVAL;
  wait.wait_usec = EXAMPLE_MAX_RTT_USEC;

  /* Start a few requests to take advantage of HW parallelism */
  num_sent = 0;
  request = benchmark->request;
  request.context.request_id = 0;
  for (i = 0; i < NUM_ONGOING; i++) {
    timestamp_get(&packet_sent_timestamp[request.context.request_id]);
    rc = ecc_request_send(ecc_benchmark_socket, &request, NULL);
    assert(!atlk_error(rc));
    num_sent++;
    request.context.request_id++;
  }

  /* Start a new request whenever an ongoing request completes */
  num_received = 0;
  num_timeouts = 0;
  for (i = 0; i < NUM_REQUESTS - NUM_ONGOING; i++) {
    rc = ecc_response_receive(ecc_benchmark_socket, &response, &wait);
    if (rc != ATLK_E_TIMEOUT) {
      assert(!atlk_error(rc));
      assert(response.rc == ECC_OK);
      num_received++;

      if (VALID_RESPONSE(response.context.request_id, request.
      context.request_id)) {
        request_id_received[response.context.request_id] = 1;
      }
    }
    else {
      num_timeouts++;
    }

    if (VALID_RESPONSE(response.context.request_id, request.
      context.request_id)) {
      timestamp_get(&ts_now);
      timestamp_delta_calc(&ts_now,
                           &packet_sent_timestamp[response.context.
      request_id],
                           &packet_rtt_ms[response.context.request_id]);
    }

    timestamp_get(&packet_sent_timestamp[request.context.request_id]);
    rc = ecc_request_send(ecc_benchmark_socket, &request, NULL);
    assert(!atlk_error(rc));

    request.context.request_id++;
    num_sent++;
  }

  /* Wait for all ongoing requests to complete */
  while ((num_received + num_timeouts) < NUM_REQUESTS) {
    rc = ecc_response_receive(ecc_benchmark_socket, &response, &wait);
    if (rc != ATLK_E_TIMEOUT) {
      assert(!atlk_error(rc));
      assert(response.rc == ECC_OK);
      num_received++;

      if (VALID_RESPONSE(response.context.request_id, request.
      context.request_id)) {
        request_id_received[response.context.request_id] = 1;
      }
    }
    else {
      num_timeouts++;
    }

    timestamp_get(&ts_now);
    if (VALID_RESPONSE(response.context.request_id, request.
      context.request_id)) {
      timestamp_delta_calc(&ts_now,
                           &packet_sent_timestamp[response.context.
      request_id],
                           &packet_rtt_ms[response.context.request_id]);
    }
  }

  timestamp_get(&ts_end);

  /* Calculate max and average delta in msec */
  max_packet_rtt_ms = 0;
  average_packet_rtt_ms = 0;
  for (i = 0; i < NUM_REQUESTS; i++) {
    if (request_id_received[i]) {
      if (max_packet_rtt_ms < packet_rtt_ms[i]) {
        max_packet_rtt_ms = packet_rtt_ms[i];
      }

      average_packet_rtt_ms += packet_rtt_ms[i];
    }
  }

  average_packet_rtt_ms /= num_received;

  printf("Latency of ECDSA verification with curve \"%s\": "
         "max = %d ms, avg = %d ms\n",
         benchmark->curve_name,
         (int)max_packet_rtt_ms,
         (int)average_packet_rtt_ms);
}

static void
run_benchmark(const struct benchmark *benchmark)
{
  atlk_rc_t rc;
  ecc_response_t response = ECC_RESPONSE_INIT;
  struct timespec ts_start;
  struct timespec ts_end;
  int32_t elapsed_time_ms;
  size_t num_received;
  size_t num_timeouts;
  size_t num_sent;
  ecc_request_t request = ECC_REQUEST_INIT;
  atlk_wait_t wait = ATLK_WAIT_INIT;

  memset(request_id_received, 0x00, sizeof(request_id_received));

  /* Send 1 fake request in order to update the Host ARP table */
  send_fake_request(benchmark);

  /* Setup wait, not forever */
  wait.wait_type = ATLK_WAIT_INTERVAL;
  wait.wait_usec = EXAMPLE_MAX_RTT_USEC;

  /* Start benchmark */
  printf("Benchmarking ECDSA verification with curve \"%s\"...\n",
         benchmark->curve_name);

  timestamp_get(&ts_start);

  /* Start a few requests to take advantage of HW parallelism */
  num_sent = 0;
  num_timeouts = 0;
  request = benchmark->request;
  request.context.request_id = 0;

  for (int i = 0; i < NUM_ONGOING; i++) {
    rc = ecc_request_send(ecc_benchmark_socket, &request, NULL);
    assert(!atlk_error(rc));
    num_sent++;
    request.context.request_id++;
  }

  /* Start a new request whenever an ongoing request completes */
  num_received = 0;
  for (int i = 0; i < NUM_REQUESTS - NUM_ONGOING; i++) {
    rc = ecc_response_receive(ecc_benchmark_socket, &response, &wait);
    if (rc != ATLK_E_TIMEOUT) {
      assert(!atlk_error(rc));
      assert(response.rc == ECC_OK);
      num_received++;

      request_id_received[response.context.request_id] = 1;
    }
    else {
      num_timeouts++;
    }

    rc = ecc_request_send(ecc_benchmark_socket, &request, NULL);
    assert(!atlk_error(rc));

    request.context.request_id++;
    num_sent++;
  }

  /* Wait for all ongoing requests to complete */
  while ((num_received + num_timeouts) < NUM_REQUESTS) {
    rc = ecc_response_receive(ecc_benchmark_socket, &response, &wait);
    if (rc != ATLK_E_TIMEOUT) {
      assert(!atlk_error(rc));
      assert(response.rc == ECC_OK);
      num_received++;

      request_id_received[response.context.request_id] = 1;
    }
    else {
      num_timeouts++;
    }
  }

  timestamp_get(&ts_end);

  /* Calculate benchmark */
  elapsed_time_ms = (ts_end.tv_sec - ts_start.tv_sec) * 1000;
  elapsed_time_ms += ((ts_end.tv_nsec - ts_start.tv_nsec) / 1000000);

  printf("ECDSA verification throughput with curve \"%s\" is %.1f Hz\n",
         benchmark->curve_name,
         (float)num_received / ((float)elapsed_time_ms / 1000));
}

void ecdsa_benchmark_start(unsigned int input)
{
  (void)input;

  printf("*** Start of ECDSA benchmark suite ***\n");

  for (size_t i = 0; i < ARRAY_SIZE(benchmarks); i++) {
    run_benchmark(&benchmarks[i]);
  }

  printf("*** End of ECDSA benchmark suite ***\n");
}

void ecdsa_latency_benchmark_start(unsigned int input)
{
  (void)input;

  printf("*** Start of ECDSA Latency benchmark suite ***\n");

  for (size_t i = 0; i < ARRAY_SIZE(benchmarks); i++) {
    run_latency_benchmark(&benchmarks[i]);
  }

  printf("*** End of ECDSA benchmark suite ***\n");
}
```

8.53 remote-posix/crypto/ecdsa-example.c

```
/* Copyright (C) 2014-2015 Autotalks Ltd. */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>
```

```
#include <pthread.h>
#include <arpa/inet.h>

#include <atlk/sdk.h>
#include <atlk/sha.h>
#include <atlk/ecc.h>
#include <atlk/ecdsa.h>
#include <atlk/ecc_service.h>
#include <atlk/hsm_service.h>
#include <atlk/hsm_emulator.h>
#include <atlk/sha_sw.h>

#include <atlk/remote.h>
#include <atlk/ecc_remote.h>

/*
  CRATON Remote ECDSA Example

  This example demonstrates a basic ECDSA signing/verification scenario using
  the HSM API, ECC API and CRATON SHA API for code running on top of CRATON
  processor with ThreadX RTOS.

  The device used in this example is a "HSM emulator", a term used to describe
  an emulated HSM device. The differences between an emulated HSM service
  and a real one are:
  - The emulated HSM service is created via hsm_emulator_create().
  - The implementation is not tamper-resistant because it uses general purpose
    hardware instead of tamper-resistant hardware.

  The purpose of the emulated HSM is basic software integration with
  the HSM API on a hardware platform that doesn't have a working HSM chip.
*/

/* HSM emulator filename */
#define HSM_EMULATOR_FILENAME "./hsm-emu.dat"

/* Number of storage cells for HSM */
#define EXAMPLE_NUM_OF_STORAGE_CELLS 128

/* HSM emulator service */
static hsm_service_t *example_hsm_service = NULL;

/* ECC service */
static ecc_service_t *example_ecc_service = NULL;

/* ECC socket */
static ecc_socket_t *example_ecc_socket = NULL;

/* Number of NVM cells to configure for HSM */
#define ECDSA_EXAMPLE_HSM_NVM_NUM_CELLS 128

/* ECDSA example message maximum data size in octets */
#define ECDSA_EXAMPLE_MSG_MAX_DATA_SIZE 64

/* ECDSA example message */
typedef struct {
  /* Data (octet string) */
  uint8_t data[ECDSA_EXAMPLE_MSG_MAX_DATA_SIZE];

  /* Data size in octets */
  size_t data_size;

  /* ECC elliptic curve */
  ecc_curve_t curve;

  /* ECC public key */
  ecc_point_t public_key;

  /* ECDSA fast verification signature */
  ecc_fast_verification_signature_t signature;

} ecdsa_example_message_t;

/* Format string for ECC scalar */
#define ECC_SCALAR_FMT \
  "0x%04x,0x%04x,0x%04x,0x%04x,0x%04x,0x%04x,0x%04x,0x%04x," \
  "0x%04x,0x%04x,0x%04x,0x%04x"

/* Format argument list for ecc_scalar_t */
#define ECC_SCALAR_FMT_ARGS(x)                    \
  x.value[0], x.value[1], x.value[2], x.value[3], \
  x.value[4], x.value[5], x.value[6], x.value[7], \
  x.value[8], x.value[9], x.value[10], x.value[11]

/* Format string for SHA digest */
#define SHA_256_DIGEST_FMT \
  "%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x" \
  "%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x"

/* Format argument list for SHA digest */
#define SHA_256_DIGEST_FMT_ARGS(x)                        \
  x.value[0], x.value[1], x.value[2], x.value[3],        \
  x.value[4], x.value[5], x.value[6], x.value[7],        \
  x.value[8], x.value[9], x.value[10], x.value[11],      \
  x.value[12], x.value[13], x.value[14], x.value[15],    \
  x.value[16], x.value[17], x.value[18], x.value[19],    \
  x.value[20], x.value[21], x.value[22], x.value[23],    \
  x.value[24], x.value[25], x.value[26], x.value[27],    \
  x.value[28], x.value[29], x.value[30], x.value[31]

static atlk_rc_t
ecdsa_example_alice(ecdsa_example_message_t *msg)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* HSM secure storage cell index */
  hsm_cell_index_t cell_index;
  /* Private key information */
  hsm_ecc_private_key_info_t private_key_info =
      HSM_ECC_PRIVATE_KEY_INFO_INIT;
  /* ECC point representing a public key */
  ecc_point_t public_key = ECC_POINT_INIT;
  /* SHA digest */
  sha_digest_t digest = SHA_DIGEST_INIT;
  /* ECDSA fast verification signature */
  ecc_fast_verification_signature_t signature =
    ECC_FAST_VERIFICATION_SIGNATURE_INIT;
  /* Example message */
  static const char example_msg[] =
    "Autotalks - The Confidence of Knowing Ahead";

  printf("\n>>> Alice\n");

  /* Print the message data */
  printf("Message data: %s\n", example_msg);
  printf("Message data size: %zu\n", sizeof(example_msg));

  /* Arbitrarily chosen HSM cell index for the sake of this example */
  cell_index = 6;
  printf("Using HSM cell index: %u\n", cell_index);

  /* Using NIST P-256 elliptic curve and an Isolated key */
  private_key_info.key_curve = ECC_CURVE_NIST_P256;
  private_key_info.key_type = HSM_PRIVATE_KEY_TYPE_ISOLATED;

  printf("Using elliptic curve ID: %u\n", private_key_info.key_curve);
  printf("Using key type ID: %u\n", private_key_info.key_type);

  /* Create private key and store it in the chosen cell */
  rc = hsm_ecc_private_key_create(example_hsm_service,
                                  cell_index,
                                  &private_key_info);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_ecc_private_key_create: %s\n", atlk_rc_to_str(rc));
    return rc;
  }

  printf("ECC private key created\n");

  /* Retrieve public key for this cell's private key */
  rc = hsm_ecc_public_key_get(example_hsm_service, cell_index, &public_key);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_ecc_public_key_get: %s\n", atlk_rc_to_str(rc));
    return rc;
  }

  assert(public_key.point_type == ECC_POINT_UNCOMPRESSED);

  /* Print retrieved ECC public key */
  printf("ECC public key created:\n");
  printf("  x: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(public_key.x_coordinate));
  printf("  y: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(public_key.y_coordinate));

  /* Compute SHA-256 digest of example message */
  rc = sha_sw_sha256_compute(example_msg, sizeof(example_msg), &digest);
  if (atlk_error(rc)) {
    fprintf(stderr, "sha_sw_sha256_compute: %s", atlk_rc_to_str(rc));
    return rc;
  }

  /* Print computed SHA-256 digest */
  printf("SHA-256 hash digest computed:\n");
  printf("  Digest: " SHA_256_DIGEST_FMT "\n", SHA_256_DIGEST_FMT_ARGS(digest));

  /* Generate ECDSA fast verification signature */
  rc = hsm_ecdsa_sign(example_hsm_service, cell_index, &digest, &signature);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_ecdsa_sign: %s\n", atlk_rc_to_str(rc));
    return rc;
  }

  assert(signature.R_point.point_type == ECC_POINT_UNCOMPRESSED);

  /* Print generated ECDSA signature */
  printf("ECDSA signature generated:\n");
  printf("  Rx: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(signature.R_point.x_coordinate));
  printf("  Ry: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(signature.R_point.y_coordinate));
  printf("  s: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(signature.s_scalar));

  /* Make sure the example message can fit into the data */
  assert(sizeof(example_msg) <= sizeof(msg->data));

  /* Produce the message */
  msg->data_size = sizeof(example_msg);
  memcpy(msg->data, example_msg, msg->data_size);
  msg->curve = private_key_info.key_curve;
  msg->public_key = public_key;
  msg->signature = signature;

  return ATLK_OK;
}

static atlk_rc_t
ecdsa_example_bob(const ecdsa_example_message_t *msg)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* SHA digest */
  sha_digest_t digest = SHA_DIGEST_INIT;
  /* ECDSA signature */
  ecc_signature_t signature = ECC_SIGNATURE_INIT;
  /* ECC request */
  ecc_request_t request = ECC_REQUEST_INIT;
  /* ECC response */
  ecc_response_t response = ECC_RESPONSE_INIT;
  /* ECC request identifier */
  ecc_request_id_t request_id;

  printf("\n>>> Bob\n");

  /* Print received message */
  printf("Message data: %s\n", msg->data);
  printf("Message data size: %zu\n", msg->data_size);
  printf("Using elliptic curve ID: %u\n", msg->curve);

  assert(msg->public_key.point_type == ECC_POINT_UNCOMPRESSED);

  /* Print received ECC public key */
  printf("ECC public key:\n");
  printf("  x: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(msg->public_key.x_coordinate));
  printf("  y: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(msg->public_key.y_coordinate));

  /* Print received ECDSA signature for fast verification */
  printf("ECDSA signature:\n");
  printf("  Rx: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(msg->signature.R_point.x_coordinate));
  printf("  Ry: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(msg->signature.R_point.y_coordinate));
  printf("  s: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(msg->signature.s_scalar));

  /* Compute SHA-256 hash value of received message */
  rc = sha_sw_sha256_compute(msg->data, msg->data_size, &digest);
  if (atlk_error(rc)) {
    fprintf(stderr, "sha_sw_sha256_compute: %s", atlk_rc_to_str(rc));
    return rc;
  }

  /* Print computed SHA-256 digest */
  printf("SHA-256 hash digest computed:\n");
  printf("  Digest: " SHA_256_DIGEST_FMT "\n", SHA_256_DIGEST_FMT_ARGS(digest));

  /* Convert ECDSA signature for fast verification */
  rc = ecdsa_signature_convert(msg->curve, &msg->signature, &signature);
  if (atlk_error(rc)) {
    fprintf(stderr, "ecdsa_signature_convert: %s", atlk_rc_to_str(rc));
    return rc;
  }

  /* Print converted ECDSA signature for fast verification */
  printf("Converted ECDSA signature for fast verification:\n");
  printf("  r: " ECC_SCALAR_FMT "\n", ECC_SCALAR_FMT_ARGS(signature.r_scalar));
  printf("  s: " ECC_SCALAR_FMT "\n", ECC_SCALAR_FMT_ARGS(signature.s_scalar));

  /* Arbitrary request identifier */
  request_id = 10;

  /* Fill ECC verification request */
  request.context.request_id = request_id;
  request.context.request_type = ECC_REQUEST_TYPE_VERIFY;
  request.context.curve = msg->curve;
  request.params.verify_params.public_key = msg->public_key;
  request.params.verify_params.digest = digest;
  request.params.verify_params.signature = signature;

  /* Send ECC request */
  rc = ecc_request_send(example_ecc_socket, &request, NULL);
  if (atlk_error(rc)) {
    fprintf(stderr,"ecc_request_send: %s\n", atlk_rc_to_str(rc));
    return rc;
  }

  /* Print ECC request ID */
  printf("Sent ECC request with ID %" PRIu32 "\n", request_id);

  /* Receive ECC response */
  rc = ecc_response_receive(example_ecc_socket, &response, &
      atlk_wait_forever);
  if (atlk_error(rc)) {
    fprintf(stderr,"ecc_response_receive: %s\n", atlk_rc_to_str(rc));
    return rc;
  }

  /* Print ECC response */
  printf("ECC response for request ID %" PRIu32 ": %d\n",
    response.context.request_id, response.rc);

  /* Print ECC response verification result */
  if (response.rc == ECC_OK) {
    printf("SUCCESS\n");
  }
  else {
    printf("FAILURE\n");
  }

  return rc;
}

int main(int argc, char *argv[])
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* ECDSA example message */
  ecdsa_example_message_t message;
  /* HSM capability information */
  hsm_capability_info_t hsm_capability_info =
      HSM_CAPABILITY_INFO_INIT;
  /* HSM emulator configuration */
  hsm_emulator_config_t hsm_emulator_config;
  /* HSM NVM configuration */
  hsm_nvm_config_t hsm_nvm_config = HSM_NVM_CONFIG_INIT;
  /* Remote transport object */
  remote_transport_t *transport = NULL;
  /* Remote IP transport configuration parameters */
  remote_ip_transport_config_t config =
      REMOTE_IP_TRANSPORT_CONFIG_INIT;
  /* Local IPv4 address */
  uint32_t local_ipv4_addr;
  /* Server IPv4 address */
  uint32_t server_ipv4_addr;

  if ((argc != 3) || ((server_ipv4_addr = inet_addr(argv[1])) == INADDR_NONE)) {
    fprintf(stderr, "Usage: %s SERVER-IP4-ADDR LOCAL-IF-NAME\n", argv[0]);
    return EXIT_FAILURE;
  }

  /* Get local IPv4 address */
  rc = remote_util_local_ipv4_address_get(argv[2], &local_ipv4_addr);
  if (atlk_error(rc)) {
    fprintf(stderr, "remote_util_local_ipv4_address_get: %s\n",
            atlk_rc_to_str(rc));
    goto out;
  }

  /* Set remote IP transport configuration parameters */
  config.local_ipv4_address = local_ipv4_addr;
  config.remote_ipv4_address = server_ipv4_addr;
  config.max_rtt_ms = 100;

  /* Create remote IP transport object */
  rc = remote_ip_transport_create(&config, &transport);
  if (atlk_error(rc)) {
    fprintf(stderr, "remote_ip_transport_create: %s\n", atlk_rc_to_str(rc));
    goto out;
  }

  /* Create remote ECC service */
  rc = ecc_remote_service_create(transport, NULL, &example_ecc_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "ecc_remote_service_create: %s\n", atlk_rc_to_str(rc));
    goto out;
  }

  /* Configure HSM emulator for FILE storage */
  hsm_emulator_config.ecc_service_ptr = example_ecc_service;
  hsm_emulator_config.nvm_file_path = HSM_EMULATOR_FILENAME;

  /* Create HSM emulator service */
  rc = hsm_emulator_create(&hsm_emulator_config, &example_hsm_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_emulator_create: %s\n", atlk_rc_to_str(rc));
    goto out;
  }

  /* Get HSM capability information */
  rc = hsm_capability_info_get(example_hsm_service, &hsm_capability_info);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_capability_info_get: %s", atlk_rc_to_str(rc));
    goto out;
  }

  printf("HSM capability information:\n");
  printf("  Maximum number of NVM cells: %u\n",
    hsm_capability_info.max_num_of_cells);
  printf("  Current number of NVM cells: %u\n",
    hsm_capability_info.current_num_of_cells);
  printf("  Maximum number of cell ranges supported by "
         "hsm_csr_ecdsa_public_keys_sign(): %u\n",
    hsm_capability_info.max_num_of_cell_ranges_for_csr);

  printf("Initializing NVM to contain %u cells\n",
    ECDSA_EXAMPLE_HSM_NVM_NUM_CELLS);

  hsm_nvm_config.num_of_cells = ECDSA_EXAMPLE_HSM_NVM_NUM_CELLS;

  /* Initialize HSM NVM */
  hsm_nvm_config.num_of_cells = EXAMPLE_NUM_OF_STORAGE_CELLS;
  rc = hsm_nvm_init(example_hsm_service, &hsm_nvm_config);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_nvm_init: %s", atlk_rc_to_str(rc));
    goto out;
  }

  /* Create ECC socket */
  rc = ecc_socket_create(example_ecc_service, &example_ecc_socket);
  if (atlk_error(rc)) {
    fprintf(stderr, "ecc_socket_create: %s\n",
        atlk_rc_to_str(rc));
    goto out;
  }

  /* Produce example message by Alice */
  rc = ecdsa_example_alice(&message);
  if (atlk_error(rc)) {
    goto out;
  }

  /* Consume example message by Bob */
  rc = ecdsa_example_bob(&message);
  if (atlk_error(rc)) {
    goto out;
  }

out:
  if (atlk_error(rc)) {
    fprintf(stderr, "ERROR\n");
  }
  /* Delete ECC verification socket */
  ecc_socket_delete(example_ecc_socket);

  /* Delete HSM emulator service */
  hsm_service_delete(example_hsm_service);

  /* Delete ECC service */
  ecc_service_delete(example_ecc_service);

  /* Remote transport delete */
  remote_transport_delete(transport);

  return atlk_error(rc);
}
```

8.54 remote-posix/crypto/ecies-example.c

```
/* Copyright (C) 2014-2016 Autotalks Ltd. */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>
#include <pthread.h>
#include <arpa/inet.h>

#include <atlk/sdk.h>
#include <atlk/rng.h>
#include <atlk/sha.h>
#include <atlk/ecc.h>
#include <atlk/ecies.h>
#include <atlk/aes.h>
#include <atlk/ecc_service.h>
#include <atlk/hsm_service.h>
#include <atlk/hsm_emulator.h>

#include <atlk/remote.h>
#include <atlk/ecc_remote.h>

/*
  CRATON Remote ECIES Example

  This example demonstrates a basic ECIES and AES-CCM encryption/decryption
  scenario inspired by IEEE Std. 1609.2-2016 using the HSM API, ECC API and
  RNG API for code running on top of CRATON processor with ThreadX RTOS.

  The device used in this example is a "HSM emulator", a term used to describe
  an emulated HSM device. The differences between an emulated HSM service
  and a real one are:
  - The emulated HSM service is created via hsm_emulator_create().
  - The implementation is not tamper-resistant because it uses general purpose
    hardware instead of tamper-resistant hardware.

  The purpose of the emulated HSM is basic software integration with
  the HSM API on a hardware platform that doesn't have a working HSM chip.
*/

/* HSM emulator filename */
#define HSM_EMULATOR_FILENAME "./hsm-emu.dat"

/* Number of storage cells for HSM */
#define EXAMPLE_NUM_OF_STORAGE_CELLS 128

/* HSM emulator service */
static hsm_service_t *example_hsm_service = NULL;

/* ECC service */
static ecc_service_t *example_ecc_service = NULL;

/* Format string for ECC scalar */
#define ECC_SCALAR_FMT \
  "0x%04x,0x%04x,0x%04x,0x%04x,0x%04x,0x%04x,0x%04x,0x%04x,"  \
  "0x%04x,0x%04x,0x%04x,0x%04x"

/* Format argument list for ecc_scalar_t */
#define ECC_SCALAR_FMT_ARGS(x)                            \
  x.value[0], x.value[1], x.value[2], x.value[3],    \
  x.value[4], x.value[5], x.value[6], x.value[7],    \
  x.value[8], x.value[9], x.value[10], x.value[11]

/* ECIES HMAC key size in octets */
#define ECIES_HMAC_KEY_SIZE 32

/* Number of NVM cells to configure for HSM */
#define ECIES_EXAMPLE_HSM_NVM_NUM_CELLS 128
```

```
/* ECIES example shared infomation between Alice and Bob */
typedef struct {
  /* Elliptic curve used */
  ecc_curve_t curve;

  /* Bob ECC public key */
  ecc_point_t bob_public_key;

} ecies_example_shared_info_t;

static ecies_example_shared_info_t ecies_example_shared_info = {
  .curve = ECC_CURVE_NIST_P224,
  .bob_public_key = ECC_POINT_INIT
};

/* ECIES example Bob's private information */
typedef struct {
  /* Index of the ECC private key */
  hsm_cell_index_t private_key_index;

} ecies_example_bob_info_t;

static ecies_example_bob_info_t ecies_example_bob_info = {
  .private_key_index = HSM_CELL_INDEX_NA
};

/* ECIES example maximum message size in octets */
#define ECIES_EXAMPLE_MESSAGE_MAX_SIZE 64

/* ECIES example message */
typedef struct {
  /* Ephemeral public key used for ECIES encryption */
  ecc_point_t ecies_ephemeral_public_key;

  /* AES-CCM key encrypted using ECIES */
  uint8_t ecies_encrypted_aes_key[AES_KEY_SIZE];

  /* ECIES authentication tag */
  ecies_authentication_tag_t ecies_authentication_tag;

  /* Ciphertext encrypted using AES-CCM */
  uint8_t aes_ccm_ciphertext[ECIES_EXAMPLE_MESSAGE_MAX_SIZE];

  /* Ciphertext size in octets */
  size_t aes_ccm_ciphertext_size;

  /* AES-CCM nonce */
  aes_ccm_nonce_t aes_ccm_nonce;

  /* AES-CCM authentication tag */
  aes_ccm_authentication_tag_t aes_ccm_tag;

} ecies_example_message_t;

static void
ecies_example_print_buffer(const uint8_t *buf, size_t buf_len)
{
  size_t i;

  for (i = 0; i < buf_len; i++) {
    printf("%02x", buf[i]);
  }
}

static atlk_rc_t
ecies_example_init(void)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* Private key information */
  hsm_ecc_private_key_info_t private_key_info =
      HSM_ECC_PRIVATE_KEY_INFO_INIT;
  /* HSM cell index to store Bob's private key */
  hsm_cell_index_t private_key_index = HSM_CELL_INDEX_NA;
  /* Bob's ECC public key */
  ecc_point_t bob_public_key = ECC_POINT_INIT;
  /* HSM capability information */
  hsm_capability_info_t hsm_capability_info =
      HSM_CAPABILITY_INFO_INIT;
  /* HSM NVM configuration */
  hsm_nvm_config_t hsm_nvm_config = HSM_NVM_CONFIG_INIT;

  printf("\n>>> Initialization\n");

  /* Get HSM capability information */
  rc = hsm_capability_info_get(example_hsm_service, &hsm_capability_info);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_capability_info_get: %s", atlk_rc_to_str(rc));
    return rc;
  }

  printf("HSM capability information:\n");
  printf("  Maximum number of NVM cells: %u\n",
    hsm_capability_info.max_num_of_cells);
  printf("  Current number of NVM cells: %u\n",
    hsm_capability_info.current_num_of_cells);
  printf("  Maximum number of cell ranges supported by "
         "hsm_csr_ecdsa_public_keys_sign(): %u\n",
    hsm_capability_info.max_num_of_cell_ranges_for_csr);

  printf("Initializing NVM to contain %u cells\n",
    ECIES_EXAMPLE_HSM_NVM_NUM_CELLS);

  hsm_nvm_config.num_of_cells = ECIES_EXAMPLE_HSM_NVM_NUM_CELLS;

  /* Initialize HSM NVM */
  rc = hsm_nvm_init(example_hsm_service, &hsm_nvm_config);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_nvm_init: %s", atlk_rc_to_str(rc));
    return rc;
  }

  /* Use NIST P-256 elliptic curve and an Isolated key for ECIES algorithm */
  private_key_info.key_curve = ECC_CURVE_NIST_P256;
  private_key_info.key_type = HSM_PRIVATE_KEY_TYPE_ISOLATED;
  private_key_info.key_algorithm = HSM_PUBLIC_KEY_ALGORITHM_ECIES
     ;

  printf("Using elliptic curve ID: %u\n", private_key_info.key_curve);
  printf("Using key_type ID: %u\n", private_key_info.key_type);
  printf("Using key_algorithm ID: %u\n", private_key_info.key_algorithm);

  /* Use the first available cell to store Bob's private key */
  private_key_index = 0;

  printf("Using HSM cell index: %u\n", private_key_index);

  /* Create Bob's private key */
  rc = hsm_ecc_private_key_create(example_hsm_service,
                                  private_key_index,
                                  &private_key_info);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_ecc_private_key_create: %s", atlk_rc_to_str(rc));
    return rc;
  }

  printf("Bob's ECC private key created\n");

  /* Get Bob's public key */
  rc = hsm_ecc_public_key_get(example_hsm_service,
                              private_key_index,
                              &bob_public_key);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_ecc_public_key_get: %s", atlk_rc_to_str(rc));
    return rc;
  }

  assert(bob_public_key.point_type == ECC_POINT_UNCOMPRESSED);

  /* Print retrieved ECC public key */
  printf("Bob's ECC public key created:\n");
  printf("  x: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(bob_public_key.x_coordinate));
  printf("  y: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(bob_public_key.y_coordinate));

  /* Store shared information */
  ecies_example_shared_info.curve = private_key_info.key_curve;
  ecies_example_shared_info.bob_public_key = bob_public_key;

  /* Store Bob's private information */
  ecies_example_bob_info.private_key_index = private_key_index;

  return ATLK_OK;
}

static atlk_rc_t
ecies_example_alice(ecies_example_message_t *msg)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* AES key */
  aes_key_t aes_key = AES_KEY_INIT;
```

```
/* AES-CCM nonce */
aes_ccm_nonce_t aes_ccm_nonce = AES_CCM_NONCE_INIT;
/* AES-CCM authentication tag */
aes_ccm_authentication_tag_t aes_ccm_tag =
    AES_CCM_AUTHENTICATION_TAG_INIT;
/* AES-CCM ciphertext */
uint8_t ciphertext[ECIES_EXAMPLE_MESSAGE_MAX_SIZE] = { 0 };
/* AES-CCM ciphertext size in octets */
size_t ciphertext_size = sizeof(ciphertext);
/* Example message */
static const char example_msg[] =
  "Autotalks - The Confidence of Knowing Ahead";
/* ECIES key */
uint8_t ecies_key[sizeof(aes_key) + ECIES_HMAC_KEY_SIZE] = { 0 };
/* ECIES ephemeral public key */
ecc_point_t ephemeral_public_key = ECC_POINT_INIT;
/* AES key encrypted using ECIES */
uint8_t encrypted_aes_key[AES_KEY_SIZE] = { 0 };
/* AES key size in octets */
size_t encrypted_aes_key_size = sizeof(encrypted_aes_key);
/* ECIES authentication tag */
ecies_authentication_tag_t ecies_authentication_tag =
  ECIES_AUTHENTICATION_TAG_INIT;

printf("\n>>> Alice\n");

/* Print the message data */
printf("Message: %s\n", example_msg);
printf("Message size: %lu\n", (long unsigned int)sizeof(example_msg));

/* Get random AES key */
rc = rng_data_get(&aes_key, sizeof(aes_key));
if (atlk_error(rc)) {
  fprintf(stderr, "rng_data_get: %s", atlk_rc_to_str(rc));
  return rc;
}

/* Print random AES key */
printf("AES key: ");
ecies_example_print_buffer(aes_key.value, sizeof(aes_key));
printf("\n");

/* Get random AES-CCM nonce */
rc = rng_data_get(&aes_ccm_nonce, sizeof(aes_ccm_nonce));
if (atlk_error(rc)) {
  fprintf(stderr, "rng_data_get: %s", atlk_rc_to_str(rc));
  return rc;
}

/* Print random AES-CCM nonce */
printf("AES-CCM nonce: ");
ecies_example_print_buffer(aes_ccm_nonce.value, sizeof(aes_ccm_nonce));
printf("\n");

/* Encrypt message with AES-CCM */
rc = aes_ccm_encrypt(&aes_key,
                     &aes_ccm_nonce,
                     example_msg,
                     sizeof(example_msg),
                     ciphertext,
                     &ciphertext_size,
                     &aes_ccm_tag);
if (atlk_error(rc)) {
  fprintf(stderr, "aes_ccm_encrypt: %s", atlk_rc_to_str(rc));
  return rc;
}

/* Print AES-CCM encrypted message and authentication tag */
printf("AES-CCM encrypted message: ");
ecies_example_print_buffer(ciphertext, ciphertext_size);
printf("\n");

printf("AES-CCM authentication tag: ");
ecies_example_print_buffer(aes_ccm_tag.value, sizeof(aes_ccm_tag));
printf("\n");

/* ECIES key size should be equal to: plaintext size + HMAC key size */

/* Create ECIES key and ephemeral public key */
rc = ecies_key_create(ecies_example_shared_info.curve,
                      &ecies_example_shared_info.bob_public_key,
                      &ephemeral_public_key,
                      ecies_key,
                      sizeof(ecies_key),
                      NULL,
                      0);
if (atlk_error(rc)) {
    fprintf(stderr, "ecdh_secret_create: %s", atlk_rc_to_str(rc));
    return rc;
  }

  assert(ephemeral_public_key.point_type == ECC_POINT_UNCOMPRESSED);

  /* Print ephemeral public key */
  printf("ECC ephemeral public key created:\n");
  printf("  x: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(ephemeral_public_key.x_coordinate));
  printf("  y: " ECC_SCALAR_FMT "\n",
    ECC_SCALAR_FMT_ARGS(ephemeral_public_key.y_coordinate));

  /* Print ECIES key */
  printf("ECIES key: ");
  ecies_example_print_buffer(ecies_key, sizeof(ecies_key));
  printf("\n");

  /* Encrypt AES key using ECIES */
  rc = ecies_encrypt(SHA_256,
                     ecies_key,
                     sizeof(ecies_key),
                     &aes_key,
                     sizeof(aes_key),
                     encrypted_aes_key,
                     &encrypted_aes_key_size,
                     &ecies_authentication_tag);
  if (atlk_error(rc)) {
    fprintf(stderr, "ecies_encrypt: %s", atlk_rc_to_str(rc));
    return rc;
  }

  /* Print ECIES encrypted AES key and authentication tag */
  printf("ECIES encrypted AES key: ");
  ecies_example_print_buffer(encrypted_aes_key, encrypted_aes_key_size);
  printf("\n");

  printf("ECIES authentication tag: ");
  ecies_example_print_buffer(ecies_authentication_tag.value,
    sizeof(ecies_authentication_tag));
  printf("\n");

  /* Produce message */
  memcpy(msg->aes_ccm_ciphertext, ciphertext, ciphertext_size);
  msg->aes_ccm_ciphertext_size = ciphertext_size;
  msg->aes_ccm_nonce = aes_ccm_nonce;
  msg->aes_ccm_tag = aes_ccm_tag;
  memcpy(msg->ecies_encrypted_aes_key, &encrypted_aes_key, AES_KEY_SIZE);
  msg->ecies_authentication_tag = ecies_authentication_tag;
  msg->ecies_ephemeral_public_key = ephemeral_public_key;

  return ATLK_OK;
}

static atlk_rc_t
ecies_example_bob(const ecies_example_message_t *msg)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* ECIES decrypted AES key */
  aes_key_t aes_key = AES_KEY_INIT;
  /* AES key size in octets */
  size_t aes_key_size = sizeof(aes_key);
  /* ECIES key */
  uint8_t ecies_key[sizeof(aes_key) + ECIES_HMAC_KEY_SIZE] = { 0 };
  /* ECIES authentication tag */
  ecies_authentication_tag_t ecies_authentication_tag =
    ECIES_AUTHENTICATION_TAG_INIT;
  /* AES-CCM authentication tag */
  aes_ccm_authentication_tag_t aes_ccm_tag =
      AES_CCM_AUTHENTICATION_TAG_INIT;
  /* AES-CCM plaintext */
  uint8_t plaintext[ECIES_EXAMPLE_MESSAGE_MAX_SIZE] = { 0 };
  /* AES-CCM plaintext size in octets */
  size_t plaintext_size = sizeof(plaintext);
  /* Example failure indication */
  int failed = 1;

  printf("\n>>> Bob\n");

  /* Derive ECIES key */
  rc = hsm_ecies_key_derive(example_hsm_service,
                            ecies_example_bob_info.private_key_index,
                            &msg->ecies_ephemeral_public_key,
                            ecies_key,
                            sizeof(ecies_key),
                            NULL,
                              0);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_ecdh_secret_derive: %s", atlk_rc_to_str(rc));
    return rc;
  }

  /* Print ECIES key */
  printf("Derived ECIES key: ");
  ecies_example_print_buffer(ecies_key, sizeof(ecies_key));
  printf("\n");

  /* Decrypt AES key with ECIES */
  rc = ecies_decrypt(SHA_256,
              ecies_key,
              sizeof(ecies_key),
              msg->ecies_encrypted_aes_key,
              sizeof(msg->ecies_encrypted_aes_key),
              aes_key.value,
              &aes_key_size,
              &ecies_authentication_tag);
  if (atlk_error(rc)) {
    fprintf(stderr, "ecies_decrypt: %s", atlk_rc_to_str(rc));
    return rc;
  }

  /* Print ECIES decrypted AES key and authentication tag */
  printf("ECIES decrypted AES key: ");
  ecies_example_print_buffer(aes_key.value, aes_key_size);
  printf("\n");

  printf("ECIES authentication tag: ");
  ecies_example_print_buffer(ecies_authentication_tag.value,
    sizeof(ecies_authentication_tag));
  printf("\n");

  /* Compare ECIES authentication tags */
  if (memcmp(&ecies_authentication_tag, &msg->ecies_authentication_tag,
      sizeof(ecies_authentication_tag)) != 0) {
    printf("ECIES encryption/decryption failed\n");
    goto out;
  }
  else {
    printf("ECIES encryption/decryption succeeded\n");
  }

  /* Decrypt message using AES-CCM */
  rc = aes_ccm_decrypt(&aes_key,
                       &msg->aes_ccm_nonce,
                       msg->aes_ccm_ciphertext,
                       msg->aes_ccm_ciphertext_size,
                       plaintext,
                       &plaintext_size,
                       &aes_ccm_tag);
  if (atlk_error(rc)) {
    fprintf(stderr, "aes_ccm_decrypt: %s", atlk_rc_to_str(rc));
    return rc;
  }

  /* Print AES-CCM decrypted message and authentication tag */
  printf("Decrypted message: %s\n", plaintext);
  printf("Decrypted message size: %lu\n", (long unsigned int)plaintext_size);

  printf("AES-CCM authentication tag: ");
  ecies_example_print_buffer(aes_ccm_tag.value, sizeof(aes_ccm_tag));
  printf("\n");

  /* Compare AES-CCM authentication tags */
  if (memcmp(&aes_ccm_tag, &msg->aes_ccm_tag, sizeof(aes_ccm_tag)) != 0) {
    printf("AES-CCM encryption/decryption failed\n");
    goto out;
  }
  else {
    printf("AES-CCM encryption/decryption succeeded\n");
  }

  /* Set failure indication flag */
  failed = 0;

out:
  printf("%s\n", failed ? "FAILURE" : "SUCCESS");

  return ATLK_OK;
}

int main(int argc, char *argv[])
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* ECDSA example message */
  ecies_example_message_t message;
  /* HSM emulator configuration */
  hsm_emulator_config_t hsm_emulator_config =
      HSM_EMULATOR_CONFIG_INIT;
  /* Remote transport object */
  remote_transport_t *transport = NULL;
  /* Remote IP transport configuration parameters */
  remote_ip_transport_config_t config =
      REMOTE_IP_TRANSPORT_CONFIG_INIT;
  /* Local IPv4 address */
  uint32_t local_ipv4_addr;
  /* Server IPv4 address */
  uint32_t server_ipv4_addr;

  if ((argc != 3) || ((server_ipv4_addr = inet_addr(argv[1])) == INADDR_NONE)) {
    fprintf(stderr, "Usage: %s SERVER-IP4-ADDR LOCAL-IF-NAME\n", argv[0]);
    return EXIT_FAILURE;
  }

  /* Get local IPv4 address */
  rc = remote_util_local_ipv4_address_get(argv[2], &local_ipv4_addr);
  if (atlk_error(rc)) {
    fprintf(stderr, "remote_util_local_ipv4_address_get: %s\n",
            atlk_rc_to_str(rc));
    goto out;
  }

  /* Set remote IP transport configuration parameters */
  config.local_ipv4_address = local_ipv4_addr;
  config.remote_ipv4_address = server_ipv4_addr;
  config.max_rtt_ms = 100;

  /* Create remote IP transport object */
  rc = remote_ip_transport_create(&config, &transport);
  if (atlk_error(rc)) {
    fprintf(stderr, "remote_ip_transport_create: %s\n", atlk_rc_to_str(rc));
    goto out;
  }

  /* Create remote ECC service */
  rc = ecc_remote_service_create(transport, NULL, &example_ecc_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "ecc_remote_service_create: %s\n", atlk_rc_to_str(rc));
    goto out;
  }

  hsm_emulator_config.ecc_service_ptr = example_ecc_service;
  hsm_emulator_config.nvm_file_path = HSM_EMULATOR_FILENAME;

  /* Create HSM emulator service */
  rc = hsm_emulator_create(&hsm_emulator_config, &example_hsm_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "hsm_emulator_create: %s\n", atlk_rc_to_str(rc));
    goto out;
  }

  /* Initialize example message */
  rc = ecies_example_init();
  if (atlk_error(rc)) {
    goto out;
  }

  /* Produce example message by Alice */
  rc = ecies_example_alice(&message);
  if (atlk_error(rc)) {
    goto out;
  }

  /* Consume example message by Bob */
  rc = ecies_example_bob(&message);
  if (atlk_error(rc)) {
    goto out;
  }

out:
  if (atlk_error(rc)) {
    fprintf(stderr, "ERROR\n");
  }

  /* Delete HSM emulator service */
  hsm_service_delete(example_hsm_service);

  return atlk_error(rc);
}
```

8.55 remote-posix/gnss/gnss-example.c

```
/* Copyright (C) 2015-2016 Autotalks Ltd. */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <inttypes.h>

#include <atlk/uart.h>
#include <atlk/gnss.h>

/*
  POSIX GNSS  Example

  This example demonstrates basic usage of GNSS API for code running
  externally to CRATON processor with an OS supporting POSIX.
*/

/* GNSS thread priority */
#define GNSS_THREAD_PRIORITY 20

/* GNSS device name */
#define GNSS_DEV_NAME "/dev/ttyUSB0"

/* Implementation of the navigation service instance */
static struct nav_service {
  int __dummy;
} nav_service;

/* Registered GNSS navigation data frame handler */
static atlk_rc_t
nav_data_publish(nav_service_t *service, nav_data_t *data)
{
  (void)service;

  printf("nav_data_publish called!\n");

  switch (data->data_type) {
  case NAV_DATA_TYPE_FIX:
  {
    nav_fix_t *fix = &data->data.fix;

    printf("time: %.1f sec (err: %.4f sec), leap: %d sec%s\n",
           fix->time.tai_seconds_since_2004, fix->
      error_time_s,
           fix->time.leap_seconds_since_2004,
           fix->time.positive_leap_second ? " (positive leap second)" : "");

    printf("lat: %.7f deg, lon: %.7f deg, alt: %.1f m (err: %.1f m)\n",
           fix->position_latitude_deg, fix->
      position_longitude_deg,
           fix->position_altitude_m, fix->
      error_position_altitude_m);

    printf("err ellipse: hdg: %.1f deg, major len: %.1f m, minor len: %.1f m\n",
           fix->error_position_horizontal_major_axis_direction_deg
      ,
           fix->error_position_horizontal_semi_major_axis_length_m
      ,
           fix->error_position_horizontal_semi_minor_axis_length_m
      );

    printf("heading: %.2f deg (err: %.2f deg)\n",
           fix->movement_horizontal_direction_deg,
           fix->error_movement_horizontal_direction_deg);

    printf("speed: %.1f mps (err: %.1f mps), v-speed: %.1f mps (err: %.1f mps)\n",
           fix->movement_horizontal_speed_mps,
           fix->error_movement_horizontal_speed_mps,
           fix->movement_vertical_speed_mps,
           fix->error_movement_vertical_speed_mps);

    printf("mode: %d, data source: 0x%" PRIu32 ", hdop: %.2f\n",
           fix->mode, fix->data_source, fix->hdop);

    printf("sat in use: %d, GP in view: %d, GL in view: %d\n",
           fix->satellites_in_use_num,
           fix->satellites_num[NAV_SATELLITES_GPS],
           fix->satellites_num[NAV_SATELLITES_GLONASS]);
    break;
  }
  case NAV_DATA_TYPE_SATELLITE_REPORT:
  {
    nav_satellite_report_t *sat = &data->data.
      satellite_report;

    printf("time: %.1f sec, leap: %d sec%s\n",
          sat->time.tai_seconds_since_2004,
          sat->time.leap_seconds_since_2004,
          sat->time.positive_leap_second ? " (positive leap second)" : "");
    for (size_t i = 0; i < sat->satellite_info_array_size; ++i) {
      printf("[%zd] %s prn: %u, elev: %u deg%s, azimuth %u deg%s, %u db%s\n",
             i + 1, (sat->satellite_info_array[i].
      satellite_system ==
                       NAV_SATELLITES_GPS) ? "GP" : "GL",
             sat->satellite_info_array[i].prn_num,
             sat->satellite_info_array[i].elevation_deg,
             (sat->satellite_info_array[i].elevation_deg ==
              NAV_SATELLITE_INFO_ELEVATION_DEG_NA) ? " (unknown)" : "",
             sat->satellite_info_array[i].azimuth_deg,
             (sat->satellite_info_array[i].azimuth_deg ==
              NAV_SATELLITE_INFO_AZIMUTH_DEG_NA) ? " (unknown)" : "",
             sat->satellite_info_array[i].cnr_db,
             (sat->satellite_info_array[i].cnr_db ==
              NAV_SATELLITE_INFO_CNR_DB_NA) ? " (not tracked)" : "");
    }
    break;
  }
  default:
    break;
  }

  return ATLK_OK;
}

int main(int argc, char *argv[])
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* GNSS configuration parameters */
  gnss_config_t config = GNSS_CONFIG_INIT;
  /* Not using argc */
  (void)argc;
  /* Not using argv */
  (void)argv;

  printf("Start GNSS example...\n");

  /* Set GNSS configuration parameters */
  config.model = GNSS_MODEL_STMICRO_TESEO_II;
  config.dev_name = GNSS_DEV_NAME;
  config.nmea_speed_bps = UART_SPEED_230400_BPS;
  config.nmea_cycle_ender_10hz = "$XXGLL";
  config.nmea_cycle_ender_1hz = "$PSTMCPU";
  config.sched_params.priority = GNSS_THREAD_PRIORITY;
  config.handler = nav_data_publish;
  config.service = &nav_service;

  /* Initialize GNSS */
  rc = gnss_init(&config);
  if (atlk_error(rc)) {
    fprintf(stderr, "gnss_init: %s\n", atlk_rc_to_str(rc));
    return EXIT_FAILURE;
  }

  while (1) {
    usleep(1000000);
  }

  return EXIT_SUCCESS;
}
```

8.56 remote-posix/mibs/mibs-example.c

```
/* Copyright (C) 2013-2015 Autotalks Ltd. */
#include <stdio.h>
#include <stdlib.h>
#include <arpa/inet.h>

#include <atlk/mib_service.h>
#include <atlk/remote.h>
#include <atlk/mib_remote.h>

#include <atlk/mibs/wlan-mib.h>
#include <atlk/mibs/snmpv2-mib.h>

/*
  Remote POSIX MIBs Example

  This example demonstrates basic usage of the remote MIB API for code
  running externally to CRATON processor with an OS supporting POSIX.

  The MIB API mirrors Autotalks proprietary MIBs as well as selected MIB
  attributes from standard MIBs.

  The example demonstrates how to set the frequency of interface 1 to 5880
  MHz using WLAN MIB API (which mirrors AUTOTALKS-WLAN-MIB.mib) and how to
  get the system description via SNMPv2 MIB API (which mirrors the standard
  SNMPv2-MIB.mib).
*/

/* Size of system description string in bytes used in this example */
#define SYS_DESCR_SIZE 300

int main(int argc, char *argv[])
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* Remote transport object */
  remote_transport_t *transport = NULL;
  /* Remote IP transport configuration parameters */
  remote_ip_transport_config_t config =
      REMOTE_IP_TRANSPORT_CONFIG_INIT;
  /* Local IPv4 address */
  uint32_t local_ipv4_addr;
  /* Server IPv4 address */
  uint32_t server_ipv4_addr;
  /* System description string */
  char sys_descr[SYS_DESCR_SIZE];
  /* Size of description string in bytes */
  size_t sys_descr_size = sizeof(sys_descr);
  /* MIB service */
  mib_service_t *mib_service = NULL;

  if ((argc != 3) || ((server_ipv4_addr = inet_addr(argv[1])) == INADDR_NONE)) {
    fprintf(stderr, "Usage: %s SERVER-IP4-ADDR LOCAL-IF-NAME\n", argv[0]);
    return EXIT_FAILURE;
  }

  /* Get local IPv4 address */
  rc = remote_util_local_ipv4_address_get(argv[2], &local_ipv4_addr);
  if (atlk_error(rc)) {
    fprintf(stderr, "remote_util_local_ipv4_address_get: %s\n",
            atlk_rc_to_str(rc));
    goto exit;
  }

  /* Set remote IP transport configuration parameters */
  config.local_ipv4_address = local_ipv4_addr;
  config.remote_ipv4_address = server_ipv4_addr;
  config.max_rtt_ms = 1500;

  /* Create remote IP transport object */
  rc = remote_ip_transport_create(&config, &transport);
  if (atlk_error(rc)) {
    fprintf(stderr, "remote_ip_transport_create: %s\n", atlk_rc_to_str(rc));
    goto exit;
  }

  /* Create remote MIB service */
  rc = mib_remote_service_create(transport, NULL, &mib_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "mib_remote_service_create: %s\n", atlk_rc_to_str(rc));
    goto exit;
  }

  /* Set frequency at interface 1 to 5880 MHz */
  rc = mib_set_wlanFrequency(mib_service, 1, 5880);
  if (atlk_error(rc)) {
    fprintf(stderr, "mib_set_wlanFrequency: %s\n", atlk_rc_to_str(rc));
    goto exit;
  }
  printf("Frequency at interface 1 set to 5880 MHz.\n");

  /* Get system description */
  rc = mib_get_sysDescr(mib_service, sys_descr, &sys_descr_size);
  if (atlk_error(rc)) {
    fprintf(stderr, "mib_get_sysDescr: %s\n", atlk_rc_to_str(rc));
    goto exit;
  }
  printf("System description: %s\n", sys_descr);

exit:
  /* Clean-up resources */
  mib_service_delete(mib_service);
  remote_transport_delete(transport);

  return atlk_error(rc);
}
```

8.57 remote-posix/v2x/v2x-example.c

```
/* Copyright (C) 2013-2015 Autotalks Ltd. */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <pthread.h>
#include <arpa/inet.h>

#include <atlk/v2x.h>
#include <atlk/v2x_service.h>

#include <atlk/remote.h>
#include <atlk/v2x_remote.h>

/*
  Remote POSIX V2X Example

  This example demonstrates basic usage of the remote V2X API for code
  running externally to CRATON processor with an OS supporting POSIX.

  Two threads are created -- a TX thread and a RX thread. A V2X service is
  retrieved and a V2X socket is created; these are used by both threads.

  The TX thread sends a broadcast frame with protocol ID 0x102. The RX thread
  receives frames with protocol ID 0x102 and prints their content as well as
  receive power.
*/

/* TX thread */
static pthread_t tx_thread;
static void *tx_thread_entry(void *arg);

/* RX thread */
static pthread_t rx_thread;
static void *rx_thread_entry(void *arg);

/* Interface index used in this example */
#define IF_INDEX 1

/* Protocol identifier used in this example */
#define PROTO_ID 0x102ULL

/* Shared V2X socket */
static v2x_socket_t *v2x_socket = NULL;

/* Example message format string: Example <seq_num> */
static const char msg_fmt[] = "Example %u";

/* Example message string maximum length */
static const size_t msg_size_max = sizeof(msg_fmt) + 10;

int main(int argc, char *argv[])
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* POSIX return value */
  int rv = 0;
  /* Remote transport object */
  remote_transport_t *transport = NULL;
  /* Remote IP transport configuration parameters */
  remote_ip_transport_config_t config =
      REMOTE_IP_TRANSPORT_CONFIG_INIT;
  /* Local IPv4 address */
  uint32_t local_ipv4_addr;
  /* Server IPv4 address */
  uint32_t server_ipv4_addr;
  /* V2X service */
  v2x_service_t *v2x_service = NULL;
  /* V2X socket configuration parameters */
  v2x_socket_config_t v2x_socket_config =
      V2X_SOCKET_CONFIG_INIT;

  if ((argc != 3) || ((server_ipv4_addr = inet_addr(argv[1])) == INADDR_NONE)) {
    fprintf(stderr, "Usage: %s SERVER-IP4-ADDR LOCAL-IF-NAME\n", argv[0]);
    return EXIT_FAILURE;
  }

  /* Get local IPv4 address */
  rc = remote_util_local_ipv4_address_get(argv[2], &local_ipv4_addr);
  if (atlk_error(rc)) {
    fprintf(stderr, "remote_util_local_ipv4_address_get: %s\n",
            atlk_rc_to_str(rc));
    goto exit;
  }
  /* Set remote IP transport configuration parameters */
  config.local_ipv4_address = local_ipv4_addr;
  config.remote_ipv4_address = server_ipv4_addr;
  config.max_rtt_ms = 10;

  /* Create remote IP transport object */
  rc = remote_ip_transport_create(&config, &transport);
  if (atlk_error(rc)) {
    fprintf(stderr, "remote_ip_transport_create: %s\n", atlk_rc_to_str(rc));
    goto exit;
  }

  /* Create the V2X service */
  rc = v2x_remote_service_create(transport, NULL, &v2x_service);
  if (atlk_error(rc)) {
    fprintf(stderr, "v2x_remote_service_create: %s\n", atlk_rc_to_str(rc));
    goto exit;
  }

  /* Set V2X socket configuration parameters */
  v2x_socket_config.if_index = IF_INDEX;
  v2x_socket_config.protocol.frame_type = V2X_FRAME_TYPE_DATA;
  v2x_socket_config.protocol.protocol_id = PROTO_ID;

  /* Create V2X socket */
  rc = v2x_socket_create(v2x_service, &v2x_socket, &v2x_socket_config);
  if (atlk_error(rc)) {
    fprintf(stderr, "v2x_socket_create: %s\n", atlk_rc_to_str(rc));
    goto exit;
  }

  /* Create TX thread */
  rv = pthread_create(&tx_thread, NULL, tx_thread_entry, NULL);
  if (rv) {
    fprintf(stderr, "pthread_create: %s\n", strerror(rv));
    rc = ATLK_E_UNSPECIFIED;
    goto exit;
  }

  /* Create RX thread */
  rc = pthread_create(&rx_thread, NULL, rx_thread_entry, NULL);
  if (rv) {
    fprintf(stderr, "pthread_create: %s\n", strerror(rv));
    rc = ATLK_E_UNSPECIFIED;
    goto exit;
  }

  /* Wait forever */
  while (1) {
    usleep(1000000);
  }

exit:
  /* Clean-up resources */
  v2x_socket_delete(v2x_socket);
  v2x_service_delete(v2x_service);
  remote_transport_delete(transport);

  return atlk_error(rc);
}

static void *tx_thread_entry(void *arg)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* Send parameters */
  v2x_send_params_t send_params = V2X_SEND_PARAMS_INIT;
  /* Message counter */
  uint32_t msg_count = 0;
  /* Not using input */
  (void)arg;

  /* Set transmit power to -10 dBm */
  send_params.power_dbm8 = -80;

  while (1) {
    /* TX buffer */
    char buf[msg_size_max];

    /* Print message into buffer (with terminating \0) and update its size */
    size_t size = 1 + snprintf(buf, sizeof(buf), msg_fmt, msg_count);
    msg_count++;

    printf("Example: Broadcast TX: \"%s\"\n", buf);

    /* Transmit V2X PDU */
    rc = v2x_send(v2x_socket, buf, size, &send_params, NULL);
    if (atlk_error(rc)) {
      fprintf(stderr, "v2x_send: %s\n", atlk_rc_to_str(rc));
      return NULL;
    }

    /* Sleep 1 second between transmissions */
    usleep(1000000);
  }

  return NULL;
}

static void *rx_thread_entry(void *arg)
{
  /* Autotalks return code */
  atlk_rc_t rc = ATLK_OK;
  /* Not using input */
  (void)arg;

  while (1) {
    /* RX buffer */
    char buf[msg_size_max];
    /* RX size */
    size_t size = sizeof(buf);
    /* Received V2X parameters */
    v2x_receive_params_t receive_params =
      V2X_RECEIVE_PARAMS_INIT;

    /* Receive frame (wait forever until it arrives) */
    rc = v2x_receive(v2x_socket, buf, &size, &receive_params,
                     &atlk_wait_forever);
    if (atlk_error(rc)) {
      fprintf(stderr, "v2x_receive: %s\n", atlk_rc_to_str(rc));
      return NULL;
    }

    /* Print source address of received frame */
    printf("Example: RX from %02x:%02x:%02x:%02x:%02x:%02x\n",
           receive_params.source_address.octets[0],
           receive_params.source_address.octets[1],
           receive_params.source_address.octets[2],
           receive_params.source_address.octets[3],
           receive_params.source_address.octets[4],
           receive_params.source_address.octets[5]);

    /* Obtain data as zero-terminated string */
    if (buf[size - 1] != '\0') {
      printf("* Bad message (not zero-terminated)\n");
    }
    else {
      printf("* Message: \"%s\"\n", buf);
    }

    /* Print RX power */
    if (receive_params.power_dbm8 != V2X_POWER_DBM8_NA) {
      printf("* RX power: %.2f dBm\n",
             (double)receive_params.power_dbm8 /
      V2X_POWER_DBM8_PER_DBM);
    }
  }

  return NULL;
}
```

Index