FUSION A0 / A1 / B0

SE Host Services API and User Guide

V 0.0.28

March 2023

Table of Contents

Contents

TODO

- OSPI needs example code to show setting of EXTERNAL / INTERNAL Keys usage.
- SERVICES APIS to add
  - Dynamic
    - Interrupt muxing
    - Event routing
  - Image loading
  - Deferred boot
  - PSA

Introduction

The Fusion product series is a scalable SoC solution for IoT Edge Computing platforms.

SERVICES provide a method for the Application CPUs (M55_HE, HP, A32) to communicate with the Secure Enclave. This secure communication path is achieved using the MHU (Message Handling Unit) hardware block.

The SERVICES library consists of C code that interfaces with an MHU driver to facilitate this communication.

Services fall into the following categories:

- Maintenance Services
- Crypto Services
- Update Services
- Secure Debug Service
- Application Services

The library source code is provided along with a test harness showing the invocation of each SERVICE library call.

The example (test harness) can be used as a framework to copy for integrating SERVICES into your application code. You should only need to include the SERVICE header files and link with the pre-built SERVICE libraries provided.

The examples can be built for all Application cores in XIP and Non-XIP mode. ATOC configuration files are also provided. The examples are stand-alone but can be built using the ALIF Ensemble CMSIS delivery.

ARM Clang and ARM GNU CC are both supported. **Note:** that the pre-built libraries are compiled using ARM Clang. Just run `make realclean` and rebuild everything.

The pre-built libraries were intended to quicken the process of integrating SERVICES into an application. For real development it is recommended to add the SERVICES to your build process

Version History

Version	Type	Change	
V0.0.28	API	Rename standby API Corstone clocks configuration as services	SE-1447 SE-1684
V0.0.27	BUG	Fix warnings in service release	SE-1709
V0.0.26	Feature	PLL Service API exposure Service for global standby mode added APIs for Finer grain Retention control clean up Makefiles CMSIS 0.5.2 updates	SE-1660 SE-1609 SE-1608 SE-1658 SE-1676
V0.0.25	Feature	CMake M55_power added	SE-1645
V0.0.24		REV_B0 release including Power example	SE-1612
V0.0.23	Feature	API Support for RTSS 0.5.0 CMSIS release	SE-1585
V0.0.22	Feature	Error codes are transport related only LocalToGlobal address translation changes Split actual message sending from SERVICES_send_msg()	
V0.0.21	Feature	Adding SPARK build flag	
V0.0.20	Feature	Update license headers	SE-1512
		RISC-V reset SERVICE	SE-1511
V0.0.19	Feature	Adding CMake build option, README.md file added	
V0.0.18	BUG Fixes	See limitations	
V0.0.17	BUG Fixes	See limitations	
V0.0.16	<>		
V0.0.15	BUG Fixes	See limitations	
V0.0.14	Example	Test harness updated	
V0.0.13	Startup	Updated m55 startup code	
V0.0.12	API	Added SERVICES_system_read_otp	
V0.0.11	API	Added SERVICES_system_get_toc_data	
V0.0.10	API	standardized variables for send/resp	
V0.0.9	API	SERVICES_uart_write added size parameter	
		New Error code SERVICES_REQ_BAD_PRINT_LENGTH	
		New Error code SERVICES_REQ_NULL_PARAMETER	

Limitations

Current releases support

- A32, M55-HE, M55-LE Application CPUs
- Bare metal systems
- LINUX

NOTE: There are numerous issues with bare metal A32, currently these are not a priority to resolve as M55_HE/HP are the focus. A32 non-xip example does work as advertised with armclang.

Known release issues:

Limitation	Notes
[JIRA] (SE-1683) [SERVICES] A32 XIP does not boot from MRAM	XIP=OFF works
[JIRA] (SE-1681) [SERVICES] A32 XIP json file has overlaping address	
[JIRA] (SE-1680) [SERVICES] makefile.gnu all install - HP image is missing	Build CPU=M55_HP XIP=ON install to workaround
[JIRA] (SE-1660) [SERVICES] PLL Service API exposure	
[JIRA] (SE-1659) [SERVICES] Revisit Header file organization	
[JIRA] (SE-1649) [SERVICES] hardfault when writing to RTC_A register	
[JIRA] (PSBT-151) L6242E: Cannot link object services_host_handler.o as its attributes are incompatible with the image attributes.	Linker error caused by default compiler settings
[JIRA] (SE-1616) [SERVICES] OTP read error return	No error code return
[JIRA] (SE-1582) [SERVICES][REV_B0] Services examples do not run from MRAM	**REV_B0** only
[JIRA] (SE-1576) [SERVICES] Services UART print is always enabled	SERVICES print doesn't go to Logger
[JIRA] (SE-1526) [SERVICES] A32 XIP example fails	SERAM does not launch. Use maintenance to remove from ATOC MRAM
[JIRA] (SE-1510) [SERVICES] CMake support for A32 build	
[JIRA] (SE-1444) [SERVICES] GNU compilation for A32 Services examples	
[JIRA] (SE-1441) [SERVICES] A32 service requests NACKed	
[JIRA] (SE-1349) [SERVICES] A32 example output incorrect	
[JIRA] (SE-1328) [SERVICES] A32 link register address incorrect	

0.0.28

- <>

0.0.27

- [JIRA] (SE-1709) [SERVICES] Fix warnings in service release -> RESOLVED

0.0.26

- [JIRA] (SE-1676) [SERVICES] CMSIS 0.5.2 updates -> RESOLVED
- [JIRA] (PSBT-150) RTSS 0.5.0 compiler warning -> RESOLVED (With CMSIS V0.5.2)

- [JIRA] (PSBT-149) Spelling mistakes -> RESOLVED
- [JIRA] (SE-1658) [SERVICES] clean up Makefiles -> RESOLVED
- [JIRA] (SE-1631) [SERVICES] build all (GNU) fails with A32 build -> RESOLVED. Now skips the A32 build rather than Fail.
- JIRA] (SE-1509) [SERVICES] GNU C Compilations lack XIP examples -> RESOLVED

0.0.25

- GCC sample ld files added for XIP
- [JIRA] (SE-1680) [SERVICES] makefile.gnu all install - HP image is missing
- [JIRA] (SE-1628) [SERVICES] Standby mode SRAM0 SRAM1 retention -> RESOLVED
- [JIRA] (SE-1674) [SERVICES] Overlap regions for HE-HP example -> RESOLVED
- [JIRA] (SE-1497) [SERVICES] TEST Services initialize polling -> RESOLVED
- [JIRA] (SE-1644) [SERVICES] Unsupported build matrix options should error not fail -> RESOLVED
- [JIRA] (SE-1645) [SERVICES] power example GCC build -> RESOLVED
- [JIRA] (PSBT-150) RTSS 0.5.0 compiler warning

0.0.24

- [JIRA] (SE-1674) [SERVICES] Overlap regions for HE-HP example
- Make option (cmsis override) – HELP banner shows 'make cmsis CMSIS=x.y.z', syntax is make cmsis=x.y.z
- [JIRA] (SE-1652) [SETOOLS] APP release builder uses wrong cfg for REV_B0 – app-cpu.cfg has wrong address 0x60000000 instead of 0x58000000. This is a bug in the release builder copying the REV_A1 JSON file instead of REV_B0 to the release.
- [JIRA] (PSBT-150) RTSS 0.5.0 compiler warning – during the build there is a –noreturn warning printed.
- [JIRA] (SE-1649) [SERVICES] hardfault when writing to RTC_A register – temporarily commented out (example still runs) until this is fixed.
- [JIRA] (SE-1631) [SERVICES] build all (GNU) fails with A32 build – use Make clean and rebuild.
- [JIRA] (SE-1612) [SERVICES] Not able to see the cpu id information for M55_HP -> RESOLVED

0.0.23

- [JIRA] (SE-1612) [SERVICES] Not able to see the cpu id information for M55_HP
- [JIRA] (PSBT-150) RTSS 0.5.0 compiler warning – You will see a compiler warning for the Reset_Handler regarding NoRETURN.
- [JIRA] (PSBT-151) L6242E: Cannot link object services_host_handler.o as its attributes are incompatible with the image attributes.
- [JIRA] (PSBT-149) Spelling mistakes
- JIRA] (SE-1447) [SERVICES] Spelling errors (In the LD files from APPS) -> RESOLVED

0.0.22

- [JIRA] (SE-1582) [SERVICES][REV_B0] Services examples do not run from MRAM
- [JIRA] (SE-1576) [SERVICES] Services UART print is always enabled
- [JIRA] (SE-1553) [SERVICES] make all for GNU still using ARM-Clang

0.0.21

- <>

0.0.20

- JIRA] (SE-1502) [SERVICES] Extra bracket character in services_lib_api.h causing build failure with C++ -> RESOLVED

0.0.19

- CMake - builds *only* for M55_HE TCM load (SE-1509)
- [JIRA] (SE-1526) [SERVICES] A32 XIP example fails -> NEW

0.0.18

- [JIRA] (SE-1496) [SERVICES] GCC Compilation warning [-Wunused-variable] -> RESOLVED
- [JIRA] (SE-1487) [SERVICES] A32 example missing json file -> RESOLVED
- [JIRA] (SE-1442) [SERVICES] unused function in MHU driver -> RESOLVED
- [JIRA] (SE-1427) [SERVICES] A32 example crashes when run from MRAM

0.0.17

- [JIRA] (SE-1487) [SERVICES] A32 example missing json file -> NEW
- [JIRA] (SE-1465) [SERVICES] debug output is enabled -> RESOLVED
- [JIRA] (SE-1471) [SERVICES] DOS based make issue -> RESOLVED
- [JIRA] (SE-1443) [SERVICES] newlib link warnings -> RESOLVED

0.0.16

- A32 example is not part of the GNU C compilation.
- [JIRA] (SE-1439) [SERVICES] Compiler warnings with GCC -> RESOLVED
- [JIRA] (SE-1442) [SERVICES] unused function in MHU driver -> RESOLVED

0.0.15

- A32 example is not part of the GNU C compilation. This was not done as there were issues with A32 example fixed in this sprint.
- [JIRA] (SE-1443) [SERVICES] newlib link warnings - with GNU C warnings are seen with SERVICES library link
- [JIRA] (SE-1442) [SERVICES] unused function in MHU driver – warning seen with GNU about unused function
- [JIRA] (SE-1349) [SERVICES] A32 example output incorrect

0.0.14

- [JIRA](SE-1423) TTY Output - <unknown> is passed for CPU type in Library revision output

0.0.13

- A32 crash seen when booting from MRAM

0.0.12

- <>

0.0.11

- OTP read is “not implemented”, awaiting SES change for OTP API reads.

0.0.10

- <>

0.0.9

- UART prints via Services are maximum size of 256 bytes.
- Pin and Pad control messages can be seen as part of the SE-UART output even if the SERVICE debug is DISABLED. This is a bug and will be fixed.

0.0.8

- RTSS / CMSIS require edits to be made to the SERVICES example to work under ARM-DS (MiniTOC). Integration of a unified MHU driver is still not completed for CMSIS.

0.0.7

- RTSS / CMSIS builds do not support REV_B0 devices. It will build, but execution fails.

0.0.6

- RTSS / CMSIS build causes compiler error with redefinition of M55_HE preprocessor flag.

V46

- <>

V42

- <>

V41

- Other running applications can interfere with operation of the Services. The MHU interrupts can stop. Suggest that other Applications, such as the Blinky, are removed and replaced with a debug stub
- Stepping over the init() function in the M55 Examples actually enters the function, if this happens just jump out of this function

In V42 it is no longer needed to put a debug stub on an M55 core before running the Services example on it. An Arm DS debugger script was added that initializes certain M55 registers, and combined with a change in the code, puts the M55 in a known state that doesn't cause issues. The details are in the section 'Running the M55 HE Host Example'.

SE Host Services Versions with SERAM Pairings

Version	SERAM	JIRAs	Notes
0.028	V69		
0.027	V68		
0.026			
0.025	V67		
0.024	V66		
0.023	V65		
0.022	V64		
0.021	V63	SE-1548	SPARK flag build
0.020	V62	SE-1511	Add FUSION_EXTERNAL_SYS0
0.0.16	V58		OTP addition
0.0.11	V51	SE-1265	Get all TOC data
0.0.10	V50	SE-1250	EWiC configuration call
0.0.9	V49		
0.0.8	V47	SE-1188	Header file change
0.0.7	V47	SE-1165	Mbed TLS accelerators
0.0.6	V46	SE-1173	API for enable / disable SES debug status
0.0.5	V46	SE-1144	API changes for Error code
0.0.4		SE-700	Boot reset addition
0.0.3		SE-827	RPC Parameter changes
0.0.2		SE-708	First refactoring
0.0.1			First implementation

Supported SE Host Services summary

Service Group	Ax	B0		Notes
Maintenance				
	●	●	SERVICES_heartbeat	Health status
	●	●	SERVICES_heartbeat_async	
System Management				
	●	●	SERVICES_system_get_toc_data	
	●	●	SERVICES_system_get_toc_number	
	●	●	SERVICES_system_get_toc_via_name	
	●	●	SERVICES_system_get_device_part_number	
	●	●	SERVICES_system_get_toc_via_cpuid	
	●	●	SERVICES_system_get_device_part_number	
	●	●	SERVICES_system_get_toc_version	N/I
	●	●	SERVICES_system_set_services_debug	Debug toggle
	●	●	SERVICES_get_se_revision	
	●	●	SERVICES_system_get_otp_data	N/I
	●	●	SERVICES_system_read_otp	
Application / Pin mux management				
	●	●	SERVICES_pinmux	
	●	●	SERVICES_padcontrol	
	●	●	SERVICES_uart_write	
		●	SERVICES_application_ospi_write_key	**REV_B0**
	●	●	SERVICES_SRAM_retention_config	
Power				
		●	SERVICES_power_stop_mode_request	**REV_B0**
		●	SERVICES_power_ewic_config	**REV_B0**
		●	SERVICES_power_wakeup_config	**REV_B0**
		●	SERVICES_power_mem_retrntion_config	**REV_B0**
		●	SERVICES_power_m55_he_vtor_save	**REV_B0**
		●	SERVICES_power_m55_hp_vtor_save	**REV_B0**
		●	SERVICES_global_standby_mode	**REV_B0**
Security /Crypto				
	●	●	SERVICES_cryptocell_get_lcs	LCS State
	●	●	SERVICES_cryptocell_get_rnd	TRNG
	●	●	SERVICES_cryptocell_mbedtls_aes_init	

	●	●	SERVICES_cryptocell_mbedtls_aes_set_key	
	●	●	SERVICES_cryptocell_mbedtls_aes_crypt	
	●	●	SERVICES_cryptocell_mbedtls_sha_starts	
	●	●	SERVICES_cryptocell_mbedtls_sha_process	
	●	●	SERVICES_cryptocell_mbedtls_sha_update	
	●	●	SERVICES_cryptocell_mbedtls_sha_finish	
	●	●	SERVICES_cryptocell_mbedtls_ccm_gcm_set_key	
	●	●	SERVICES_cryptocell_mbedtls_ccm_gcm_crypt	
	●	●	SERVICES_cryptocell_mbedtls_ccm_gcm_chachapoly_crypt	
	●	●	SERVICES_cryptocell_mbedtls_ccm_gcm_poly1305_crypt	
	●	●	SERVICES_cryptocell_mbedtls_cmac_init_setkey	
	●	●	SERVICES_cryptocell_mbedtls_cmac_update	
	●	●	SERVICES_cryptocell_mbedtls_cmac_finish	
	●	●	SERVICES_cryptocell_mbedtls_cmac_reset	
Boot				
	●	●	SERVICES_boot_process_toc_entry	
	●	●	SERVICES_boot_cpu	
	●	●	SERVICES_boot_release_cpu	
	●	●	SERVICES_boot_reset_cpu	
	●	●	SERVICES_boot_reset_soc	
Clock				
		●	SERVICES_clocks_select_osc_source	
		●	SERVICES_clocks_select_pll_source	
		●	SERVICES_clocks_enable_clock	
		●	SERVICES_clocks_set_ES0_frequency	
		●	SERVICES_clocks_set_ES1_frequency	
		●	SERVICES_clocks_select_a32_source	
		●	SERVICES_clocks_select_aclk_source	
		●	SERVICES_clocks_set_divider	

N/I - Not implemented means the SERVICE exists but is not fully completed in SES.

SE Host Services Delivery Components

A release package from ALIF consists of the following components:

- Source code SERVICES library
  - Public header files
- Makefile(s) for Linux, ARM Clang and ARM GNU C builds
- Examples
  - Example ports for Bare metal and Linux
    - Example SERVICE library initializations.
  - Example use cases for M55_HE, M55_HP, A32 and M55_HE+M55_HP
    - Example runs a test program calling all available SERVICES API.
    - Output is sent via the SE-UART to save having to install extra UART debug ports.

Pre-requisites

The following components are required to be installed before using / building SE SERVICES:

- ALIF Ensemble RTSS Release >= Version 0.5.0
    - CMSIS Packs for Ensemble devices
    - Following the installation instructions for this package
- GNU Make V4.4
- Cmake V3.22.2
- Security Toolkit (SETOOLS)
    - Not required for building, but for generating the ATOC packages for the Target

Building SE Services – Windows / LINUX

Unpack the se-host-services-release-SE_FW_0.<version#>.000_DEV.zip

Name	Date modified	Type	Size
build	12/9/2021 9:44 AM	File folder	
example	12/9/2021 9:44 AM	File folder	
include	12/9/2021 9:44 AM	File folder	
lib	12/9/2021 9:54 AM	File folder	
services_lib	12/9/2021 9:44 AM	File folder	
Makefile	12/9/2021 9:44 AM	File	9 KB

The release archive consists of the following target components

Name	Purpose	Notes
build	Build objects	
example	Source code for A32, M55 examples	
include	Services header files	
lib	Pre-built libraries for A32, M55	
services_lib	Host source code for Services	
Makefile	Library and example builder	armclang
Makefile.gnu	Library and example builder	ARM GNU C

The top-level Makefile can build both the Host Services libraries and the examples.

The example directories have their own Makefiles which can be executed separately. They do rely on the Host libraries to be built.

Makefile Build Options

Option	Purpose	Notes
help	Show all supported targets	
all	Builds all the example directories	
lib	Build Host Services library	Uses CPU as target, default M55
example	Build example binary	Uses CPU as target
Install	Install the examples into SETOOLS	
Install-power	Install the power example	
Installdir	Install the example into <installdir>	default is ../app-release-exec
power	Build STOP mode test harness	REV_B only
clean	Removes build objects	Removes example binaries
realclean	Removes build objects and library	Removes example binaries
clean-power	Removed build objects for M55_power	
display	Shows target used	
display-env	Shows environment variables used	

Extra Notes:

lib	Builds only the SERVICES and MHU libaries
example	Builds the Example code based on the CPU parameter. Default is M55_HE.
Install	Builds the Example code based on the CPU parameter and copies the required files to your app-release-exec directory.
clean	Removes build objects for the CPU specified. Default is M55_HE.
realclean	Removes build objects for the CPU specified and the libraries. Default is M55_HE.

Makefile Command line options

Option	Values	Default	Notes
DEBUG	ON OFF	ON	Sets Debug option
DEVICE_REVISION	REV_A REV_A1 REV_B0 SPARK	REV_A1	SoC revision
CPU	M55_HE M55_HP A32	M55_HE	Application target
XIP	ON OFF	OFF	Enable XIP example builds

Makefile handling of CMSIS Packs

The examples require that ARM and ALIF CMSIS Packs are installed.

The se-host-service-release has a file called `toolchain_make.mak` which contains:

```
# Point at required Includes and your base ARM-DS / CMSIS installation
ALIF_CMSIS_VERSION        = 0.5.2
ARM_CMSIS_VERSION         = 5.9.0

# Point at required Includes and your base ARM-DS installation
ifneq ($(HOST_OS), LINUX_OS)
USER_BASE    := $(subst \,/,$(USERPROFILE))
INCLUDE_BASE := $(USER_BASE)/AppData/Local/Arm/Packs/ARM/CMSIS/5.9.0
else
USER_BASE    := /home/$(shell whoami)
INCLUDE_BASE := $(USER_BASE)/.cache/arm/packs/ARM/CMSIS/5.9.0
endif
```

Alternatively, you can override the defaults using:

```
$ make cmsis=5.9.0
$ make alifcmsis=0.5.0
```

Building with ARM Clang

```
$ cd se-host-services-release
$ make example
```
OR
```
$ make install
```
OR
```
$ make all
```

The SERVICE libraries are already pre-built and supplied with the release.

Building with ARM GNU C

```
$ cd se-host-services-release
$ make -f Makefile.gnu realclean
$ make -f Makefile.gnu lib
```
OR
```
$ make -f Makefile.gnu install
```
OR
```
$ make -f Makefile.gnu all
```

Option realclean is required as the default library supplied library builds are built using ARM Clang.

ARM GCC compiler used: `gcc version 11.3.1 20220712 (Arm GNU Toolchain 11.3.Rel1)`

Using CMake

```
$ cd se-host-services-release/build
$ cmake -G "Unix Makefiles" ..
$ make
```

This build will place the SERVICES and MHU libraries into the build directory and build the M55 HE, HP and Power examples (for REV_B0 release only).

The default compiler is ARM GNU CC

Some limitations:

- Does not support A32 baremetal build target.
- No option for arm-clang compiler
- M55_power example is for REV_B0 release only. However, CMakelists.txt still has this as a target – if you get errors on REV_A1 releases, comment out this line and rebuild.

Cmake Build Targets

```
$ make help
The following are some of the valid targets for this Makefile:
... all (the default if no target is provided)
... clean
... depend
... edit_cache
... install
... install/local
... install/strip
... list_install_components
... rebuild_cache
... m55_he_power_test.bin
... m55_he_services_test.bin
... m55_hp_services_test.bin
... m55_he_power_test.elf
... m55_he_services_test.elf
... m55_hp_services_test.elf
... mhu_m55_lib
... services_m55_lib

$ make m55_he_power_test.bin
```

Building the Host Service Library only

```
$ cd se-host-service-release

$ make CPU=M55_HE lib       Make M55 Library
$ make CPU=CPU=A32 lib      Make A32 Library
$ make all                  Make everything
```

Builds libservices_m55_lib. a and mhu library DEBUG = OFF. These are defaults.

Two libraries are built: services and MHU. The CPU name is appended to the output library name.

NOTE: By default, the libraries are pre-built using the ARM CLANG toolchain. If you are using ARM GNU C then you will need to rebuild these libraries as the formats are not compatible.

Building the Host Libraries and examples

```
$ cd se-host-service-release
$ make CPU=M55_HE
```
OR
```
$ make CPU=A32                Builds libservices_a32_lib. a, DEBUG = OFF
OR
$ make CPU-M55_HP
```

This will build the required Host Service Library that you can link with your application.
Note the M55 libraries does not depend on M55_HE or M55_HP build options.

SERVICES Library Dependencies

The SERVICES library and support file for starting SERVICES has a few dependencies.

- CMSIS Package installation
- Compiler flags

CMSIS Package

CMSIS dependency is mainly for header files for the targeted Application CPU e.g., “M55_HE.h”

CMSIS RTSS V0.5.0 Release

This RTSS release includes the MHU and SERVICES library in the CMSIS pack. This release also unified the Interrupt handler names for MHU M55_HE and HP.

SERVICES V23

- now supports this through its build system.
- The RTE sources previously supplied are no longer included and are expected to come from the installed RTSS release.
- No source code changes are required to any of the SERVICES examples.

There is ***no*** backwards compatibility with previous RTSS releases, we strongly recommend moving to RTSS V0.5.0 release.

This SERVICES release still contains the Source code for the SERVICE library and examples as these releases are usually ahead of the official RTSS CMSIS releases.

Compiler flags

There are a few compiler flags required, again mainly for the targeted Application CPU e.g. -DM55_HE

JSON Configurations

The examples can be built for the following Application cores:

- M55_HE
- M55_HP
- A32

Sample JSON files are provided for running in XIP and combinations of all Application cores are provided as well.

JSON File	Purpose
services-he-hp-a32.json	Boot all Apps cores
services-he-hp-a32-b0.json	Boot all Apps cores for REV_B0
services-he-hp-a32-xip.json	Boot all Apps cores, run XIP
services-he.json	Boot M55-HE
services-he-b0.json	Boot M55-HE for REV_B0
services-he-xip.json	Boot M55_HE, run XIP
services-hp.json	Boot M55-HE
services-he-xip.json	Boot M55_HE, run XIP
services-he-hp.json	Boot M55_HE and M55_HP
services-he-hp-b0.json	Boot M55_HE and M55_HP for REV_B0
services-hp-xip.json	Boot M55_HP, run XIP
services-a32.json	Boot A32
services-a32-xip.json	Boot A32, run XIP

Examples

Examples build matrix

SERVICES library and example can be built with various options:

- Compiler
  - ARM Clang (covers all combinations)
  - ARM GNU C.
    - A32 is not supported.
- CPU
  - M55_HE, HP, A32 or combinations of both or all
- XIP
  - ON or OFF
- Power example
  - REV_B0 only.
- CMake
  - Only GCC is supported for M55_HE and HP

When building with ARM Clang and switching to GCC (or vice versa) you may see some linker warnings. If these are seen run:

```
$ make realclean
```

Then re-run your previous Make command.

The libraries are supplied pre-built using ARM Clang and so are not compatible with GCC. We may chose to remove the pre-built libraries in future releases due to this incompatibility.

HE	HP	A32	Method	Compiler	XIP=ON	XIP=OFF	Power	REV_B0	REV_A1
●	●	●	Make all	GNU	✓	✓	×	×	✓
●	●	●	Make all	Clang	✓	✓	×	×	✓
●			Make	Clang	✓	✓	✓	✓	×
	●		Make	Clang	✓	✓	✓	✓	×
		●	Make	Clang	✓	✓	✓	×	×
●	●		Make	Clang	✓	✓	✓	×	×
●	●	●	Make	Clang	✓	✓	✓	×	×
●			Make	GNU C	✓	✓	×	✓	✓
	●		Make	GNU C	✓	✓	×	✓	✓
		●	Make	GNU C	×	×	×	✓	✓
●	●		Make	GNU C	✓	✓	×	✓	✓
●	●	●	Make	GNU C	×	×	×	✓	✓
●			CMAKE	GNU C	×	✓	✓	✓	✓
	●		CMAKE	GNU C	×	✓	×	✓	✓
		●	CMAKE	GNU C	×	×	×	×	✓
●	●		CMAKE	GNU C	×	×	×	×	✓
●	●	●	CMAKE	GNU C	×	×	×	×	✓
●			CMAKE	Clang	×	×	×	×	×
	●		CMAKE	Clang	×	×	×	×	×
		●	CMAKE	Clang	×	×	×	×	×
●	●		CMAKE	Clang	×	×	×	×	×
●	●	●	CMAKE	Clang	×	×	×	×	×

NOTE:

- GNU C / GCC
  - A32 is not supported due to a lack of a suitable bare metal linker script file.
- CMAKE
  - Only GCC XIP=OFF, M55_HE is supported currently

Examples build options

Output of the results from the example test can be via the ARM-DS Console or the SE-UART.

In `services_test.c` there are the following defines

```
#define TEST_PRINT_ENABLE           1   /* Enable printing from Test harness */
#define PRINT_VIA_CONSOLE           0   /* Print via Debugger console        */
#define PRINT_VIA_SE_UART           1   /* Print via SE UART terminal        */
```

Flag	Meaning
TEST_PRINT_ENABLE	Turn on output from the test
PRINT_VIA_CONSOLE	Print messages to arm-ds (printf())
PRINT_VIA_SE_UART	Print messages to the SE-UART

You can enable both Console and SE-UART.

If you want to run the test from MRAM the PRINT_VIA_CONSOLE must be disabled.

Building the M55 HE Example and run from the ARM-DS debugger

```
$ cd <host-release directory>
$ make example
```
OR
```
$ make all
```

This will compile the sample example for M55 HE processor. The sample example is a test harness for (most) of the SERVICE calls available.

The sample contains the library interfaces required by Services. It then calls each of the available Services.

The example executable is stored in the build directory of the example. You will need to point your Debugger to this file and run from the debugger e.g. ARM-DS.

Building the M55 HE Example and run from the SEGGER Ozone debugger.

```
$ cd <host-release directory>
$ make -f Makefile.gnu example
```
OR
```
$ make -f Makefile.gnu
```

This will compile the sample example for M55 HE processor. The sample example is a test harness for (most) of the SERVICE calls available.

The sample contains the library interfaces required by Services. It then calls each of the available Services.

The example executable is stored in the build directory of the example. You will need to point your Debugger to this file and run from the debugger e.g. ARM-DS.

Building the M55 HE Example run from MRAM

There are two json files supplied in the Services release:

- services-he-.json
  - Single Core
- services-he-hp.json
  - Dual Core

Follow the instructions to build the M55_HE or HP example.

```
$ cd <host-release directory>
$ make example DEVICE_REVISION=REV_A1 CPU=M55_HE

EITHER
$ cp example/m55_he/services-he.json ../app-release-exec/build/config
$ cp example/m55_he/build/m55_he_services_test.bin ../app-
release/build/images

OR
$ cd <host-release directory>
$ make install installdir=<app-release directory> DEVICE_REVISION=REV_A1
CPU=M55_HE

$ cd ../app-release-exe
$ app-gen-toc -f build/config/services-he.json
$ app-write-mram
```

To boot both M55 application CPUs you need these steps:

```
$ make example DEVICE_REVISION=REV_A1 CPU=M55_HE
$ cp example/m55_he/build/m55_he_services_test.bin ../app-release-
exec/build/images
$ make clean
$ make example DEVICE_REVISION=REV_A1 CPU=M55_HP
$ cp example/m55_he/build/m55_hp_services_test.bin ../app-release-
exec/build/images
$ cp example/m55_hp/services-he-hp.json ../app-release-exec/build/config
$ cd ../app-release-exec
$ app-gen-toc -f build/config/services-he.json
$ app-write-mram
```

Building the M55 Host XIP Example from MRAM

```
$ cd <host-release directory>
$ make example DEVICE_REVISION=REV_A1 CPU=M55_HE XIP=ON
$ cp example/m55_he/services-he-xip.json ../app-release/build/config
$ cp example/m55_he/build/m55_he_services_test_xip.bin ../app-
release/build/images
$ cd ../app-release
$ app-gen-toc -f build/config/services-he-xip.json
$ app-write-mram
```

Also write the XIP images to MRAM address specified in the JSON file configuration used i.e. services-he-xip.json.

NOTE: To save doing the copies you can use the install option.

```
$ cd <host-release directory>
$ make install DEVICE_REVISION=REV_A1 CPU=M55_HE XIP=ON
```

Building the A32 Host Example

```
$ cd <host-release directory>
$ make example DEVICE_REVISION=REV_A1 CPU=A32
OR
$ cd example/a32_bare_metal
$ make
```

This will compile the sample example for A32 application processor.

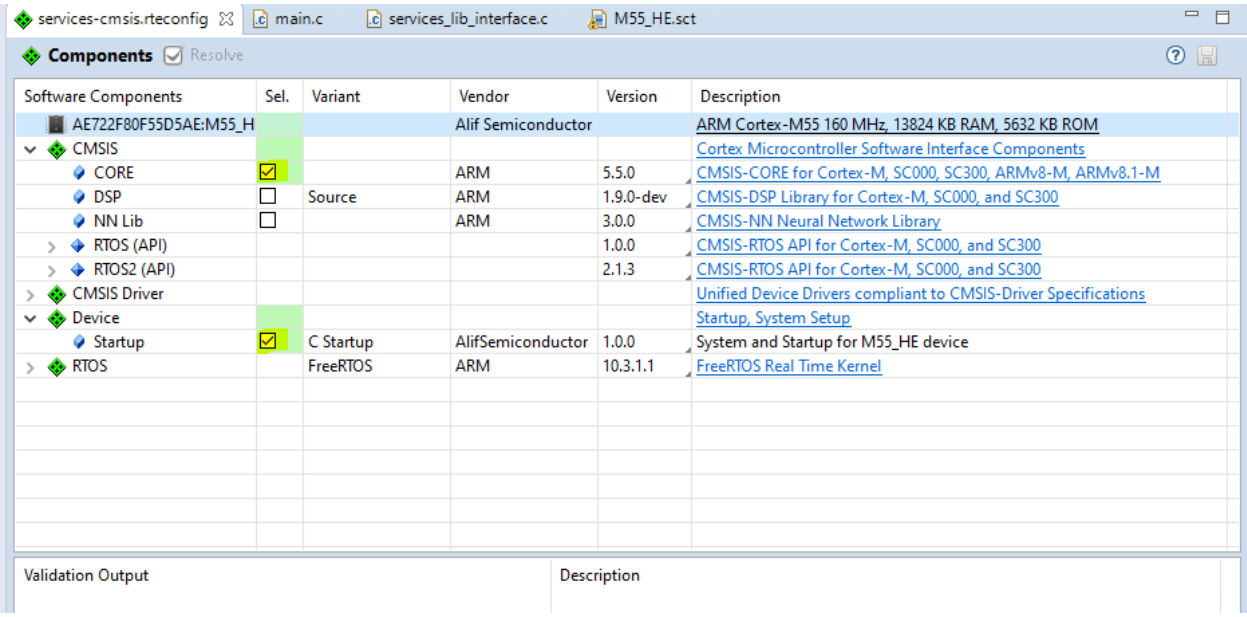
Building the A32 Host XIP Example

```
$ cd <host-release directory>
$ make example DEVICE_REVISION=REV_A1 CPU=A32 XIP=ON
```

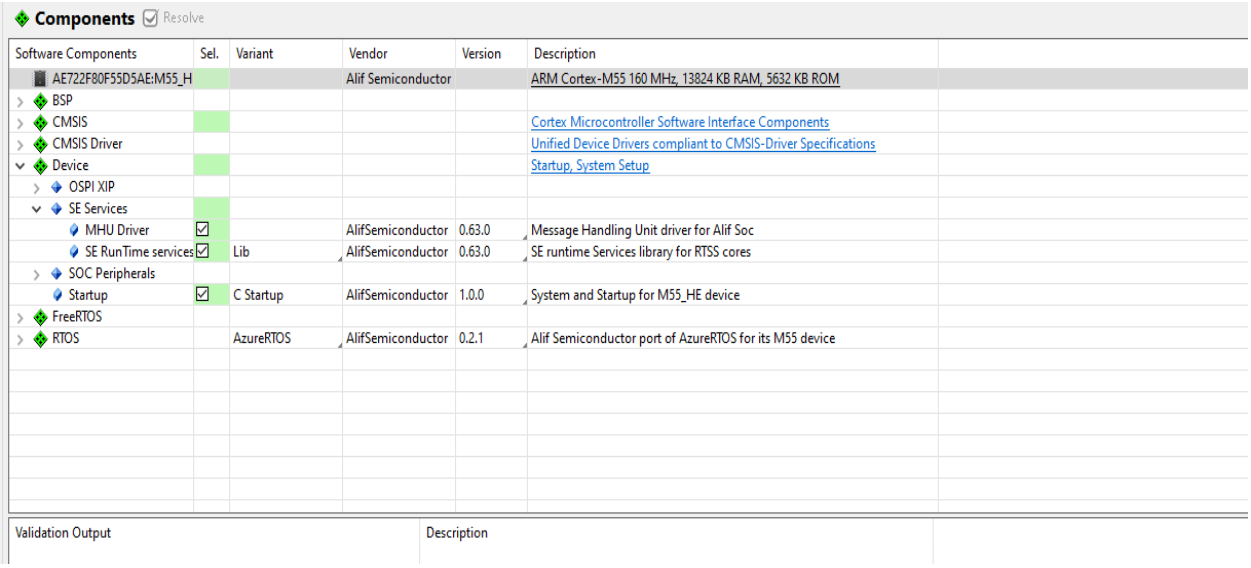
Building the M55 Host Example under ARM-DS

Before starting, ensure you have the ALIFSemiconductor CMSIS Pack installed (See AP002 Getting Started with Bare Metal & Azure RTOS)

- Create a new Project -> C Project -> CMSIS C/C++ Project
- Select Device -> Alif Semiconductor -> Ensemble -> E7 -> AE722F80F55D5AE (or the other one) -> AE722F80F55AE:M55_HE
- In the Project Components window,
    - Check the following highlighted boxes,
    - Then File -> Save



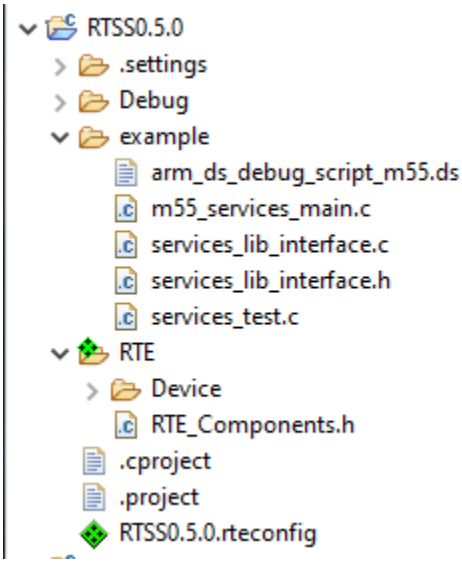
Select the SE Services



Creating a SERVICES based project in ARM-DS

- Copy the source files from your unpacked services release into your project or from the CMSIS Pack (RTSS V0.5.0 or above)
- Copy the following directories / files from the SE Services release over to the ARM-DS project.
  - `./example/m55_he`
  - `./example/common`
  - **NOTE:** Do *not copy* the A32 dir

In the below example, a New Project (RTSS0.5.0) has been created in ARM-DS:



You can copy the services library and MHU driver source as well.

The example source has been included as this contains the main() entry point.

Building M55_HE Power Example

NOTE: This is only available with the REV_B0 release. See build matrix for all options

```
$ make power DEVICE_REVISION=REV_B0
OR
$ make -f Makefile.gnu power DEVICE_REVISION=REV_B0
```

Building SE Host Services – LINUX

Unpack the se-host-services-release-SE_FW_0.<version#>.000_DEV.zip

There is a separate makefile file for building the Services library for Linux - 'Makefile_linux', so that file should be used instead of the default 'Makefile' –

```
$ make -f Makefile_linux lib
```

By default, things are set up to use the native GCC compiler in Cygwin.

To use the Alif Yocto cross compiler toolchain and generate binaries for the Alif Linux distribution, a couple of changes are needed.

- comment out the compiler definitions (like 'CC = gcc') in Makefile_linux. The Yocto toolchain provides its own definitions.
- modify the file services_lib\services_host_handler_linux.c and replace '#if 0' with '#if 1', to include the Linux kernel header file for the MHU driver.

Installing examples

The examples come with supplied JSON files for A32, M55_HE and M55_HP processors, also are variants for XIP running.

There is an option to install these examples into your Application Release to enable building an ATOC for putting into MRAM. To build the ATOC you need the JSON file and the binary image for Application. These files are copied from the se-host-services-release into your application release

```
$ cd se-host-services-release
$ make install installdir=<path to your application release>
```

Note the use of the `installdir` to specify where your application release lives.

When you unpack your application release you will get a directory structure as follows:

```
app-release-exec-windows-SE-FW_0.<version>
     + app-release-exec
          + build
               + config
               + images
```

The JSON files will be copied to the config directory and the binaries will be copied to images. This is where the ATOC generation tools will look.

An example of using the installdir is as follows:

```
$ make install installdir=app-release-exec-windows-SE-
FW_0.<version>/app-release-exec
```

If you do not specify the `installdir` then the default is `../app-release-exec`

Installing REV_B0 Power examples

```
$ cd se-host-services-release
$ make install-power installdir=<path to your application release>
```

Running Services

Running the A32 Host Example

Build the example:

```
$ cd se-host-service-release
$ make install CPU=A32 -j 8
```

We have used the install option to copy the binaries and configurations into the tools release directory.

```
$ cd ../app-release-exec/
$ app-gen-toc.exe -f build/config/services-a32.json
$ app-write-mram.exe
```

Using the maintenance tool, we can see the A32 is booted.

```
Available options:

1 - Get TOC info
2 - Get SES Banner
3 - Get CPU boot info
4 - Device enquiry
5 - Get revision info

Select an option (Enter to return): 1
 +----------+--------+------------+------------+------------+------------+----------+-----------+--------+----------+
 |   Name   |  CPU   | Store Addr |  Obj Addr  | Dest Addr  | Boot Addr  |   Size   |  Version  |  Flags | Time (ms)|
 +----------+--------+------------+------------+------------+------------+----------+-----------+--------+----------+
 |   DEVICE |  CM0+  | 0x805C1EE0 | 0x805C14E0 | ---------- | ---------- |      672 |      0.5.0| u V    |     9.38 |
 |   SERAM0 |  CM0+  | ---------- | 0x000000E0 | ---------- | ---------- |    66752 |      1.0.0| ------ |     0.00 |
 |   SERAM1 |  CM0+  | ---------- | 0x00020AE0 | ---------- | ---------- |    66752 |      1.0.0| ------ |     0.00 |
 |   HE_DBG | M55-HE | 0x805C2B80 | 0x805C2180 | 0x60000000 | 0x60000000 |     2256 |      1.0.0| uLVB   |     0.00 |
 |  SRV-A32 | A32_0  | 0x80579CE0 | 0x805792E0 | 0x02000000 | 0x02000000 |    25300 |      1.0.0| uLVB   |    12.68 |
 +----------+--------+------------+------------+------------+------------+----------+-----------+--------+----------+

 Legend: (u)(C)ompressed,(L)oaded,(V)erified,(s)kipped verification,(B)ooted,(E)ncrypted,(D)eferred
```

Running the M55 HE Host Example with ARM-DS

Firstly, follow the build instructions for the M55 HE Host example.

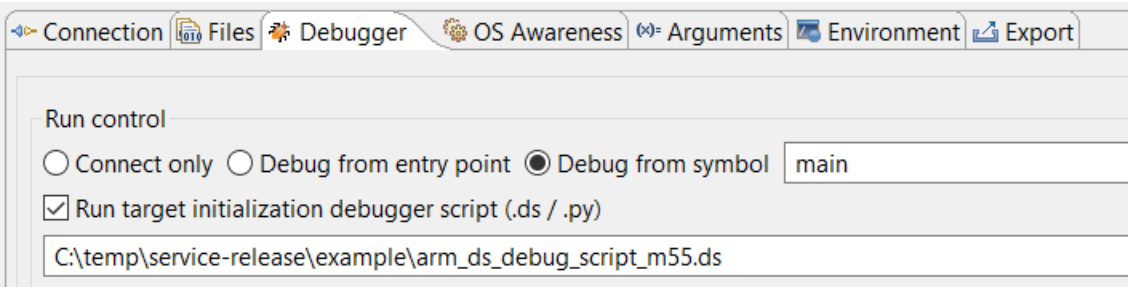
Launch the ARM-DS debugger and set the following:

Debug Configurations -> Files to
`./se-host-service-release\example\m55_bare_metal\build\m55_bare_metal.axf`

Debug Configurations -> Debugger

The Services release includes an ARM-DS debugger script named arm_ds_debug_script_m55.ds, that initializes the M55 core before running anything on it via Arm DS. The script is in folder services-release/example.

Initializing the M55 prevents various issues, so it is recommended to configure the debug connection to execute the script, as shown on the screenshot -



The script lives in./se-host-service-release/example

With the debugger launched, this will bring up the following breakpoint at main():

```
extern void SERVICES_test(uint32_t services_handle);

int main(void)
{
  SERVICES_print("[SE SERVICES] %s Test harness - STARTS\n", CPU_STRING);

  /**
   * Initialise the MHU and SERVICES Library
   */
  mhu_initialize();
  SERVICES_Setup(s_mhu_driver_out.send_message);

  SERVICES_wait_ms(0x1000000);

  uint32_t services_handle = SERVICES_register_channel(MHU_M55_SE_MHU0, 0);

#if CONTINUOUS_RUN == 1
  for (;;)
#else
  for (int test_run=0; test_run < LIMITED_RUN; test_run++)
#endif
  {
    SERVICES_test(services_handle);
  }

  SERVICES_print("[SE SERVICES] Test harness ENDS\n");

  while(1);

  return 0;
}
```

We can step through the code until SERVICES_test().

Stepping over this function will give us the following output:

```
SERVICES version 0.0.5
** TEST heartbeat       error_code=SERVICES_REQ_SUCCESS
** TEST pinmux          error_code=SERVICES_REQ_SUCCESS         service_resp=0x00000000
** TEST padcontrol      error_code=SERVICES_REQ_SUCCESS         service_resp=0x00000000
** TEST crypto TRNG 64  error_code=SERVICES_REQ_SUCCESS         64-bit Random value = 0x443bf4c19f1d3ad8 service_resp=0
** TEST crypto TRNG 32  error_code=SERVICES_REQ_SUCCESS         32-bit Random value = 0xa84fb766 service_resp=0
** TEST crypto TRNG 64  error_code=SERVICES_REQ_SUCCESS         64-bit Random value = 0xdc23c000f7dc58b3 service_resp=0
** TEST crypto TRNG 32  error_code=SERVICES_REQ_SUCCESS         32-bit Random value = 0x6f151dc4 service_resp=0
** TEST crypto LCS      error_code=SERVICES_REQ_SUCCESS         LCS State 0x0 service_resp=0
** TEST get ATOC        error_code=SERVICES_REQ_SUCCESS         Application TOC number = 1 service_resp=0x00000000
** TEST TOC via name HE error_code=SERVICES_REQ_SUCCESS         service_resp=0x00000000
** TEST TOC via name HP error_code=SERVICES_REQ_SUCCESS         service_resp=0x00000000
** TEST TOC via cpuid   error_code=SERVICES_REQ_SUCCESS         id HE_DBG flags 63 1 0 0 service_resp=0x00000000
** TEST TOC via cpuid   error_code=SERVICES_REQ_SUCCESS         id HE_DBG flags 63 1 0 0 service_resp=0x00000000
** TEST soc id          error_code=SERVICES_REQ_SUCCESS         Device number 0xA100 service_resp=0x00000000
** TEST Boot TOC A32    error_code=SERVICES_REQ_SUCCESS         service_resp=0x00000000
```

The Application Console print outs show the call to the SERVICES library, if you have SE-UART connected you will see the following output:

```
COM9 - REV_A1 VT
File  Edit  Setup  Control  Window  Help
[SES] System       partition processed (0x00000000) BL_STATUS_OK
[SES] Application partition processed (0x00000000) BL_STATUS_OK

[SES] FC:Rgn - 7:1 7:2 7:3 7:4 7:5 8:1 8:2 8:3 8:4 8:5 13:0 13:1 13:2
[SES] Protected areas:
        0x80580000 - 0x805FFFFF

+----------+--------+-----------+-----------+-----------+-----------+----------+-------+--------+----------+
|   Name   |  CPU   | Store Addr | Obj Addr  | Dest Addr | Boot Addr |   Size   |Version| Flags  | Time (ms)|
+----------+--------+-----------+-----------+-----------+-----------+----------+-------+--------+----------+
|  SERAM0  |  CM0+  | ---------- | 0x000000E0 | ---------- | ---------- |   57024  | 1.0.0 | u s    |     0.00 |
|  SERAM1  |  CM0+  | ---------- | 0x00020AE0 | ---------- | ---------- |   57024  | 1.0.0 | u s    |     0.00 |
|  DEVICE  |  CM0+  | 0x805C1EE0 | 0x805C14E0 | ---------- | ---------- |     680  | 0.5.0 | ------ |     9.42 |
|  HE_DBG  | M55-HE | ---------- | 0x805C2190 | ---------- | ---------- |    2256  | 1.0.0 | uLs    |     0.00 |
| BLINK-HE | M55-HE | 0x8057EC90 | 0x8057E290 | 0x60000000 | 0x90000000 |    4912  | 1.0.0 | uLVB   |     9.46 |
+----------+--------+-----------+-----------+-----------+-----------+----------+-------+--------+----------+
Legend: (u)(C)ompressed,(L)oaded,(V)erified,(s)kipped verification,(B)ooted,(E)ncrypted,(D)eferred


[SES] CM0+ frequency is 100 MHz
[SES] os Kernel V10.4.2
[SES] Main Task - looping forever...
[SRV] RX<-- SID= 0x068, Receiver ID=4, Address=0x9083FD88
[TTY] SERVICES version 0.0.5
[SRV] TX--> SID= 0x068, Receiver ID=4, Address=0x9083FD88
[SRV] RX<-- SID= 0x000, Receiver ID=4, Address=0x9083FF90
[SRV] TX--> SID= 0x000, Receiver ID=4, Address=0x9083FF90
[SRV] RX<-- SID= 0x068, Receiver ID=4, Address=0x9083FD68
[TTY] ** TEST heartbeat       error_code=SERVICES_REQ_SUCCESS
[SRV] TX--> SID= 0x068, Receiver ID=4, Address=0x9083FD68
[SRV] RX<-- SID= 0x065, Receiver ID=4, Address=0x9083FF84
[SRV] TX--> SID= 0x065, Receiver ID=4, Address=0x9083FF84
[SRV] RX<-- SID= 0x068, Receiver ID=4, Address=0x9083FD60
[TTY] ** TEST pinmux          error_code=SERVICES_REQ_SUCCESS          service_resp=0x00000000
[SRV] TX--> SID= 0x068, Receiver ID=4, Address=0x9083FD60
[SRV] RX<-- SID= 0x066, Receiver ID=4, Address=0x9083FF84
[SRV] TX--> SID= 0x066, Receiver ID=4, Address=0x9083FF84
[SRV] RX<-- SID= 0x068, Receiver ID=4, Address=0x9083FD60
[TTY] ** TEST padcontrol      error_code=SERVICES_REQ_SUCCESS          service_resp=0x00000000
[SRV] TX--> SID= 0x068, Receiver ID=4, Address=0x9083FD60
[SRV] RX<-- SID= 0x190, Receiver ID=4, Address=0x9083FB68
[SRV] TX--> SID= 0x190, Receiver ID=4, Address=0x9083FB68
[SRV] RX<-- SID= 0x068, Receiver ID=4, Address=0x9083FD50
[TTY] ** TEST crypto TRNG 64  error_code=SERVICES_REQ_SUCCESS          64-bit Random value = 0x443bf4c19f1d3ad8 service_
```

The [SRV] prints from SES are showing that a SERVICE request (RX) with Service ID (SID) has been sent and SES has sent a response (TX) back to the requestor.

The [TTY] prints from SES are SERVICE prints from the Application CPU that are being printed using the SE-UART.

Running the M55 HE Host Example from MRAM

Build the example:

```
$ cd se-host-service-release
$ make install -j 8
```

We have used the install option to copy the binaries and configurations into the tools release directory.

```
$ cd ../app-release-exec/
$ app-gen-toc.exe -f build/config/services-he.json
$ app-write-mram.exe
```

Using the Maintenance tool:

```
Available options:

1 - Get TOC info
2 - Get SES Banner
3 - Get CPU boot info
4 - Device enquiry
5 - Get revision info

Select an option (Enter to return): 1
 +----------+--------+------------+------------+------------+------------+----------+-----------+--------+----------+
 |   Name   |  CPU   | Store Addr |  Obj Addr  | Dest Addr  | Boot Addr  |   Size   |  Version  |  Flags | Time (ms)|
 +----------+--------+------------+------------+------------+------------+----------+-----------+--------+----------+
 |   DEVICE |  CM0+  | 0x805C1EE0 | 0x805C14E0 | ---------- | ---------- |      672 |     0.5.0| u V    |     9.38 |
 |   SERAM0 |  CM0+  | ---------- | 0x000000E0 | ---------- | ---------- |    66752 |     1.0.0| ------ |     0.00 |
 |   SERAM1 |  CM0+  | ---------- | 0x00020AE0 | ---------- | ---------- |    66752 |     1.0.0| ------ |     0.00 |
 |   HE_DBG | M55-HE | ---------- | 0x805C2180 | ---------- | ---------- |     2256 |     1.0.0| uLs    |     0.00 |
 | SRV-HE-T | M55-HE | 0x8057B450 | 0x8057B0E0 | 0x60000000 | 0x60000000 |    19308 |     1.0.0| uLsB   |     3.09 |
 +----------+--------+------------+------------+------------+------------+----------+-----------+--------+----------+

 Legend: (u)(C)ompressed,(L)oaded,(V)erified,(s)kipped verification,(B)ooted,(E)ncrypted,(D)eferred
```

The SRV-HE-T test program is loaded from MRAM.

In the maintenance tool, chose the Terminal Mode and press [RESET] on your board

```
[SES] CM0+ frequency is 100 MHz
[SES] os Kernel V10.4.2
[SES] Main Task - looping forever...
[SRV] RX<-- SID= 0x0D0, Receiver ID=4, Address=0x908001F4
[TTY] [M55_HE] SERVICES version 0.0.14
[TTY] ** TEST heartbeat       error_code=SERVICES_REQ_SUCCESS
[TTY] ** TEST pinmux          error_code=SERVICES_REQ_SUCCESS                service_resp=0x00000000
[TTY] ** TEST padcontrol      error_code=SERVICES_REQ_SUCCESS                service_resp=0x00000000
[TTY] ** TEST crypto TRNG 64  error_code=SERVICES_REQ_SUCCESS                64-bit Random value = 0xbd9489f16101a45c service_resp=0
[TTY] ** TEST crypto TRNG 32  error_code=SERVICES_REQ_SUCCESS                32-bit Random value = 0x99f53cb2 service_resp=0
[TTY] ** TEST crypto TRNG 64  error_code=SERVICES_REQ_SUCCESS                64-bit Random value = 0xec8e9364a463f6a6 service_resp=0
[TTY] ** TEST crypto TRNG 32  error_code=SERVICES_REQ_SUCCESS                32-bit Random value = 0x4bba2d49 service_resp=0
[TTY] ** TEST crypto get LCS  error_code=SERVICES_REQ_SUCCESS                LCS State 0x0 service_resp=0
[TTY] ** TEST OTP get data    error_code=SERVICES_REQ_COMMAND_NOT_IMPLEMENTED service_resp=0x000000F3
[TTY] ** TEST TOC get data    error_code=SERVICES_REQ_SUCCESS                Application TOC number = 1 service_resp=0x00000000
[TTY] ** TEST TOC via name HE error_code=SERVICES_REQ_SUCCESS                service_resp=0x00000000
[TTY] ** TEST TOC via name HP error_code=SERVICES_REQ_SUCCESS                service_resp=0x00000000
[TTY] ** TEST TOC via cpuid   error_code=SERVICES_REQ_SUCCESS                id HE_DBG flags 63 1 0 0 service_resp=0x00000000
[TTY] ** TEST TOC via cpuid   error_code=SERVICES_REQ_SUCCESS                id HE_DBG flags 63 1 0 0 service_resp=0x00000000
[TTY] ** TEST TOC version     error_code=SERVICES_REQ_SUCCESS                service_resp=0x000000F3 version=0
[TTY] ** TEST TOC get data    error_code=SERVICES_REQ_SUCCESS                TOC number = 5 service_resp=0x00000000
[TTY] |   Name   | CPU |Load Address|Boot Address|Image Size|Version|    Flags   |
[TTY] +-------------------------------------------------------------------------+
[TTY] |   DEVICE |  0  | 0x00000000 | 0x00000000 |      672 | 0.5.0 | 0x00000000 |
[TTY] |   SERAM0 |  0  | 0x00000000 | 0x00000000 |    66752 | 1.0.0 | 0x00000000 |
[TTY] |   SERAM1 |  0  | 0x00000000 | 0x00000000 |    66752 | 1.0.0 | 0x00000000 |
[TTY] |   HE_DBG |  3  | 0x00000000 | 0x00000000 |     2256 | 1.0.0 | 0x00000063 |
[TTY] | SRV-HE-T |  3  | 0x60000000 | 0x60000000 |    19308 | 1.0.0 | 0x00000063 |
[TTY] ** TEST get SE revision error_code=SERVICES_REQ_SUCCESS                SES A1 EVALUATION_BOARD SE_FW_0.57.000_DEV v0.57.0 Sep  3 2022 0
:14:24 service_resp=0x00000000
[TTY] ** TEST get soc id      error_code=SERVICES_REQ_SUCCESS                Device number 0xA100 service_resp=0x00000000
[TTY] ** TEST read otp        error_code=SERVICES_REQ_INVALID_OTP_OFFSET     service_resp=SERVICES_REQ_INVALID_OTP_OFFSET
[TTY] ** OTP offset 0x59 OTP Value = 0x00000000
[TTY] ** OTP offset 0x5a OTP Value = 0x00000000
[TTY] ** TEST Boot TOC A32    error_code=SERVICES_REQ_SUCCESS                service_resp=0x00000000
[TTY] ** TEST MbedTLS AES     error_code=SERVICES_REQ_SUCCESS                service_resp=0x00000000
[TTY] ** TEST Bounds Tests    error_code=SERVICES_REQ_BAD_PRINT_LENGTH       service_resp=0x00000000
[TTY] ** TEST Bounds Tests    error_code=SERVICES_REQ_BAD_PRINT_LENGTH       service_resp=0x00000000
[TTY] ** TEST Bounds Tests    error_code=SERVICES_REQ_NULL_PARAMETER         service_resp=0x00000000

[SERVICES] Stop mode request not implemented
[TTY] ** TEST EWIC config     error_code=SERVICES_REQ_COMMAND_NOT_IMPLEMENTED Device number 0x460B service_resp=0x00000000

[SERVICES] Stop mode request not implemented
[TTY] ** TEST VBAT Wake Up config error_code=SERVICES_REQ_COMMAND_NOT_IMPLEMENTED Device number 0x46F2 service_resp=0x00000000
```

You can now see the Services calls coming through SES as well as SERVICES prints coming from the application code being printed on the SE-UART.

Running with Debug disabled

The test harness has a call to SERVICES_system_set_services_debug() which can disable or enable the debug traffic from SES for the SERVICE traffic.

With the SERVICES debug set to false:

```
[SES] CM0+ frequency is 100 MHz
[SES] os Kernel V10.4.2
[SES] Main Task - looping forever...
[SRV] RX<-- SID= 0x0CE, Receiver ID=4, Address=0x9083FFA8
[TTY] SERVICES version 0.0.6
[TTY] ** TEST heartbeat       error_code=SERVICES_REQ_SUCCESS
[TTY] ** TEST pinmux          error_code=SERVICES_REQ_SUCCESS        service_resp=0x00000000
[TTY] ** TEST padcontrol      error_code=SERVICES_REQ_SUCCESS        service_resp=0x00000000
[TTY] ** TEST crypto TRNG 64  error_code=SERVICES_REQ_SUCCESS        64-bit Random value = 0x1cfed9edc1501c20 service_resp=0
[TTY] ** TEST crypto TRNG 32  error_code=SERVICES_REQ_SUCCESS        32-bit Random value = 0xa479c88f service_resp=0
[TTY] ** TEST crypto TRNG 64  error_code=SERVICES_REQ_SUCCESS        64-bit Random value = 0xbc3fcf15f8479420 service_resp=0
[TTY] ** TEST crypto TRNG 32  error_code=SERVICES_REQ_SUCCESS        32-bit Random value = 0x5cfbb90c service_resp=0
[TTY] ** TEST crypto LCS      error_code=SERVICES_REQ_SUCCESS        LCS State 0x0 service_resp=0
[TTY] ** TEST get ATOC        error_code=SERVICES_REQ_SUCCESS        Application TOC number = 0 service_resp=0x00000000
[TTY] ** TEST TOC via name HE error_code=SERVICES_REQ_SUCCESS        service_resp=0x00000000
[TTY] ** TEST TOC via name HP error_code=SERVICES_REQ_SUCCESS        service_resp=0x00000000
[TTY] ** TEST TOC via cpuid   error_code=SERVICES_REQ_SUCCESS        id HE_DBG flags 63 1 0 0 service_resp=0x00000000
[TTY] ** TEST TOC via cpuid   error_code=SERVICES_REQ_SUCCESS        id HE_DBG flags 63 1 0 0 service_resp=0x00000000
[TTY] ** TEST soc id          error_code=SERVICES_REQ_SUCCESS        Device number 0xA100 service_resp=0x00000000
[TTY] ** TEST Boot TOC A32    error_code=SERVICES_REQ_SUCCESS        service_resp=0x00000000
```

The SERVICE call to set the debug output off can be seen (the default is enabled in SES). After that, there is no SERVICE debug traffic from SES.

Running the M55 HP Host Example from MRAM

Build the example:

```
$ cd se-host-service-release
$ make install CPU=M55_HP -j 8
```

We have used the install option to copy the binaries and configurations into the tools release directory.

```
$ cd ../app-release-exec/
$ app-gen-toc.exe -f build/config/services-hp.json
$ app-write-mram.exe
```

Using the Maintenance tool:

```
Available options:

1 - Get TOC info
2 - Get SES Banner
3 - Get CPU boot info
4 - Device enquiry
5 - Get revision info

Select an option (Enter to return): 1
 +----------+--------+------------+------------+------------+------------+----------+-----------+--------+----------+
 |   Name   |  CPU   | Store Addr |  Obj Addr  | Dest Addr  | Boot Addr  |   Size   |  Version  |  Flags | Time (ms)|
 +----------+--------+------------+------------+------------+------------+----------+-----------+--------+----------+
 |   DEVICE |  CM0+  | 0x805C1EE0 | 0x805C14E0 | ---------- | ---------- |      672 |      0.5.0| u V    |     9.39 |
 |   SERAM0 |  CM0+  | ---------- | 0x000000E0 | ---------- | ---------- |    66752 |      1.0.0| ------ |     0.00 |
 |   SERAM1 |  CM0+  | ---------- | 0x00020AE0 | ---------- | ---------- |    66752 |      1.0.0| ------ |     0.00 |
 |   HE_DBG | M55-HE | 0x805C2B80 | 0x805C2180 | 0x60000000 | 0x60000000 |     2256 |      1.0.0| uLVB   |     0.00 |
 | SRV-HP-T | M55-HP | 0x8057B890 | 0x8057AE90 | 0x50000000 | 0x50000000 |    18216 |      1.0.0| uLVB   |    11.63 |
 +----------+--------+------------+------------+------------+------------+----------+-----------+--------+----------+

 Legend: (u)(C)ompressed,(L)oaded,(V)erified,(s)kipped verification,(B)ooted,(E)ncrypted,(D)eferred
```

The SRV-HP-T test program is loaded from MRAM.

In the maintenance tool, chose the Terminal Mode and press [RESET] on your board.

```
[SES] CM0+ frequency is 100 MHz
[SES] os Kernel V10.4.2
[SES] Main Task - looping forever...
[SRV] RX<-- SID= 0x0D0, Receiver ID=2, Address=0x888001F4
[TTY] [M55_HP] SERVICES version 0.0.14
[TTY] ** TEST heartbeat       error_code=SERVICES_REQ_SUCCESS
[TTY] ** TEST pinmux          error_code=SERVICES_REQ_SUCCESS                service_resp=0x00000000
[TTY] ** TEST padcontrol      error_code=SERVICES_REQ_SUCCESS                service_resp=0x00000000
[TTY] ** TEST crypto TRNG 64  error_code=SERVICES_REQ_SUCCESS                64-bit Random value = 0x924e04901e86d3be service_resp=0
[TTY] ** TEST crypto TRNG 32  error_code=SERVICES_REQ_SUCCESS                32-bit Random value = 0x20605b14 service_resp=0
[TTY] ** TEST crypto TRNG 64  error_code=SERVICES_REQ_SUCCESS                64-bit Random value = 0x320e5068883aac83 service_resp=0
[TTY] ** TEST crypto TRNG 32  error_code=SERVICES_REQ_SUCCESS                32-bit Random value = 0x7e74f500 service_resp=0
[TTY] ** TEST crypto get LCS  error_code=SERVICES_REQ_SUCCESS                LCS State 0x0 service_resp=0
[TTY] ** TEST OTP get data    error_code=SERVICES_REQ_COMMAND_NOT_IMPLEMENTED service_resp=0x000000F3
[TTY] ** TEST TOC get data    error_code=SERVICES_REQ_SUCCESS                Application TOC number = 1 service_resp=0x00000000
[TTY] ** TEST TOC via name HE error_code=SERVICES_REQ_SUCCESS                service_resp=0x00000000
[TTY] ** TEST TOC via name HP error_code=SERVICES_REQ_SUCCESS                service_resp=0x00000000
[TTY] ** TEST TOC via cpuid   error_code=SERVICES_REQ_SUCCESS                id HE_DBG flags 63 1 0 0 service_resp=0x00000000
[TTY] ** TEST TOC via cpuid   error_code=SERVICES_REQ_SUCCESS                id HE_DBG flags 63 1 0 0 service_resp=0x00000000
[TTY] ** TEST TOC version     error_code=SERVICES_REQ_SUCCESS                service_resp=0x000000F3 version=0
[TTY] ** TEST TOC get data    error_code=SERVICES_REQ_SUCCESS                TOC number = 5 service_resp=0x00000000
[TTY] |   Name   | CPU |Load Address|Boot Address|Image Size|Version|    Flags   |
[TTY] +-------------------------------------------------------------------------+
[TTY] |   DEVICE |  0  | 0x00000000 | 0x00000000 |      672 | 0.5.0 | 0x00000000 |
[TTY] |   SERAM0 |  0  | 0x00000000 | 0x00000000 |    66752 | 1.0.0 | 0x00000000 |
[TTY] |   SERAM1 |  0  | 0x00000000 | 0x00000000 |    66752 | 1.0.0 | 0x00000000 |
[TTY] |   HE_DBG |  3  | 0x60000000 | 0x60000000 |     2256 | 1.0.0 | 0x00000063 |
[TTY] | SRV-HP-T |  2  | 0x50000000 | 0x50000000 |    18216 | 1.0.0 | 0x00000062 |
[TTY] ** TEST get SE revision error_code=SERVICES_REQ_SUCCESS                SES A1 EVALUATION_BOARD SE_FW_0.57.000_DEV v0.57.0 Sep  3 2022 01
:14:24 service_resp=0x00000000
[TTY] ** TEST get soc id      error_code=SERVICES_REQ_SUCCESS                Device number 0xA100 service_resp=0x00000000
[TTY] ** TEST read otp        error_code=SERVICES_REQ_INVALID_OTP_OFFSET     service_resp=SERVICES_REQ_INVALID_OTP_OFFSET
[TTY] ** OTP offset 0x59 OTP Value = 0x00000000
[TTY] ** OTP offset 0x5a OTP Value = 0x00000000
[TTY] ** TEST Boot TOC A32    error_code=SERVICES_REQ_SUCCESS                service_resp=0x00000000
[TTY] ** TEST MbedTLS AES     error_code=SERVICES_REQ_SUCCESS                service_resp=0x00000000
[TTY] ** TEST Bounds Tests    error_code=SERVICES_REQ_BAD_PRINT_LENGTH       service_resp=0x00000000
[TTY] ** TEST Bounds Tests    error_code=SERVICES_REQ_BAD_PRINT_LENGTH       service_resp=0x00000000
[TTY] ** TEST Bounds Tests    error_code=SERVICES_REQ_NULL_PARAMETER         service_resp=0x00000000

[SERVICES] Stop mode request not implemented
[TTY] ** TEST EWIC config     error_code=SERVICES_REQ_COMMAND_NOT_IMPLEMENTED Device number 0x41C7 service_resp=0x00000000

[SERVICES] Stop mode request not implemented
[TTY] ** TEST VBAT Wake Up config error_code=SERVICES_REQ_COMMAND_NOT_IMPLEMENTED Device number 0x42AE service_resp=0x00000000
```

Running the M55 HE and M55_HP Host Example from MRAM

This example runs both the M55_HP and M55_HE Application CPUs.

Build the example:

```
$ cd se-host-service-release
$ make install -j 8
$ make install CPU=M55_HP -j 8
$ cd ../app-release-exec/
$ app-gen-toc.exe -f build/config/services-hp-he.json
$ app-write-mram.exe
```

This will program will boot both the M55_HE and M55_HP.

```
[SES] CM0+ frequency is 100 MHz
[SES] os Kernel V10.4.2
[SES] Main Task - looping forever...
[SRV] RX<-- SID= 0x0D0, Receiver ID=2, Address=0x888001F4
[TTY] [M55_HP] SERVICES version 0.0.14
[TTY] ** TEST heartbeat       error_code=SERVICES_REQ_SUCCESS
[TTY] ** TEST pinmux          error_code=SERVICES_REQ_SUCCESS                service_resp=0x00000000
[TTY] ** TEST padcontrol      error_code=SERVICES_REQ_SUCCESS                service_resp=0x00000000
[TTY] ** TEST crypto TRNG 64  error_code=SERVICES_REQ_SUCCESS                64-bit Random value = 0x4a205e22182e92fb service_resp=0
[TTY] ** TEST crypto TRNG 32  error_code=SERVICES_REQ_SUCCESS                32-bit Random value = 0x7c439a6b service_resp=0
[TTY] ** TEST crypto TRNG 64  error_code=SERVICES_REQ_SUCCESS                64-bit Random value = 0x3ec6d03872073595 service_resp=0
[TTY] ** TEST crypto TRNG 32  error_code=SERVICES_REQ_SUCCESS                32-bit Random value = 0x3fac3a27 service_resp=0
[TTY] ** TEST crypto get LCS  error_code=SERVICES_REQ_SUCCESS                LCS State 0x0 service_resp=0
[TTY] ** TEST OTP get data    error_code=SERVICES_REQ_COMMAND_NOT_IMPLEMENTED service_resp=0x000000F3
[TTY] ** TEST TOC get data    error_code=SERVICES_REQ_SUCCESS                Application TOC number = 2 service_resp=0x00000000
[TTY] ** TEST TOC via name HE error_code=SERVICES_REQ_SUCCESS                service_resp=0x00000000
[TTY] ** TEST TOC via name HP error_code=SERVICES_REQ_SUCCESS                service_resp=0x00000000
[TTY] ** TEST TOC via cpuid   error_code=SERVICES_REQ_SUCCESS                id HE_DBG flags 63 1 0 0 service_resp=0x00000000
[TTY] ** TEST TOC via cpuid   error_code=SERVICES_REQ_SUCCESS                id HE_DBG flags 63 1 0 0 service_resp=0x00000000
[TTY] ** TEST TOC version     error_code=SERVICES_REQ_SUCCESS                service_resp=0x000000F3 version=0
[TTY] ** TEST TOC get data    error_code=SERVICES_REQ_SUCCESS                TOC number = 6 service_resp=0x00000000
[TTY] |   Name   | CPU |Load Address|Boot Address|Image Size|Version|    Flags   |
[TTY] +-----------------------------------------------------------------------+
[TTY] |   DEVICE |  0  | 0x00000000 | 0x00000000 |      672 | 0.5.0 | 0x00000000 |
[TTY] |   SERAM0 |  0  | 0x00000000 | 0x00000000 |    66752 | 1.0.0 | 0x00000000 |
[TTY] |   SERAM1 |  0  | 0x00000000 | 0x00000000 |    66752 | 1.0.0 | 0x00000000 |
[TTY] |   HE_DBG |  3  | 0x00000000 | 0x00000000 |     2256 | 1.0.0 | 0x00000063 |
[TTY] | SRV-HE-T |  3  | 0x60000000 | 0x60000000 |    19308 | 1.0.0 | 0x00000063 |
[TTY] | SRV-HP-T |  2  | 0x50000000 | 0x50000000 |    18216 | 1.0.0 | 0x00000062 |
[TTY] ** TEST get SE revision error_code=SERVICES_REQ_SUCCESS                SES A1 EVALUATION_BOARD SE_FW_0.57.000_DEV v0.57.0 Sep  3
[TTY] ** TEST crypto TRNG 32  error_code=SERVICES_REQ_SUCCESS                32-bit Random value = 0x735e07ab service_resp=0

[SERVICES] Stop mode request not implemented
[TTY] ** TEST EWIC config     error_code=SERVICES_REQ_COMMAND_NOT_IMPLEMENTED Device number 0x41C7 service_resp=0x00000000
[TTY] ** TEST crypto TRNG 64  error_code=SERVICES_REQ_SUCCESS                64-bit Random value = 0x2f1b2aadd7f0c0fd service_resp=0

[SERVICES] Stop mode request not implemented
[TTY] ** TEST VBAT Wake Up config error_code=SERVICES_REQ_COMMAND_NOT_IMPLEMENTED Device number 0x42AE service_resp=0x00000000
[TTY] ** TEST crypto TRNG 32  error_code=SERVICES_REQ_SUCCESS                32-bit Random value = 0xbc66139a service_resp=0
[TTY] ** TEST crypto get LCS  error_code=SERVICES_REQ_SUCCESS                LCS State 0x0 service_resp=0
[TTY] ** TEST OTP get data    error_code=SERVICES_REQ_COMMAND_NOT_IMPLEMENTED service_resp=0x000000F3
[TTY] ** TEST TOC get data    error_code=SERVICES_REQ_SUCCESS                Application TOC number = 2 service_resp=0x00000000
[TTY] ** TEST TOC via name HE error_code=SERVICES_REQ_SUCCESS                service_resp=0x00000000
[TTY] ** TEST TOC via name HP error_code=SERVICES_REQ_SUCCESS                service_resp=0x00000000
[TTY] ** TEST TOC via cpuid   error_code=SERVICES_REQ_SUCCESS                id HE_DBG flags 63 1 0 0 service_resp=0x00000000
[TTY] ** TEST TOC via cpuid   error_code=SERVICES_REQ_SUCCESS                id HE_DBG flags 63 1 0 0 service_resp=0x00000000
[TTY] ** TEST TOC version     error_code=SERVICES_REQ_SUCCESS                service_resp=0x000000F3 version=0
[TTY] ** TEST TOC get data    error_code=SERVICES_REQ_SUCCESS                TOC number = 6 service_resp=0x00000000
[TTY] |   Name   | CPU |Load Address|Boot Address|Image Size|Version|    Flags   |
[TTY] +-----------------------------------------------------------------------+
[TTY] |   DEVICE |  0  | 0x00000000 | 0x00000000 |      672 | 0.5.0 | 0x00000000 |
```

Adding ALIF SERVICES to your Application code

Calling SERVICES from your Application requires the following:

- Include the header file `/service-release/include/services_lib_api.h` into your code.
- Link with
  - `/service-release/lib/libservices_m55_lib.a (or _a32_)`
  - `/service-release/lib/libmhu_m55_lib.a (or _a32_)`
- Link with pre-compiled libraries in the CMSIS release
- Copy or create your own service_lib_interface.c file and add to your build.
  - Change any interrupt sources as required.
  - Implement wait function for your environment as required.

SE Host Services Library API

The Host Service API is built on the transport protocol layer. This is to facilitate changing the underlying protocol without affecting the rest of the library.

The services library package consists of the following:

Component	Description
`libservices_m55_lib.a`	Host Services M55 Library
`libservices_a32_lib.a`	Host Services A32 Library
`libmhu_m55_lib.a`	Host Services M55 MHU Library (Baremetal)
`libmhu_a32_lib.a`	Host Services A32 MHU Library (Baremetal)
`services_lib_api.h`	APIs to access the services library
`services_lib_interface.c`	To be completed by the user. Compiled with the host CPU application program

There is a porting / abstraction interface component which the user must update depending upon their operating system choice and driver interface to the Message Handling hardware (MHU).

ALIF supply completed interfaces (currently) for

- Bare metal
- ~~FreeRTOS~~
- ~~ThreadX~~
- Linux

The Host services library provides APIs to facilitate service requests from a host CPU to the SE. it must be set up and initialized before dispatching a Host service request to the SE. It needs access to the MHU driver functions to facilitate MHU communication.

The Host services library also requires other generic functions:

`SERVICES_wait_ms(uint32_t wait_time_ms)`	Delay function
`SERVICES_send_mhu_message_to_se(uint32_t message)`	Interface to the MHU driver

This layer is intended for any Operating System abstraction.

Host Services Library Interface API Porting Layer

This needs to be updated by the user depending upon the operating system being used (or base metal) and the interface to the Message handling hardware. The requirements from the operating system are very light.

The file `services_lib_interface.c` is the porting interface which needs to be filled in by the user.

SERVICES_wait_ms

```
// Delay function
int wait_ms(uint32_t wait_time_ms)
```

SERVICES_send_mhu_message_to_se

```
// MHU send message to SE on MHU0 channel0
int send_mhu_message_to_se(uint32_t message)
```

The above functions must be configured in `services_init_params` structure and pass to the service library initialization function below.

Host Services Library API Layer

A Service call from an application processor looks like any other C function call, it can take parameters and return results via pass by reference parameters.

The Host Services library is responsible for taking the application Service call and communicating this to the Secure Enclave using the MHU.

SERVICES_initialize

```
// Service library initialization
uint32_t SERVICES_initialize(services_lib_t * init_params)

Error_code = SERVICES_initialize(services_lib_t * init_params);

#define SERVICES_INIT_SUCCESS                      0x0
#define SERVICES_INIT_FAILED                       0x1

// Service synchronization
int SERVICES_synchronize_with_se(uint32_t services_handle)

number_of_retries = SERVICES_synchronize_with_se(services_handle);

The M55-HE and M55-HP are started before SERAM is ready to process
service calls. This function sends heartbeat requests until one of
them succeeds.  It returns the number of retries. The maximum number
of retries is 100.
```

SERVICES_send_request

```
// Service request call
uint32_t SERVICES_send_request(uint32_t services_handle,
                               uint16_t service_id,
                               void * service_data,
                               SERVICES_sender_callback callback);

Error_code = SERVICES_send_request(handle, SERVICE_HEARTBEAT_ID,
&service_data, 0);
```

The service request dispatches the service request to the SE. If the callback parameter is NULL, the function waits for the SE to send a response back and then returns an error code. This is analogous to a remote procedure call. If a callback is provided, the call returns immediately after sending the request. The services transport layer calls the provided back when the service response arrives.
It needs access to the host CPUs MHU driver functions to send, receive and ACK messages over the MHU.

SERVICES_send_msg_acked_callback

```
// MHU message ACK callback function
void SERVICES_send_msg_acked_callback(void)
```

The above callback function must be passed to the MHU driver during initialization. It is called by the driver when an MHU message is ACKed by the SE. Channel clear interrupt CH_INT_ST is set when SE has

received the MHU message and SE clears the channel status CH_ST bits by setting CH_CLR. This is assumed to be an ACK from SE that it has received an MHU message sent by the host CPU.

SERVICES_rx_msg_callback

```
// MHU message received callback function
void SERVICES_rx_msg_callback(uint32_t message);
```

The above callback function must be passed to the MHU driver during initialization. It is called by the driver when an MHU message is received from the SE as a response to a service request earlier to the SE

```
// Pinmux service
int PINMUX_config(Port_t port_num, Pin_t pin_num, Pinfunction_t
function);
```

SE Host Service Library Error Codes

The following are the valid return and Error codes for the services library.

Error Code	Value	Meaning
SERVICES_REQ_SUCCESS	0x00	
SERVICES_REQ_NOT_ACKNOWLEDGE	0xFF	
SERVICES_REQ_TIMEOUT	0xFD	
SERVICES_REQ_UNKNOWN_COMMAND	0xFC	

SE Host Services API

The services provided by the SE via the MHU are as follows.

Miscellaneous

SERVICES_Initialize

Syntax:

uint32_t SERVICES_initialize(services_lib_t * init_params)

Description:

Initialize the services.

User needs to supply the following platform specific data and functions for the following operations

- Global address of the CPU's local data memory 0x0 for A32, start of DTCMs for the M55 cores.
- Send MHU message function – provided by the MHU driver
- wait (delay) function – platform and OS specific
- print function – platform and OS specific

Parameters:

init_params	Initialization parameters

Returns:

Restrictions:

None

Example:

```
#include "services_lib_api.h" /* services_lib_t lives here */

int main (void)
{
  uint32_t ErrorCode = SERVICES_OK;
  services_lib_t  services_init_params =
  {
    .wait_ms      = &my_wait_ms_function,
    .send_message = &my_send_mhu_message_function
  };

  ErrorCode = SERVICES_boot_process_toc_entry(&entry_id);
  if (ErrorCode != SERVICES_REQ_SUCCESS)
  {
    return ErrorCode;
  }
```

}

SERVICES_version

Syntax:

const char *SERVICES_version(void)

Description:

Returns the version of the Host library

Parameters:
None

Returns:

Version string

Restrictions:
None

Example:

```
#include <services_lib_api.h>

int main (void)
{
  uint32_t ErrorCode = SERVICES_OK;

  printf("SERVICES version %s\n", SERVICES_version());
}
```

SERVICES_register_channel

Syntax:

Uint32_t SERVICES_version(uint32_t mhu_id, uint32_t channel_number);

Description:

Returns a handle for a specific MHU and channel, to be used in subsequent service calls.

Parameters:

mhu_id	MHU ID
channel_number	Channel number (within the MHU)

Returns:

Service channel handle

Restrictions:
The MHU ID and channel number must be valid

Example:

```
#include <services_lib_api.h>

int main (void)
{
  uint32_t services_handle = SERVICES_register_channel(0, 0);

  printf("SERVICES handle %d\n", services_handle);
}
```

Maintenance Services

The maintenance services provide a mechanism to maintain a reliable connection between the sender and receiver and/or request general information from the receiver. The following maintenance services are supported by SE.

SERVICES_heartbeat

Syntax:

uint32_t SERVICES_heartbeat (uint32_t services_handle)

Description:

Heartbeat request.

This service is analogous to “ping”.

It is a message sent by the sender to tell the receiver that it is alive. It can also be sent by SE to check if another core is alive and responding. When this message is ACKed by the receiver, the sender knows that the receiver is alive. This message does not warrant a response from the receiver other than ACK.

Parameters:

services_handle

Returns:

Restrictions:
None

Example:

```
int main (void)
{
  uint32_t ErrorCode = SERVICES_OK;

  ErrorCode = SERVICES_heartbeat(services_handle);
  if (ErrorCode != SERVICES_REQ_SUCCESS)
  {
    return ErrorCode;
  }
}
```

System Management

SERVICES_system_set_services_debug

Syntax:

```
uint32_t SERVICES_system_set_services_debug (uint32_t services_handle,
                                             bool debug_enable,
                                             uint32_t *error_code)
```

Description:

Enable / Disable Service debug traffic from SES

Parameters:

service_handle	Service Handle
debug_enable	Toggle debug output
error_code	Service Error Code

Returns:

Restrictions:

None

Example:

```
int main (void)
{
  uint32_t ErrorCode = SERVICES_OK;
  uint32_t service_error_code;


SERVICES_system_set_services_debug(services_handle,
                                   false,   /* False = NO debug output */
                                   &service_error_code);
```

SERVICES_get_se_revision

Syntax:

```
uint32_t SERVICES_get_se_revision(uint32_t services_handle,
                                  uint8_t *revision_data,
                                  uint32_t *error_code)
```

Description:

Retrieve the SES Banner string

Parameters:

service_handle	Service Handle
revision_data	banner string return
error_code	Service Error Code

Returns:

Restrictions:

None

Example:

```
int main (void)
{
  uint32_t error_code = SERVICES_OK;
  uint32_t service_error_code;
  uint8_t se_revision[80];

  error_code = SERVICES_get_se_revision(services_handle,
                                        (uint8_t*)&se_revision[0],
                                        &services_error_code);
```

SERVICES_system_read_otp

Syntax:

```
uint32_t SERVICES_system_read_otp      (uint32_t services_handle,
                                        uint32_t otp_offset,
                                        uint32_t *otp_value_word,
                                        uint32_t *error_code)
```

Description:

Read an OTP offset

Parameters:

service_handle	Service Handle
otp_offset	OTP Byte offset to read
otp_value_word	OTP value at otp_offset
error_code	Service Error Code

Returns:
SERVICES_REQ_INVALID_OTP_OFFSET

Restrictions:

Example:

```
int main (void)
{
  uint32_t ErrorCode = SERVICES_OK;
  uint32_t service_error_code;
  uint32_t otp_value;

  ErrorCode = SERVICES_system_read_otp(services_handle,
                                       0x51, /* Offset */
                                       &otp_value, & service_error_code);
  if (ErrorCode != SERVICES_REQ_SUCCESS)
  {
    return ErrorCode;
  }
}
```

SERVICES_system_get_otp_data

Syntax:

```
uint32_t SERVICES_system_get_otp_data (uint32_t services_handle,
                                       SERVICES_otp_data_t *otp_info,
                                       uint32_t * error_code)
```

Description:

Returns details of OTP data

Parameters:

service_handle	Service Handle
otp_info	Details of OTP contents
error_code	Service Error Code

Returns:
SERVICES_REQ_COMMAND_NOT_IMPLEMENTED (for now)

Restrictions:
OTP format is still under definition. This function returns SERVICES_REQ_COMMAND_NOT_IMPLEMENTED (for now). This function will be deprecated eventually.

Example:

```
int main (void)
{
  uint32_t ErrorCode = SERVICES_OK;
  uint32_t service_error_code;
  SERVICES_otp_data_t otp_info;

  ErrorCode = SERVICES_system_get_otp_data(services_handle,
                                           &otp_info, & service_error_code);
  if (ErrorCode != SERVICES_REQ_SUCCESS)
  {
    return ErrorCode;
  }
}
```

SERVICES_system_get_toc_data

Syntax:

```
uint32_t SERVICES_system_get_toc_data (uint32_t services_handle,
```

```
                                        SERVICES_toc_data_t *toc_info,
                                        uint32_t * error_code)
```

Description:

Returns details of TOC objects in MRAM.

```
typedef struct
{
  uint8_t   image_identifier[8];      /**< TOC name    */
  uint32_t  version;                  /**< TOC Version */
  uint32_t  cpu;                      /**< TOC Cpu ID  */
  uint32_t  store_address;
  uint32_t  load_address;             /**< TOC load    */
  uint32_t  boot_address;
  uint32_t  image_size;
  uint32_t  flags;
} SERVICES_toc_info_t;

/**
 * @struct SERVICES_toc_data_t
 */
typedef struct
{
  uint32_t number_of_toc_entries;
  SERVICES_toc_info_t toc_entry[SERVICES_NUMBER_OF_TOC_ENTRIES];

} SERVICES_toc_data_t;
```

The number of TOC entries found is returned followed by the TOC entry details.

Parameters:

service_handle	Service Handle
toc_info	Details for all TOCs
error_code	Service Error Code

Returns:

Restrictions:
None

Example:

```
int main (void)
{
  uint32_t ErrorCode = SERVICES_OK;
```

```
  uint32_t service_error_code;
SERVICES_toc_data_t toc_info;

  ErrorCode = SERVICES_system_get_toc_data(services_handle,
                                           &toc_info, & service_error_code);
  if (ErrorCode != SERVICES_REQ_SUCCESS)
  {
    return ErrorCode;
  }
}
```

SERVICES_system_get_toc_number

Syntax:

```
uint32_t SERVICES_system_get_toc_number(uint32_t services_handle,
                                        uint32_t *toc_number,
                                        uint32_t * error_code)
```

Description:

Returns the number of Table of contents in MRAM

Parameters:

service_handle	Service Handle
toc_number	Number of TOCs
error_code	Service Error Code

Returns:

Restrictions:

None

Example:

```
int main (void)
{
  uint32_t ErrorCode = SERVICES_OK;
  uint32_t number_of_tocs;
  uint32_t service_error_code;

  ErrorCode = SERVICES_system_get_toc_number(services_handle, &number_of_tocs, &
service_error_code);
  if (ErrorCode != SERVICES_REQ_SUCCESS)
  {
    return ErrorCode;
  }
}
```

SERVICES_system_get_toc_via_name

Syntax:

```
uint32_t SERVICES_system_get_toc_via_name(uint32_t services_handle,
                                          const uint8_t *cpu_name,
                                          uint32_t * error_code);
```

Description:

Returns the ??

Parameters:

service_handle	Service Handle
cpu_name	name of Application
error_code	Service Error Code

Returns:

Restrictions:

None

Example:

```
int main (void)
{
  uint32_t ErrorCode = SERVICES_OK;
  uint32_t service_error_code;

  ErrorCode = SERVICES_system_get_toc_via_name(services_handle, (uint8_t *)"M55-HP",
&service_error_code);
  if (ErrorCode != SERVICES_REQ_SUCCESS)
  {
    return ErrorCode;
  }
}
```

SERVICES_system_get_toc_via_cpuid

Syntax:

```
uint32_t SERVICES_system_get_toc_via_cpuid(uint32_t services_handle,
                                           SERVICE_cpuid_t cpuid,
                                           SERVICES_toc_data_t *toc_info,
                                           uint32_t * error_code);
```

Description:

Returns the TOC information for a given CPU.

Valid CPUs are

```
typedef enum
{
  FUSION_A32_0    = 0,
  FUSION_A32_1    = 1,
  FUSION_M55_HP   = 2,
  FUSION_M55_HE   = 3

} SERVICE_cpuid_t;
```

If there are more than one TOC entry per CPUID this will be reflected in the toc_info structure returned from the SERVICE call.

Parameters:

service_handle	Service Handle
cpuid	Which Application CPU
toc_info	ATOC information
error_code	Service Error Code

Returns:

Restrictions:

Example:

```
int main (void)
{
  uint32_t ErrorCode = SERVICES_OK;
  SERVICES_toc_data_t toc_info;
  Uint32_t service_error_code;

 error_code = SERVICES_system_get_toc_via_cpuid(services_handle,
    FUSION_M55_HE, &toc_info, &service_error_code);
  if (ErrorCode != SERVICES_REQ_SUCCESS)
```

```
  {
    return ErrorCode;
  }

  /* Process each TOC entry found */
  for (int each_toc = 0; each_toc < toc_info.number_of_toc_entries; each_toc++)
  {
    SERVICES_toc_info_t *toc_entry_p;

    toc_entry_p = (SERVICES_toc_info_t *)&toc_info.toc_entry[each_toc];

    /* do something with the TOC information */
}
```

SERVICES_system_get_device_part_number

Syntax:

```
uint32_t SERVICES_system_get_device_part_number(uint32_t services_handle,
                                                uint32_t *device_part_number,
                                                uint32_t * error_code)
```

Description:

Returns the SoC device identifier

Parameters:

service_handle	Service Handle
device_part_number	Device id
error_code	Service Error Code

Returns:

Restrictions:
None

Example:

```
int main (void)
{
  uint32_t ErrorCode = SERVICES_OK;
  uint32_t device_id;
  uint32_t service_error_code;

  ErrorCode = SERVICES_system_get_device_part_number(services_handle,
                                                      &device_part_number,
                                                      &service_error_code);
  if (ErrorCode != SERVICES_REQ_SUCCESS)
  {
    return ErrorCode;
  }
}
```

Application Services

Application services provide mechanisms to configure certain functions. The SE can be requested to make these configuration changes.

SERVICES_uart_write

Syntax:

uint32_t SERVICES_uart_write(uint32_t services_handle, size_t size, const uint8_t *uart_data)

Description:

SE-UART write. The buffer provided is printed via the Secure enclave UART port.

Parameters:

services_handle	Service handle
size	Number of bytes to write
uart_data	Buffer containing print data

None

Returns:

Restrictions:
None

Example:

```
int main (void)
{
  uint32_t ErrorCode = SERVICES_OK;
  uint8_t buffer[256];

  ... <format print buffer>

  ErrorCode = SERVICES_uart_write(services_handle,
                                  sizeof(buffer),
                                  (uint8_t *)buffer);
  if (ErrorCode != SERVICES_REQ_SUCCESS)
  {
    return ErrorCode;
  }
}
```

SERVICES_pinmux

Refer document se-mhu-pinmux-pad_configuration

Syntax:

uint32_t SERVICES_pinmux(uint32_t services_handle, uint8_t port_number,
uint8_t pin_number, uint8_t configuration_value,
uint32_t * error_core)

Description:

Pinmux request

Parameters:

services_handle	
port_number	Port Number
pin_number	Pin Number
configuration_value	?
error_code	Service Error Code

Returns:

Restrictions:
None

Example:

```
int main (void)
{
  uint32_t ErrorCode = SERVICES_OK;
  uint32_t service_error_code;

  ErrorCode = SERVICES_pinmux(services_handle, 1, 14, 0, &service_error_code);
  if (ErrorCode != SERVICES_REQ_SUCCESS)
  {
    return ErrorCode;
  }
}
```

SERVICES_padcontrol

NOTE: Refer to document se-mhu-pinmux-pad_configuration

Syntax:

uint32_t SERVICES_padcontrol(uint32_t services_handle, uint8_t port_number,
uint8_t pin_number, uint8_t configuration_value,
uint32_t * error_core)

Description:

Pad control request.

Parameters:

services_handle	
port_number	Port Number
pin_number	Pin Number
configuration_value	?
error_code	Service Error Code

Returns:

Restrictions:
None

Example:

```
int main (void)
{
  uint32_t ErrorCode = SERVICES_OK;
  uint32_t service_error_code;

  ErrorCode = SERVICES_padcontrol(services_handle, 1, 14, 0, &service_error_code);
  if (ErrorCode != SERVICES_REQ_SUCCESS)
  {
    return ErrorCode;
  }
}
```

SERVICES_application_ospi_write_key

Syntax:

uint32_t SERVICES_application_ospi_write_key(uint32_t services_handle, uint32_t command, uint8_t *key, uint32_t * error_code)

Description:

Write an AES decryption key to the OSPI registers. The command field indicates whether to use an externally provided key or a key stored in the OTP, and which OSPI to apply it to – OSPI0 or OSPI1.

```
#define OSPI_WRITE_OTP_KEY_OSPI0             0
#define OSPI_WRITE_OTP_KEY_OSPI1             1
#define OSPI_WRITE_EXTERNAL_KEY_OSPI0        2
#define OSPI_WRITE_EXTERNAL_KEY_OSPI1        3
```

Parameters:

services_handle	Service handle
command	Indicates OSPI0/OSPI1 and external/OTP key
key	Buffer containing print data
error_code	Service error code

Returns:

Restrictions:
None

Example:

SERVICES_SRAM_retention_config

Syntax:

uint32_t SERVICES_SRAM_retention_config(uint32_t services_handle,
uint32_t sram_mem_retention,
uint32_t *service_error_code);

Description:
Configure retention for global SRAM0 or SRAM1

Parameters:

services_handle	Service handle
sram_mem_retention	Which SRAM
service_error_code	Return error code

```
#define POWER_MEM_RETENTION_SRAM0 0x30

#define POWER_MEM_RETENTION_SRAM1 0x31
```

Returns:
ErrorCode - SERVICES_REQ_SUCCESS, SERVICES_REQ_CANNOT_EXECUTE_SERVICE

Service_error_code ERROR_POWER_SRAM_RETENTION_INVALID Incorrect SRAM bank specified.

Restrictions:
REV_A1 does not configure any retention.

Example:

```
int main (void)
{
  uint32_t error_code = SERVICES_REQ_SUCCESS;

  uint32_t service_error_code;

  error_code = SERVICES_SRAM_retention_config(services_handle,
                                              POWER_MEM_RETENTION_SRAM0,
                                              &service_error_code);

  if (error_code != SERVICES_REQ_SUCCESS)
  {
```

```
    return error_code;
  }
}
```

Clock Management

Set or Get System Clock settings

Interrupt muxing

<Action: Add more details>

Event routing

<Action: Add more details>

Power Services

SERVICES_power_stop_mode_request

Syntax:

uint32_t SERVICES_power_stop_mode_request(uint32_t services_handle)

Description:

Request the Secure Enclave to enter stop mode.

Parameters:

services_handle

Returns:
ErrorCode - SERVICES_REQ_SUCCESS, SERVICES_REQ_CANNOT_EXECUTE_SERVICE

Restrictions:

Example:

```
int main (void)
{
  uint32_t ErrorCode = SERVICES_OK;

  error_code = SERVICES_power_stop_mode_request(services_handle);
  if (ErrorCode != SERVICES_REQ_SUCCESS)
  {
    return ErrorCode;
  }
}
```

SERVICES_power_ewic_config

Syntax:

uint32_t SERVICES_power_ewic_config(uint32_t services_handle,

uint32_t ewic_source);

Description:

Configure the EWIC

Parameters:

services_handle

ewic_source EWIC source

Returns:

ErrorCode - SERVICES_REQ_SUCCESS, SERVICES_REQ_CANNOT_EXECUTE_SERVICE

Restrictions:

Example:

```
int main (void)
{
  uint32_t error_code = SERVICES_REQ_SUCCESS;
  uint32_t ewic_config;

  ewic_config &= (1 << 6);
  error_code = SERVICES_power_ewic_config(services_handle,
                                          ewic_config);

  if (error_code != SERVICES_REQ_SUCCESS)
  {
    return error_code;
  }
}
```

SERVICES_power_wakeup_config

Syntax:

uint32_t SERVICES_power_wakeup_config(uint32_t services_handle,

uint32_t vbat_wakeup_source,

services_power_profile_t power_profile)

Description:

Configure the wake up source

Parameters:

services_handle

vbat_wakeup_source Wake up source

```
typedef enum
{
  VBAT_WAKEUP_MDM           = 0x1,          // bit0
  VBAT_WAKEUP_RTC_SE        = 0x10,         // bit4
  VBAT_WAKEUP_RTC_A         = 0x20,         // bit5
  VBAT_WAKEUP_LPCMP         = 0x40,         // bit6
  VBAT_WAKEUP_BROWN_OUT     = 0x80,         // bit7
  VBAT_WAKEUP_LPTIMER       = 0XF00,        // bit11:8
  VBAT_WAKEUP_LPGPIO        = 0XFF0000,     // bit23:16
} SERVICES_wakeup_cfg_t;
```

power_profile Power profile

```
typedef enum
{
  LOWEST_POWER_PROFILE = 0,       /**< LOWEST_POWER_PROFILE */
  HIGH_PERFORMANCE_POWER_PROFILE,/**< HIGH_PERFORMANCE_POWER_PROFILE */
  USER_SPECIFIED_PROFILE,         /**< USER_SPECIFIED_PROFILE */
  DEFAULT_POWER_PROFILE,          /**< DEFAULT_POWER_PROFILE */
  NUMBER_OF_POWER_PROFILES        /**< NUMBER_OF_POWER_PROFILES */
} services_power_profile_t;
```

Returns:

ErrorCode - SERVICES_REQ_SUCCESS, SERVICES_REQ_CANNOT_EXECUTE_SERVICE

Restrictions:

Example:

```
int main (void)
{
  uint32_t error_code = SERVICES_REQ_SUCCESS;

  error_code = SERVICES_power_wakeup_config(services_handle,
                                            VBAT_WAKEUP_RTC_SE
                                            | VBAT_WAKEUP_RTC_A,
                                            LOWEST_POWER_PROFILE);

  if (error_code != SERVICES_REQ_SUCCESS)
  {
    return error_code;
  }
}
```

SERVICES_power_mem_retention_config

Syntax:

uint32_t

SERVICES_power_mem_retention_config(uint32_t services_handle,

uint32_t mem_retention,

services_power_profile_t power_profile)

Description:

Configure memory retention.

Parameters:

services_handle

mem_retention Memory to be retained.

```
// Memory retention bit encoding for mem_retention_enable
#define POWER_MEM_RET_FIREWALL_RAM            0x01UL
#define POWER_MEM_RET_SE_SRAM                 0x02UL
#define POWER_MEM_RET_BACKUP_RAM_4KB          0x04UL
// M55-HE TCM RET1: ITCM 0-128kb; DTCM 0-128kb
#define POWER_MEM_RET_ES1_TCM_RET1            0x08UL
// M55-HE TCM RET1: ITCM 128-256kb; DTCM 128-256kb
#define POWER_MEM_RET_ES1_TCM_RET2            0x10UL
// XTENSA TCM RET1: ITCM 128-512kb
#define POWER_MEM_RET_XTENSA_TCM_RET1         0x20UL
// XTENSA TCM RET1: ITCM 64-128kb
#define POWER_MEM_RET_XTENSA_TCM_RET2         0x40UL
// XTENSA TCM RET1: ITCM 0-64kb
#define POWER_MEM_RET_XTENSA_TCM_RET3         0x80UL
// M55-M TCM RET1: ITCM 1MB; DTCM 384kb
#define POWER_MEM_RET_M55_M_TCM_RET1          0x100UL
#define POWER_MEM_RET_MODEM_BACKUP_RAM_16KB   0x200UL
```

power_profile Power profile

```
typedef enum
{
  LOWEST_POWER_PROFILE = 0,       /**< LOWEST_POWER_PROFILE */
  HIGH_PERFORMANCE_POWER_PROFILE,/**< HIGH_PERFORMANCE_POWER_PROFILE */
  USER_SPECIFIED_PROFILE,         /**< USER_SPECIFIED_PROFILE */
  DEFAULT_POWER_PROFILE,          /**< DEFAULT_POWER_PROFILE */
  NUMBER_OF_POWER_PROFILES        /**< NUMBER_OF_POWER_PROFILES */
} services_power_profile_t;
```

Returns:
ErrorCode - SERVICES_REQ_SUCCESS, SERVICES_REQ_CANNOT_EXECUTE_SERVICE

Restrictions:

Example:

```
int main (void)
{
  uint32_t error_code = SERVICES_REQ_SUCCESS;

  error_code = SERVICES_power_mem_retention_config(services_handle,
                                                   POWER_MEM_RETENTION_SE_RAM,
                                                   LOWEST_POWER_PROFILE);

  if (error_code != SERVICES_REQ_SUCCESS)
  {
    return error_code;
  }
}
```

SERVICES_power_m55_he_vtor_save

Syntax:

SERVICES_power_m55_he_vtor_save(uint32_t services_handle,

uint32_t ns_vtor_addr,

uint32_t se_vtor_addr,

services_power_profile_t power_profile)

Description:

m55-he VTOR value save for wake up

Parameters:

services_handle

ns_vtor_addr	Non-secure VTOR address
se_vtor_addr	Secure VTOR address
power_profile	Power profile

```
typedef enum
{
  LOWEST_POWER_PROFILE = 0,       /**< LOWEST_POWER_PROFILE */
  HIGH_PERFORMANCE_POWER_PROFILE,/**< HIGH_PERFORMANCE_POWER_PROFILE */
  USER_SPECIFIED_PROFILE,         /**< USER_SPECIFIED_PROFILE */
  DEFAULT_POWER_PROFILE,          /**< DEFAULT_POWER_PROFILE */
  NUMBER_OF_POWER_PROFILES        /**< NUMBER_OF_POWER_PROFILES */
} services_power_profile_t;
```

Returns:

ErrorCode - SERVICES_REQ_SUCCESS, SERVICES_REQ_CANNOT_EXECUTE_SERVICE

Restrictions:

Example:

```
int main (void)
{
__uint32_t error_code = SERVICES_REQ_SUCCESS;

    error_code = SERVICES_power_m55_he_vtor_save(services_handle,
                                                 0x0,
```

```
                                                    0x0,
                                                    LOWEST_POWER_PROFILE);

  if (error_code != SERVICES_REQ_SUCCESS)
  {
    return error_code;
  }
}
```

SERVICES_power_m55_hp_vtor_save

Syntax:

SERVICES_power_m55_hp_vtor_save(uint32_t services_handle,

uint32_t ns_vtor_addr,

uint32_t se_vtor_addr,

services_power_profile_t power_profile)

Description:

m55-hp VTOR value save for wake up

Parameters:

services_handle

ns_vtor_addr	Non-secure VTOR address
se_vtor_addr	Secure VTOR address
power_profile	Power profile

```
typedef enum
{
  LOWEST_POWER_PROFILE = 0,       /**< LOWEST_POWER_PROFILE */
  HIGH_PERFORMANCE_POWER_PROFILE,/**< HIGH_PERFORMANCE_POWER_PROFILE */
  USER_SPECIFIED_PROFILE,         /**< USER_SPECIFIED_PROFILE */
  DEFAULT_POWER_PROFILE,          /**< DEFAULT_POWER_PROFILE */
  NUMBER_OF_POWER_PROFILES        /**< NUMBER_OF_POWER_PROFILES */
} services_power_profile_t;
```

Returns:

ErrorCode - SERVICES_REQ_SUCCESS, SERVICES_REQ_CANNOT_EXECUTE_SERVICE

Restrictions:

Example:

```
int main (void)
{
__uint32_t error_code = SERVICES_REQ_SUCCESS;

    error_code = SERVICES_power_m55_hp_vtor_save(services_handle,
                                                 0x0,
```

```
                                                    0x0,
                                                    LOWEST_POWER_PROFILE);

  if (error_code != SERVICES_REQ_SUCCESS)
  {
    return error_code;
  }
}
```

SERVICES_corestone_standby_mode

Syntax:

SERVICES_corestone_standby_mode (uint32_t services_handle,

host_cpu_clus_pwr_req_t host_cpu_clus_pwr_req,

bsys_pwr_req_t bsys_pwr_req,

uint32_t *error_code)

Description:

Function to configure corestone standby mode

Parameters:

services_handle

host_cpu_clus_pwr_req	Host CPU cluster power state request configuration
bsys_pwr_req	Base system power request configuration
power_profile	Power profile

Returns:

`ErrorCode - SERVICES_REQ_SUCCESS, SERVICES_REQ_CANNOT_EXECUTE_SERVICE`

Restrictions:

Example:

```
int main (void)
{
  uint32_t error_code = SERVICES_REQ_SUCCESS;
    host_cpu_clus_pwr_req_t host_cpu_clus_pwr_req;
    bsys_pwr_req_t bsys_pwr_req;

    host_cpu_clus_pwr_req.word = 0;
    host_cpu_clus_pwr_req.bits.mem_ret_req = 0;
    host_cpu_clus_pwr_req.bits.pwr_req = 1;

    bsys_pwr_req.word = 0;
    bsys_pwr_req.bits.systop_pwr_req = 1;
    bsys_pwr_req.bits.dbgtop_pwr_req = 0;
    bsys_pwr_req.bits.refclk_req = 1;
    bsys_pwr_req.bits.wakeup_en = 0;
```

```
    error_code = SERVICES_corestone_standby_mode(services_handle,
                                                 host_cpu_clus_pwr_req,
                                                 bsys_pwr_req,
                                                 &service_error_code);

  if (error_code != SERVICES_REQ_SUCCESS)
  {
    return error_code;
  }
}
```

Reset Services

Set or Get system reset.

<ACTION: Define policy>

Boot Services

SERVICES_boot_process_toc_entry

Syntax:

uint32_t SERVICES_boot_process_toc_entry(uint32_t services_handle, const uint8_t * entry_id, uint32_t * error_code)

Description:

Request to process a TOC entry. Depending on the information in the TOC entry, this could result in the booting of a CPU core.

Parameters:

services_handle
entry_id ID of the TOC entry to process.
The 'entry_id' field is 8 bytes in size, matching the corresponding TOC entry field 'image_identifier'.
error_code Service Error Code

Returns:

Restrictions:
None

SERVICES_boot_cpu

Syntax:

uint32_t SERVICES_boot_cpu(uint32_t services_handle, uint32_t cpu_id, uint32_t address, uint32_t * error_code)

Description:

Request to boot a CPU core. This service does not perform image loading, verification, etc., it just boots the core, specifying the boot address.

Parameters:

services_handle	
cpu_id	ID of the CPU to boot
address	Boot address for the CPU
error_code	Service Error Code

Returns:

```
SERVICES_REQ_SUCCESS
SERVICES_REQ_NOT_ACKNOWLEDGE
SERVICES_REQ_ACKNOWLEDGE
SERVICES_REQ_TIMEOUT
SERVICES_RESP_UNKNOWN_COMMAND
SERVICES_REQ_BAD_PACKET_SIZE
SERVICES_REQ_CANNOT_EXECUTE_SERVICE
SERVICES_REQ_BAD_PAYLOAD
SERVICES_REQ_BAD_PAYLOAD_LENGTH
SERVICES_REQ_PAYLOAD_OK
SERVICES_REQ_PIN_LOCKED
```

Restrictions:

None

SERVICES_boot_release_cpu

Syntax:

uint32_t SERVICES_boot_release_cpu(uint32_t services_handle, uint32_t cpu_id, uint32_t * error_code)

Description:

Request to release a CPU core. This service does not perform image loading, verification, etc., and does not reset the CPU or specify the boot address, it just releases the core.

Supported CPU ids are
FUSION_A32_0
FUSION_A32_1
FUSION_M55_HP
FUSION_M55_HE
FUSION_EXTERNAL_SYS0

Parameters:

services_handle	
cpu_id	ID of the CPU to boot
error_code	Service Error Code

Returns:

Restrictions:

FUSION_EXTERNAL_SYS0 is not a valid operation on FUSION Ensemble or Crescendo devices.

SERVICES_boot_reset_cpu

Syntax:

uint32_t SERVICES_boot_reset_cpu(uint32_t services_handle, uint32_t cpu_id, uint32_t * error_code)

Description:

Request to reset a CPU core, which effectively stops the core.

Parameters:

services_handle	
cpu_id	ID of the CPU to boot
error_code	Service Error Code

Returns:

Restrictions:
None

SERVICES_boot_reset_soc

Syntax:

uint32_t SERVICES_boot_reset_soc(uint32_t services_handle)

Description:

Request to reset the entire SoC.

Parameters:

services_handle

Returns:

Restrictions:
None

Image loading

Image loading, release, run.
<Action: Add more details>

Deferred boot

Request to boot another CPU.
<Action: Add more details>

Crypto Services

The SE provides several crypto services to other cores as detailed below.

SERVICES_cryptocell_get_rnd

Syntax:

uint32_t SERVICES_cryptocell_get_rnd(uint32_t services_handle, uint16_t rnd_length, void * rnd_value, uint32_t * error_code)

Description:

Request random number

The service SERVICES_cryptocell_get_rnd returns a random vector generated by the cryptocell-rt library using the MBedTLS API call mbedtls_ctr_drbg_random().

The desired length of the vector to generate is passed as an input parameter. Currently, the maximum supported vector length is 128 bytes.

Parameters:

services_handle	
rnd_length	Length of random number vector
rnd_value	returned Random number
error_code	Service Error Code

None

Returns:

Restrictions:
None

Example:

```
int main (void)
{
  uint32_t ErrorCode = SERVICES_OK;
  uint64_t rnd_value;
  uint32_t service_error_code;

  ErrorCode = SERVICES_cryptocell_get_rnd(services_handle,
              sizeof(uint64_t),/* random number/vector length in bytes*/
              &rnd_value,
```

```
            &service_error_code);
  if (ErrorCode != SERVICES_REQ_SUCCESS)
  {
    return ErrorCode;
  }
}
```

SERVICES_cryptocell_get_lcs

Syntax:

uint32_t SERVICES_cryptocell_get_lcs(uint32_t services_handle, uint32_t *lcs_state, uint32_t * error_code)

Description:
The service SERVICES_cryptocell_get_lcs returns the current Life Cycle State.

Parameters:

services_handle

lcs_state Life cycle state

error_code Service Error Code

Returns:

Restrictions:
None

Example:

```
int main (void)
{
  uint32_t ErrorCode = SERVICES_OK;
  uint32_t lcs_state;
  uint32_t service_error_code

  ErrorCode = SERVICES_cryptocell_get_lcs(services_handle, &lcs_state,
&service_error_code);
  if (ErrorCode != SERVICES_REQ_SUCCESS)
  {
    return ErrorCode;
  }
}
```

MbedTLS Services

These services expose the hardware accelerated functionality provided by the Arm CryptoCell-RT library in SERAM. Arm has chosen to use MbedTLS as the public API to that functionality. For that reason, the exposed Services correspond to MbedTLS public APIs.

IMPORTANT: **These Services are not intended to be used directly by applications**. Instead, they should be used by a client-side MbedTLS library implementation in which hardware acceleration is done by calling the Services.

To simplify the Services APIs and to avoid introducing MbedTLS types into them, all parameters of the MbedTLS functions are passed as uint32_t. The client-side MbedTLS implementation must convert them to the appropriate types. Also, to reduce the number of Service APIs, some of them cover multiple MbedtTLS API functions.

SERVICES_cryptocell_mbedtls_hardware_poll

Syntax:

```
uint32_t SERVICES_cryptocell_mbedtls_hardware_poll(uint32_t services_handle,
                                                   uint32_t * error_code,
                                                   uint32_t data,
                                                   uint32_t output,
                                                   uint32_t len,
                                                   uint32_t olen)
```

Description:
Service API replacement for mbedtls_hardware_poll()

SERVICES_cryptocell_mbedtls_aes_init

Syntax:

```
uint32_t SERVICES_cryptocell_mbedtls_aes_init(uint32_t services_handle,
                                              uint32_t * error_code,
                                              uint32_t ctx)
```

Description:
Service API replacement for mbedtls_aes_init()

SERVICES_cryptocell_mbedtls_aes_set_key

Syntax:

```
uint32_t SERVICES_cryptocell_mbedtls_aes_set_key(uint32_t services_handle,
                                                 uint32_t * error_code,
                                                 uint32_t ctx,
                                                 uint32_t key,
                                                 uint32_t keybits,
                                                 uint32_t dir)
```

Description:
Service API replacement for mbedtls_aes_set_key_enc() and mbedtls_aes_set_key_dec()

SERVICES_cryptocell_mbedtls_aes_crypt

Syntax:

```
uint32_t SERVICES_cryptocell_mbedtls_aes_crypt(uint32_t services_handle,
                                               uint32_t * error_code,
                                               uint32_t ctx,
                                               uint32_t crypt_type,
                                               uint32_t mode,
                                               uint32_t length,
                                               uint32_t iv,
                                               uint32_t input,
                                               uint32_t output)
```

Description:

Service API replacement for the mbedtls_aes_crypt_XXX functions

SERVICES_cryptocell_mbedtls_sha_starts

Syntax:

```
uint32_t SERVICES_cryptocell_mbedtls_sha_starts(uint32_t services_handle,
                                                uint32_t * error_code,
                                                uint32_t ctx,
                                                uint32_t sha_type)
```

Description:

Service API replacement for mbedtls_sha_starts()

SERVICES_cryptocell_mbedtls_sha_process

Syntax:

```
uint32_t SERVICES_cryptocell_mbedtls_sha_process(uint32_t services_handle,
                                                 uint32_t * error_code,
                                                 uint32_t ctx,
                                                 uint32_t sha_type,
                                                 uint32_t data)
```

Description:

Service API replacement for mbedtls_sha_process()

SERVICES_cryptocell_mbedtls_sha_update

Syntax:

```
uint32_t SERVICES_cryptocell_mbedtls_sha_update(uint32_t services_handle,
                                                uint32_t * error_code,
                                                uint32_t ctx,
                                                uint32_t sha_type,
                                                uint32_t data,
                                                uint32_t data_length)
```

Description:

Service API replacement for mbedtls_sha_update()

SERVICES_cryptocell_mbedtls_sha_finish

Syntax:

```
uint32_t SERVICES_cryptocell_mbedtls_sha_finish(uint32_t services_handle,
                                                uint32_t * error_code,
                                                uint32_t ctx,
                                                uint32_t sha_type,
                                                uint32_t data)
```

Description:

Service API replacement for mbedtls_sha_finish()

SERVICES_cryptocell_mbedtls_ccm_gcm_set_key

Syntax:

```
uint32_t SERVICES_cryptocell_mbedtls_ccm_gcm_set_key(uint32_t services_handle,
   uint32_t * error_code,
   uint32_t context_addr,
   uint32_t key_type,
   uint32_t cipher,
   uint32_t key_addr,
   uint32_t key_bits)
```

Description:

Service API replacement for mbedtls_ccm_set_key() and mbedtls_gcm_set_key()

SERVICES_cryptocell_mbedtls_ccm_gcm_crypt

Syntax:

```
uint32_t SERVICES_cryptocell_mbedtls_ccm_gcm_crypt(uint32_t services_handle,
   uint32_t * error_code,
   uint32_t context_addr,
   uint32_t crypt_type,
   uint32_t length,
   uint32_t iv_addr,
   uint32_t iv_length,
   uint32_t add_addr,
   uint32_t add_length,
   uint32_t input_addr,
   uint32_t output_addr,
   uint32_t tag_addr,
   uint32_t tag_length)
```

Description:

Service API replacement for the mbedtls CCM and GCM crypto functions

SERVICES_cryptocell_mbedtls_chacha20_crypt

Syntax:

```
uint32_t SERVICES_cryptocell_mbedtls_chacha20_crypt(uint32_t services_handle,
   uint32_t * error_code,
   uint32_t key_addr,
   uint32_t nonce_addr,
   uint32_t counter,
   uint32_t data_len,
   uint32_t input_addr,
   uint32_t output_addr)
```

Description:

Service API replacement for mbedtl_chacha20_crypt()

SERVICES_cryptocell_mbedtls_chachapoly_crypt

Syntax:

```
uint32_t SERVICES_cryptocell_mbedtls_chachapoly_crypt(uint32_t services_handle,
   uint32_t * error_code,
   uint32_t context_addr,
   uint32_t crypt_type,
   uint32_t length,
   uint32_t nonce_addr,
   uint32_t aad_addr,
   uint32_t aad_len,
   uint32_t tag_addr,
   uint32_t input_addr,
   uint32_t output_addr)
```

Description:

Service API replacement for the mbedtl chachapoly crypto functions

SERVICES_cryptocell_mbedtls_poly1305_crypt

Syntax:

```
uint32_t SERVICES_cryptocell_mbedtls_poly1305_crypt(uint32_t services_handle,
   uint32_t * error_code,
   uint32_t key_addr,
   uint32_t input_addr,
   uint32_t ilen,
   uint32_t mac_addr)
```

Description:

Service API replacement for mbedtl_poly1305_mac()

SERVICES_cryptocell_mbedtls_cmac_init_setkey

Syntax:

```
uint32_t SERVICES_cryptocell_mbedtls_cmac_init_setkey(uint32_t services_handle,
   uint32_t * error_code,
   uint32_t context_addr,
   uint32_t key_addr,
   uint32_t key_bits)
```

Description:
Service API replacement for mbedtls_cmac_init_setkey()

SERVICES_cryptocell_mbedtls_cmac_update

Syntax:

```
uint32_t SERVICES_cryptocell_mbedtls_cmac_update(uint32_t services_handle,
   uint32_t * error_code,
   uint32_t context_addr,
   uint32_t input_addr,
   uint32_t input_length)
```

Description:
Service API replacement for mbedtls_cmac_update()

SERVICES_cryptocell_mbedtls_cmac_finish

Syntax:

```
uint32_t SERVICES_cryptocell_mbedtls_cmac_finish(uint32_t services_handle,
   uint32_t * error_code,
   uint32_t context_addr,
   uint32_t output_addr)
```

Description:
Service API replacement for mbedtls_cmac_finish()

SERVICES_cryptocell_mbedtls_cmac_reset

Syntax:

```
uint32_t SERVICES_cryptocell_mbedtls_cmac_reset(uint32_t services_handle,
   uint32_t * error_code,
   uint32_t context_addr)
```

Description:
Service API replacement for mbedtls_cmac_finish()

Clocks Services

Services to manipulate the Clock Generation Unit (GCU) settings.

SERVICES_clocks_select_osc_source

Syntax:

uint32_t SERVICES_clocks_select_osc_source (uint32_t services_handle, oscillator_source_t source, oscillator_target_t target, uint32_t * error_code)

Description:

Selects between RC or XTAL clock source for various modules (HF or LF). The selected clock is referred to as the 'OSC' clock.

Parameters:

services_handle	
source	RC or XTAL (either HF or LF, depending on the target)
target	SYS clocks, PERIPH clocks, S32K clock
error code	Service error code

Returns:

Transport layer error code

SERVICES_clocks_select_pll_source

Syntax:

uint32_t SERVICES_clocks_select_pll_source(uint32_t services_handle, pll_source_t source, pll_target_t target, uint32_t * error_code)

Description:

Select OSC or PLL as the source clock for various modules.

Parameters:

services_handle	
source	OSC or PLL
target	SYSREFCLK, SYSCLK, ES0, ES1
error code	Service error code

Returns:

Transport layer error code

SERVICES_clocks_enable_clock

Syntax:

uint32_t SERVICES_clocks_enable_clock(uint32_t services_handle, clock_enable_t clock, bool enable, uint32_t * error_code)

Description:

Enable or disable a clock.

Parameters:

services_handle	
clock	Clock to enable or disable
enable	Enable/disable flag
error code	Service error code

Returns:
Transport layer error code

SERVICES_clocks_set_ES0_frequency

Syntax:

uint32_t SERVICES_clocks_set_ES0_frequency(uint32_t services_handle, clock_frequency_t frequency, uint32_t * error_code)

Description:

Set the frequency of External System 0 (M55-HP).

Parameters:

services_handle	
frequency	Frequency to set
error code	Service error code

Returns:
Transport layer error code

SERVICES_clocks_set_ES1_frequency

Syntax:

uint32_t SERVICES_clocks_set_ES1_frequency(uint32_t services_handle, clock_frequency_t frequency, uint32_t * error_code)

Description:

Set the frequency of External System 1 (M55-HE).

Parameters:
services_handle
frequency Frequency to set
error code Service error code

Returns:
Transport layer error code

SERVICES_clocks_select_a32_source

Syntax:

uint32_t SERVICES_clocks_select_a32_source (uint32_t services_handle, a32_source_t source, uint32_t * error_code)

Description:

Selects the clock source for the A32 CPU cores.

Parameters:
services_handle
source Clock source – CPUPLL, SYSPLL, REFCLK, Clock gate
error code Service error code

Returns:
Transport layer error code

SERVICES_clocks_select_aclk_source

Syntax:

uint32_t SERVICES_clocks_select_aclk_source (uint32_t services_handle, aclk_source_t source, uint32_t * error_code)

Description:

Selects the clock source for the AXI bus.

Parameters:
services_handle
source Clock source – SYSPLL, REFCLK, Clock gate
error code Service error code

Returns:

Transport layer error code

SERVICES_clocks_set_divider

Syntax:

uint32_t SERVICES_clocks_set_divider (uint32_t services_handle, clock_divider_t divider, uint32_t value, uint32_t * error_code)

Description:

Selects the value of a clock divider.

Parameters:

services_handle	
divider	Which divider to set – CPUPLL, SYSPLL, ACLK (Corstone), HCLK, PCLK (Alif)
value	Divider value. 0x0 to 0x1F for Corstone dividers, 0x0 to 0x2 for Alif divider
error code	Service error code

Returns:

Transport layer error code

Lifecycle control

<Action: Add more details>

Update Services

<Action: Add more details>

Document History

Version	Date	Author	Change Log
0.1	Jan 2022	R. ONYETT	Initial concept and realization
0.2	Feb 2022	R. ONYETT	Screenshot updates
0.3	Feb 2022	R. ONYETT	Updated API docs, ARM-DS use
V43-03	Feb 2022	R. ONYETT	Added release version suffix
V44-03	Mar 2022	R. ONYETT	Added SERVICES_uart_write
V45.03	Mar 2022	R. ONYETT	Describe example builds and json files
V46 005	Apr 2022	R. ONYETT	API updates. Changed version
V0.0.6	Apr 2022	R. ONYETT	Added debug toggle API
V0.0.9	May 2022	R. ONYETT	UART write extra parameter
V0.0.10	May 2022	G. Stoykov	Add MbedTLS symmetric crypto services
V0.0.13	July 2022	S. KENKARE	Example restructure. A32 changes.
V0.0.20	Nov 2022	R. ONYETT	Formatting