Software Datasheet

Synergy Software Package (SSP)

Software Descriptive Datasheet

Renesas Synergy™ Platform
Synergy Software
SSP v1.4.0

Renesas Electronics
www.renesas.com

Rev.1.02 May 2018

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Synergy Software Package (SSP) v1.4.0

Contents

1. Description of Renesas Synergy™ Software Package

The Synergy Software Package (SSP), the heart of the Synergy Platform, was designed to industry best practices and tested to commercial standards to be a fully integrated software package that is qualified and maintained by Renesas.

The major components of the SSP include Express Logic's X-Ware™. X-Ware includes the ThreadX® Real Time Operating System (RTOS) plus middleware and stacks, including NetX™ IPv4 and NetX Duo™ IPv4/IPv6 compliant TCP/IP stacks respectively, USBX™ USB Host/Device protocol stack, FileX® MS-DOS compatible file system, and GUIX™ graphics runtime library.

Integrated into the SSP are a large set of Application Frameworks, Hardware Abstraction Layer (HAL) drivers, Board Support Packages (BSPs), additional libraries, all of which are optimized for use with Synergy Microcontrollers (MCU). Developed according to the IEC/ISO/IEEE-12207 Software Life Cycle Process standard while using many of the MISRA C: 2012 coding guidelines, Renesas stands behind this qualified SSP, which is supported and maintained on continuous basis.

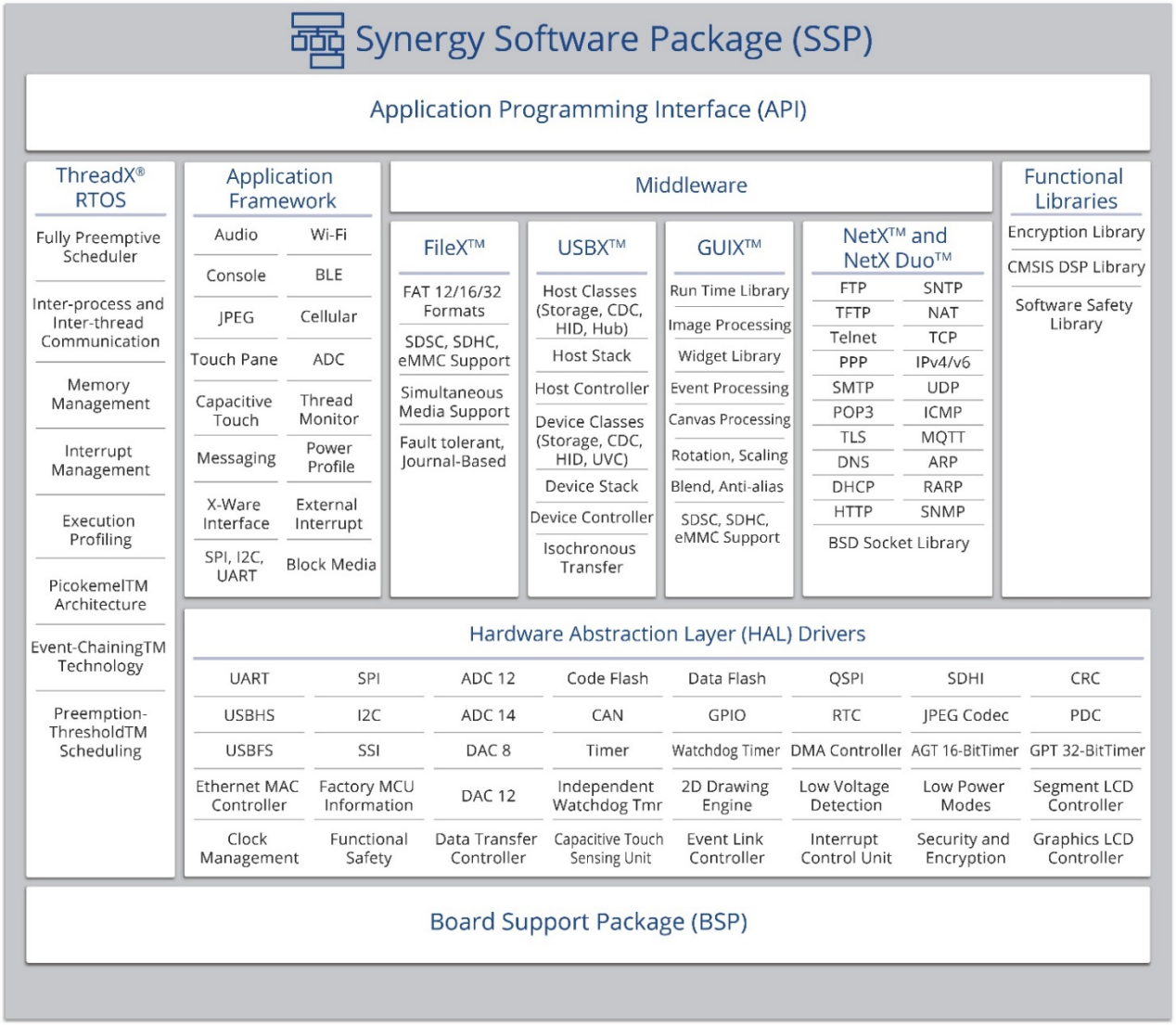


Figure 1.1 Renesas Synergy™ Software Package

SSP works with two development environments.

1. Renesas e^2 studio integrated solutions development environment (ISDE) features a GCC C/C++ Compiler Tool Chain augmented and fortified with unique ISDE Software Configurators. Users can also install and use the IAR compiler with the e^2 studio ISDE
2. IAR Embedded Workbench® for Renesas Synergy™ (IAR EW for Synergy)

Both development environments are available as downloads from the Synergy Gallery. In addition to individual Installers for each major component, SSP v1.3.3 and above (including this SSP v1.4.0 release) have new Platform Installers that make it easier to get up and running quickly.

1.1 Key Features

ThreadX® RTOS

- Multithreaded, deeply embedded, real-time systems
- Small, fast Picokernel™ architecture
- Multitasking capabilities
- Preemptive and cooperative scheduling
- Flexible thread priority support (32-1024 priority levels)
- Small memory footprint and fast response times
- Optimized interrupt handling
- Stack Pointer Overflow Monitor

GUIX™

- Supports 2D Graphics Acceleration in Hardware
- Unlimited objects (screens, windows, widgets)
- Dynamic object creation/deletion
- Alpha blending and anti-aliasing at higher color depths
- Complete windowing support, including viewports and Z-order maintenance
- Multiple canvases and physical displays
- Window blending and fading
- Screen transitions, sprites, and dynamic animations
- Touchscreen and virtual keyboards
- Multilingual support with UTF8 string encoding
- Automatic size scaling
- 8-bit Color Lookup Table (CLUT) support
- Touch Rotation
- Radial Progress Bar

USBX™

- USB 2.0 Full Speed and High Speed support
- Device class: MSC, HID, CDC
- Host class: MSC, HID, CDC ACM, UVC
- Supports fast DMA and isochronous transfers

FileX®

- MS-DOS compatible file system integrated with ThreadX
- FAT12, 16, 32-bit support
- Fault-tolerant file system (uses journaling)
- Multiple media instances

NetX™

- Ethernet Driver
- IPv4 compliant TCP/IP Protocol Stack
- Integrated with ThreadX
- Zero-copy API
- UDP Fast Path Technology
- BSD-compatible socket layer
- RFC 791 Internet Protocol (IP)
- RFC 826 Address Resolution Protocol (ARP)
- RFC 903 Reverse Address Resolution Protocol (RARP)
- RFC 792 Internet Control Message Protocol (ICMP)
- RFC 3376 Internet Group Management Protocol (IGMP)
- RFC 768 User Datagram Protocol (UDP)
- RFC 793 Transmission Control Protocol (TCP)

NetX Duo™

- IPv4 and IPv6 compliant TCP/IP Protocol Stack
- Integrated with ThreadX
- Zero-copy API
- UDP Fast Path Technology
- BSD-compatible socket layer
- RFC 2460 IPv6 Specification
- RFC 4861 Neighbor Discovery for IPv6
- RFC 4862 IPv6 Stateless Address
- RFC 1981 Path MTU Discovery for IPv6
- RFC 4443 ICMPv6
- RFC 791 Internet Protocol (IP)
- RFC 826 Address Resolution Protocol (ARP)
- RFC 903 Reverse Address Resolution Protocol (RARP)
- RFC 792 Internet Control Message Protocol (ICMP)
- RFC 3376 Internet Group Management Protocol (IGMP)
- RFC 768 User Datagram Protocol (UDP)
- RFC 793 Transmission Control Protocol (TCP)
- RFC 1112 Host Extensions for IP Multicasting

NetX™ Applications (IPv4 Networking Services)

- DHCP Client and Server
- DNS Client
- HTTP Client and Webserver
- FTP Client and Server
- TFTP Client and Server
- Telnet Client and Server
- Auto IP
- NAT
- SMTP Client
- POP3 Client and Server
- SNMP Agent
- SNTP Client
- PPP (Not currently supported in Synergy Configuration tool)

NetX Duo™ Applications (IPv4/v6 Networking Services)

- DHCP Client and Server
- DNS Client
- HTTP Client and Webserver
- FTP Client and Server
- TFTP Client and Server
- Telnet Client and Server
- Auto IP
- NAT
- SMTP Client
- POP3 Client and Server
- SNMP Agent
- SNTP Client
- PPP (Not currently supported in Synergy Configuration tool)

NetX Secure

- TLS v1.0 and v1.2

- RFC 1112 Host Extensions for IP Multicasting

Application Frameworks

- Audio Playback framework
- Audio Playback HW DAC framework
- Audio Playback HW I²S framework
- Audio Record framework
- Audio Recording HW ADC framework
- Block Media Interface for SD Multi Media Card
- Bluetooth Low Energy (BLE) framework
- Capacitive Touch Sensing Unit framework
- Capacitive Touch Sensing Unit Button framework
- Capacitive Touch Sensing Unit Slider framework
- Cellular framework
- Console framework
- External Interrupt framework
- I²C framework
- Inter-Thread Messaging framework
- JPEG Decode framework
- Periodic Sampling ADC framework
- Power Profile framework (Version 1)
- Power Profile framework Version 2
- SPI Framework
- Synergy FileX® Interface framework
- Synergy GUIX™ Interface framework
- Synergy NetX Communication framework
- Synergy USBX Communication framework
- Thread Monitor framework
- Touch Panel framework
- UART framework
- Wi-Fi Framework

Security Cryptographic (SCE) Library

- True RNG (TRNG)
- SHA1, SHA224/SHA256
- ECC P-192 and P-256, ECDSA
- AES 128, 192, and 256-bits
- 3DES, 192-bit key, ECB, CBC, CTR
- ARC4
- RSA up to 2048-bit keys
- DLP, DSA up to 2048-bit keys
- Key Generation (plaintext and wrapped keys) and Installation.
- Signature Generation and Verification
- MD5

CMSIS DSP Library

- Basic math functions
- Fast math functions
- Complex math functions
- Filters
- Convolution
- Matrix functions
- Transforms
- Motor control functions
- Statistical functions
- Support functions
- Interpolation functions

MQTT for NetX Duo

- MQTT

Memory support

- Flash programming support via JTAG
- Code and Data Flash drivers
- External memory bus support

Human Machine Interface (HMI)

- Graphics LCD controller driver
- Segment LCD controller driver
- Capacitive Touch Sensing Unit (CTSU)

Hardware Abstract Layer (HAL) Driver Modules

- Analog to Digital Converter (ADC) (12-bit, 14-bit)
- Asynchronous General Purpose Timer (AGT)
- Capacitive Touch Sensing Unit (CTSU)
- Clock Frequency Accuracy Measurement (CAC)
- Clock Generation Circuit (CGC)
- Controller Area Network Interface (CAN)
- Cyclic Redundancy Check calculator (CRC)
- Data Operation Circuit (DOC)
- Data Transfer Controller (DTC)
- Digital to Analog converter (DAC)
- Digital to Analog converter 8-bit (DAC8)
- Direct Memory Access Controller (DMAC)
- Event Link Controller (ELC)
- Factory Microcontroller Information (FMI)
- Flash Memory-High Performance (FLASH_HP)
- Flash Memory-Low Power (FLASH_LP)
- General Purpose I/O Port (GPIO / IOPORT)
- General Purpose Timer (GPT)
- General PWM Timer with Input Capture (GPT_INPUT_CAPTURE)
- Graphics LCD Controller (GLCD)
- I2IC (RIIC)
- Independent Watchdog Timer (IWDT)
- Interrupt Controller Unit (ICU)
- JPEG Codec (JPEG_COMMON, JPEG_ENCODE, JPEG_DECODE)
- Keyboard Interrupt Interface (KINT)
- Deprecated – Low Power Mode (LPM)
- Low Power Mode Version 2 (LPMv2)
- Low Voltage Detection (LVD)
- Parallel Data Capture Unit (PDC)
- Quad SPI (QSPI)
- Real Time clock (RTC)
- SD Multi Media Card (SDMMC)
- Segment LCD (SLCD)
- Serial Communication Interface I²C (SCI_I2C)
- Serial Communication Interface SPI (SCI_SPI)
- Serial Communication Interface UART (SCI_UART)
- Serial Peripheral Interface (SPI)
- Serial Sound Interface (SSI)
- Watchdog Timer (WDT)

Board Support Package (BSP)

- Supports S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9 and S7G2 Group MCUs
- Supports PE-HMI1, DK-S7G2, DK-S3A7, DK-S124, PK-S5D9, and SK-S7G2 Kits, Target Boards for S1JA, S3A1, S3A3, S3A6 and S5D5
- Creation of custom BSPs using e^2 studio
- System initialization and configuration during startup
- Software and hardware access control
- Register Write Protection

GPIO and Key Interrupts

- GPIO module
- Key Interrupts module

1.2 Introduction to this Software Datasheet

This SSP Software Descriptive Datasheet includes functional descriptions for the major software modules in SSP version 1.4.0.

This release supports the following Synergy MCU Groups: **S1JA**, **S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9,** and **S7G2**.

This Software Descriptive Datasheet is designed to give the Developer:

- An inventory of what components and layers are included in SSP.
- An overview of the various layers and components of SSP.
- A basic block diagram for each component.
- Estimated Memory Requirements for each component.

In addition to this Software Descriptive Datasheet, the following resources are available:

- SSP Software User's Manual
- SSP v1.4.0 Release Notes
- SSP Customer Care document
- e² studio Release Notes
- IAR Embedded Workbench® for Renesas Synergy™ Release Notes
- Synergy Standalone Configurator Release Notes
- Application Projects containing Application Notes and Project Code

1.2.1 SSP Warranty

For users holding an SSP Development and Production License, Renesas warrants that the Renesas SSP (Synergy Software Package) will conform to the specification described in the Software Descriptive Datasheet for this release version. The warranty does not cover any other specifications described in any other document. For details, refer to the SSP Customer Care document.

1.2.2 Software Quality Assurance and Test Data

Renesas provides significant and detailed software quality assurance and test data on SSP and its components, modules, and libraries.

For an overview, see: https://www.renesas.com/en-us/products/synergy/software/quality.html

This page lists the latest *Synergy Software Quality Handbook* and *Software Quality Summary*.

1.2.3 Compatible Development Tools

ThreadX, X-Ware, and all of SSP work with the following compatible and tested software development tools:

Tool	Version
e² studio	6.2.0
IAR Embedded Workbench® for Renesas Synergy™	8.21.1
Synergy Standalone Configurator (SSC)	6.2.0
GNU Arm® Compiler (GCC)	GCC_4.9.3.20150529
IAR Compiler	8.21.1

All of SSP is compatible and tested with these tools. See the *SSP Release Notes* for more information.

1.2.4 Supported Kits

The following kits are supported by SSP v1.4.0:

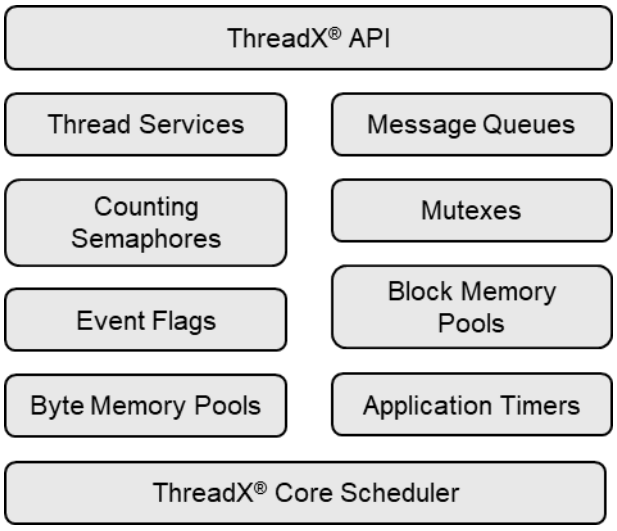
Kit	Version	Description
PE-HMI1	2.0	Product Example (PE) for Human Machine Interface to evaluate Renesas Synergy™ S7G2 Group MCU
DK-S124	3.1	Development Kit for Renesas Synergy™ S124 Group MCU
DK-S7G2	3.1	Development Kit for Renesas Synergy™ S7G2 Group MCU
SK-S7G2	3.3	Starter Kit for Renesas Synergy™ S7G2 Group MCU
DK-S3A7	2.0	Development Kit for Renesas Synergy™ S3A7 Group MCU
PK-S5D9	1.0	Promotion Kit for Renesas Synergy™ S5D9 Group MCU
DK-S128	1.1	Development Kit for Renesas Synergy™ S128 Group MCU
TB-S5D5	1.0	Kit for Renesas Synergy™ S5D5 Group MCU
TB-S3A6	1.05b	Kit for Renesas Synergy™ S3A6 Group MCU
TB-S3A3	0.6	Kit for Renesas Synergy™ S3A3 Group MCU
TB-S3A1	0.5A*	Kit for Renesas Synergy™ S3A1 Group MCU
TB-S1JA	0.5A*	Kit for Renesas Synergy™ S1JA Group MCU
J-Link Software	6.30g	SEGGER J-Link® debug probe is the quasi standard for Arm® Cortex®-M based MCUs.

Note: * Pre-production versions of these kits were used for SSP testing; test results with production kits may vary slightly.

2. ThreadX® RTOS and X-Ware

2.1 Introduction to ThreadX and X-ware

ThreadX® API and X-Ware (NetX/NetX Duo, USBX, FileX®, GUIX™, and the Windows PC side tools TraceX® and GUIX™ Studio) are integrated into SSP.



Component	Express Logic Version
ThreadX®	5.8
NetX™	5.9 SP3
NetX Duo™	5.10 SP3
NetX Application bundle	5.9 SP3
NetX Duo Application bundle	5.10 SP3
NetX Secure	5.11
USBX™ Host	5.8 SP2
USBX™ Device	5.8 SP2
FileX®	5.5
GUIX™	5.4
GUIX™ Studio	5.4.0
TraceX®	5.2.0
MQTT for NetX Duo	5.10 SP4
SNMP for NetX	5.9 SP4
SNMP for NetX Duo	5.10 SP4

Figure 2.1 ThreadX® Features and Versions

2.1.1 Complete List of Software Framework Modules in SSP v1.4.0

Modules are available for respective MCUs based on the following criteria:

1. If the core functionality of the module has been tested and works on a MCU, even if there are known bugs, the module is supported on the MCUs.
2. If the core functionality is broken or not tested on a MCU, then that module is not supported on the MCU.
3. If a module has been tested on one of the Synergy MCUs and is independent of the underlying MCU hardware or HAL drivers, the module is supported on all Synergy MCUs on which the underlying driver/framework/stacks the module depends have been completely tested on that MCU.

Synergy Software X-Ware Stacks	SSP Feature	Supported Synergy MCU Group
fx	FileX	S124, S3A3, S3A6, S3A7, S5D9, S7G2
gx	GUIX	S5D9, S7G2
nx	NetX	S5D9, S7G2
nx_auto_ip	NetX Auto IP	S5D9*, S7G2
nx_bsd	NetX BSD	S5D9*, S7G2
nx_dhcp_client	NetX DHCP Client	S5D9*, S7G2
nx_dhcp_server	NetX DHCP Server	S5D9*, S7G2
nx_dns_client	NetX DNS Client	S5D9*, S7G2
nx_ftp_client	NetX FTP Client	S5D9*, S7G2
nx_ftp_server	NetX FTP Server	S5D9*, S7G2
nx_http_client	NetX HTTP Client	S5D9*, S7G2
nx_http_server	NetX HTTP Server	S5D9*, S7G2
nx_pop3	NetX POP3	S5D9*, S7G2
nx_ppp	NetX PPP	S5D9, S7G2*
nx_smtp_client	NetX SMTP Client	S5D9*, S7G2
nx_snmp_agent	NetX SNMP Agent	S5D9*, S7G2
nx_sntp_client	NetX SNTP Client	S5D9*, S7G2
nx_telnet_client	NetX Telnet Client	S5D9*, S7G2
nx_telnet_server	NetX Telnet Server	S5D9*, S7G2
nx_tftp_client	NetX TFTP Client	S5D9*, S7G2
nx_tftp_server	NetX TFTP Server	S5D9*, S7G2
nxd	NetX Duo Stack	S5D9*, S7G2
nxd_auto_ip	NetX Duo Auto IP	S5D9*, S7G2
nxd_bsd	NetX Duo BSD	S5D9*, S7G2
nxd_dhcp	NetX Duo DHCP IPv4 Client	S5D9*, S7G2
nxd_dhcp	NetX Duo DHCP IPv6 Client	S5D9*, S7G2
nxd_dhcp_server	NetX Duo DHCP IPv4 Server	S5D9*, S7G2
nxd_dhcp_server	NetX Duo DHCP IPv6 Server	S5D9*, S7G2
nxd_dns	NetX Duo DNS Client	S5D9*, S7G2
nxd_ftp_client	NetX Duo FTP Client	S5D9*, S7G2
nxd_ftp_server	NetX Duo FTP Server	S5D9*, S7G2
nxd_http_client	NetX Duo HTTP Client	S5D9*, S7G2
nxd_http_server	NetX Duo HTTP Server	S5D9*, S7G2
nxd_nat	NetX Duo NAT	S5D9*, S7G2
nxd_pop3	NetX Duo POP3	S5D9*, S7G2
nxd_ppp	NetX Duo PPP	S5D9*, S7G2*
nxd_smtp_client	NetX Duo SMTP Client	S5D9*, S7G2
nxd_snmp_agent	NetX Duo SNMP Agent	S5D9*, S7G2
nxd_sntp_client	NetX Duo SNTP Client	S5D9*, S7G2
nxd_telnet_client	NetX Duo Telnet Client	S5D9*, S7G2

Synergy Software X-Ware Stacks	SSP Feature	Supported Synergy MCU Group
nxd_telnet_server	NetX Duo Telnet Server	S5D9*, S7G2
nxd_tftp_client	NetX Duo TFTP Client	S5D9*, S7G2
nxd_tftp_server	NetX Duo TFTP Server	S5D9*, S7G2
nxd_mqtt_client	NetX Duo MQTT Client	S5D9, S7G2
nxd_tls_secure	NetX Duo TLS Secure	S5D9, S7G2
tx	ThreadX	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
ux_device_class_storage	USBX Device Class Mass Storage	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D9, S7G2
ux_device_class_hid	USBX Device Class HID	S124, S128, S3A3, S3A6, S3A7, S5D9, S7G2
ux_device_class_cdc_acm	USBX Device Class CDC-ACM	S124, S128, S3A3, S3A6, S3A7, S5D9, S7G2
ux_host_class_cdc_acm	USBX Host Class CDC-ACM	S3A3, S3A6, S3A7, S5D9, S7G2,
ux_host_class_hid	USBX Host Class HID	S3A3, S3A7, S5D9, S7G2
ux_host_class_hub	USBX Host HUB	S5D9, S7G2
ux_host_class_storage	USBX Host Class Mass Storage	S3A3, S3A7, S5D9, S7G2

* NetX and NetX Duo applications are MCU-independent application layer protocols that depend on the NetX and Ethernet drivers. All MCUs on which NetX has been tested and verified support these protocols.

Experimental modules: Is a module dependent on a SSP HAL driver module that has been tested and supported on a particular MCU, but module itself has not been tested on the MCU. These experimental modules are currently not supported by Synergy Configuration tools. Use of experimental modules in customer projects is not supported by Renesas at this time and is not covered by warranty.

Experimental Modules		
ux_device_class_cdc_ecm	USBX Device Class CDC-ECM	S124, S3A1, S3A3, S3A7, S5D9, S7G2
ux_device_class_rndis	USBX Device Class RNDIS	S124, S3A3, S3A7, S5D9, S7G2
ux_host_class_gser	USBX Host Class Generic Serial	S3A3, S3A7, S5D9, S7G2
ux_host_class_printer	USBX Host Class Printer	S3A3, S3A7, S5D9, S7G2
ux_host_class_prolific	USBX Host Class Prolific	S3A7, S3A3, S5D9, S7G2
ux_host_class_swar	USBX Host Class Swar	S3A7, S3A3, S5D9, S7G2
ux_network_driver	USBX Network Driver	S124, S3A3, S3A7, S5D9, S7G2

2.2 ThreadX® Component Introduction

At the core of Synergy Software Package (SSP) is the Express Logic, Inc. ThreadX® RTOS. Optimized for Synergy MCUs and tightly integrated with the SSP, ThreadX includes an optimized, high-performance real-time kernel designed specifically for real-time embedded systems running on microcontrollers. ThreadX provides a fast, sub microsecond context switching time and a small footprint (as small as 2-KB Flash memory).

The key features of ThreadX include:

- Picokernel™ design where services are not layered
- Preemptive and preemption-threshold scheduling
- Event-chaining
- Inter-task synchronization
- Highly optimized interrupt processing where only scratch registers are saved/restored upon ISR entry/exit, unless preemption is necessary
- Fast interrupt response time
- Fast context switching
- Low RTOS service overhead
- Stack pointer overflow monitor

ThreadX memory protection ensures that application threads and the ThreadX kernel are protected against accidental read or write access from other threads. This prevents code or data corruption from latent application bugs, and eliminates one of the most common causes of application crashes.

2.2.1 ThreadX® Certifications

ThreadX® has been pre-certified by TUV and UL to IEC 61508 SIL 4, IEC 62304 Class C, ISO 26262 ASIL D, UL/IEC 60730, UL/IEC 60335, UL 1998, and SW-SIL EN 50128.

2.2.2 ThreadX® API

With an intuitive and consistent API naming convention (noun-verb naming, all API's have a leading "tx_" to easily identify the call as a ThreadX call), the ThreadX API provide the foundation on which multi-threaded, real-time Internet of Things (IoT) applications can be built.

Blocking API's have an optional thread timeout feature to defeat dead-hangs, and many API's are directly available from application interrupt service routines (ISRs).

2.2.3 Thread Services

With dynamic thread creation, Thread Services allows for an unlimited number of threads (based on available hardware resources and real-time demands)

2.2.4 Message Queues

Like Thread Services, Messages Queues allow for dynamic queue creation. There are no limits on the number of queues (based on available hardware resources and real-time demands). Messages can be copied by value or by reference via pointer. Message sizes are from 1 to 16, 32-bit words. Optional thread suspension on empty and full, and optional timeout on all suspensions, helps to avoid lock-ups.

2.2.5 Counting Semaphores

With dynamic semaphore creation and no limits on the number of semaphores, these 32-bit semaphores provide inter-thread coordination services. Both consumer-producer and resource-protection models are included. Optional thread suspension when the semaphore is unavailable and optional timeout on suspension increase robustness.

2.2.6 Mutexes

Another form of inter-thread communication synchronization, there are no limits to the number of mutexes you can have (based on available hardware resources) and allowing for dynamic mutex creation, this system supports nested resource protection. Optional priority inheritance is supported. Optional thread suspension when the mutex is unavailable are supported as well.

2.2.7 Event Flags

As with other ThreadX® resources, dynamic event flag group creation and no limits on event flag groups (as always, based on available hardware resources), event flags allow synchronization of one thread or multiple threads. Atomic "get" and "clear" is supported, as is optional multi-thread suspension on AND/OR set of events and optional timeout on all suspension.

2.2.8 Block Memory Pools

ThreadX offers Dynamic block pool creation, with no limits on the number of block pools (except for physical memory limits), and no limits on the size of fixed-size blocks or the size of the pool. The fastest possible memory allocation/deallocation is supported, and features like optional thread suspension on empty pool and optional timeout on all suspension are available.

2.2.9 Byte Memory Pools

ThreadX offers Dynamic byte pool creation, with no limits on the number of byte pools (except for physical memory limits), and no limits on the number of byte pools managed. This is the most flexible variable-length memory allocation/deallocation, and allocation size locality is supported. Includes optional thread suspension on empty pool and optional timeout on all suspension are available.

2.2.10 Application Timers

ThreadX® offers Dynamic timer creation, with no limits on the number of timers, and support for periodic or one-shot timers. Periodic timers may have different initial expiration value, and there is no searching on timer activation or deactivation.

2.2.11 ThreadX® Core Scheduler

ThreadX® offers as low as a minimal 2-KB flash footprint, 1-KB RAM footprint capability, but the most important feature of the Scheduler is very fast, sub-microsecond context switch times. Fully deterministic regardless of the number of threads, this priority-based, fully pre-emptive Scheduler has 32 default priority levels (optionally up to 1024). In addition to pre-emptive scheduling, it can also perform cooperative scheduling within a priority level (FIFO). Preemption-threshold technology prevents thread inversion. Optional time services include:

- Per-thread optional time-slice
- Optional time-out on all blocking APIs Requires on hardware time interrupt.

Execution profiling to help tune your application is offered, as well as system-wide trace.

3. SNMP Agent

The NetX and NetX Duo SNMP Agent module provides high-level APIs that implements the SNMP agent for SSP architecture. It's part of the NetX and NetX Duo application bundle included in the SSP X-Ware integration. This module is MCU independent, so any MCU supporting NetX and NetX Duo can implement the SNMP agent.

Note: Unless otherwise stated there is no difference in how this module works in NetX or NetX Duo projects.

Supported Features

- The NetX and NetX Duo SNMP agent module is compliant with RFC1155, RFC1157, RFC1215, RFC1901, RFC1905, RFC1906, RFC1907, RFC1908, RFC2571, RFC2572, RFC2574, RFC2575, RFC 3414 and related RFCs.
- The SNMP agent operates only on UDP. TCP is not supported.
- The SNMP agent module doesn't support Transport Layer Security (TLS) or Datagram Transport Layer Security (DTLS)
- The NetX and NetX Duo SNMP protocol implements SNMP Version 1, 2, and 3. The SNMPv3 implementation supports MD5 and Secure Hash Algorithm 1 (SHA-1) authentication, and Data Encryption Standard (DES) encryption. This version of the NetX and NetX Duo SNMP Agent has the following constraints:
  - One SNMP Agent per NetX IP Instance.
  - No support for RMON.
  - SNMP v3 Inform messages are not supported
- Provides a mechanism to register callbacks for handling username, get, set, and getnext when creating a SNMP agent.

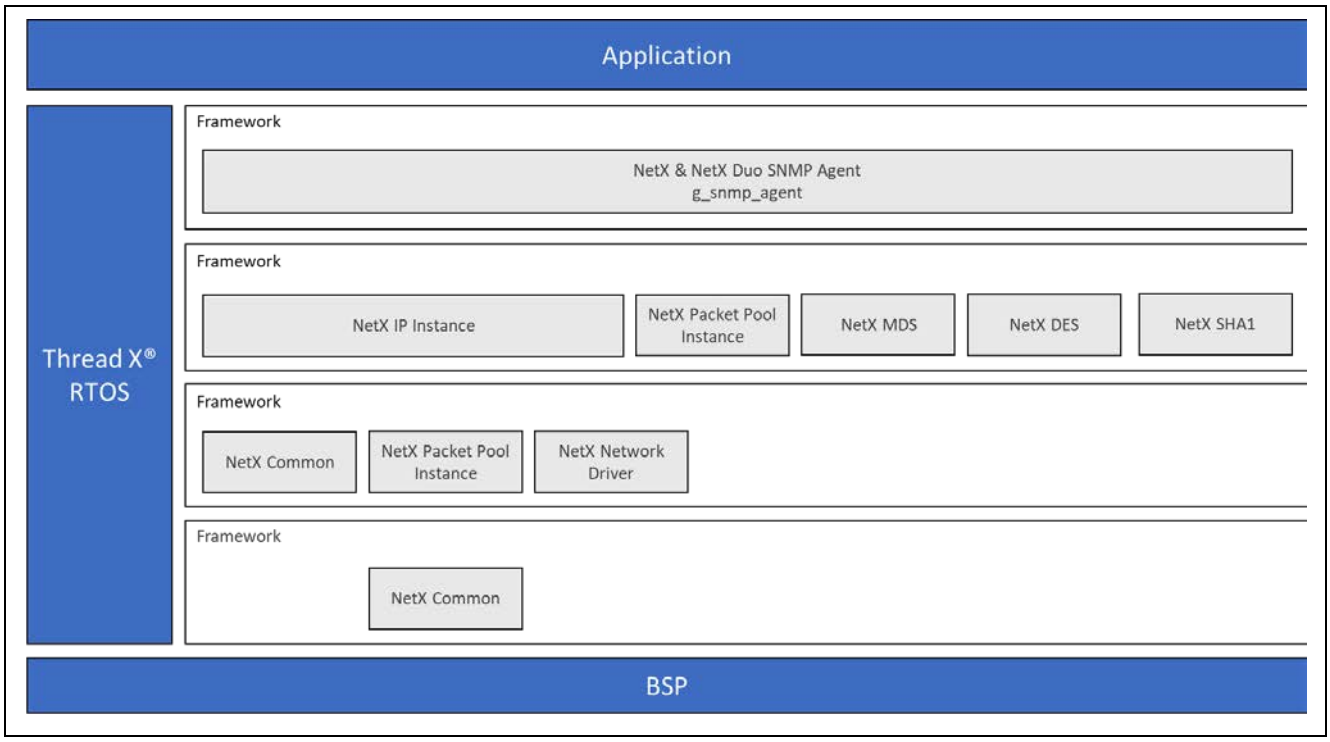


Figure 3.1 SNMP Agent

4. NetX™ Embedded TCP/IP and UDP Stacks

SSP includes an optimized embedded TCP/IP-IPv4-compliant protocol stack, NetX™, for enabling Internet of Things (IoT), Machine to Machine (M2M) communication protocols, and embedded applications that require network connectivity. NetX ™ is completely integrated with ThreadX® and is based on Express Logic's unique Piconet™ architecture that provides a zero-copy API interface for applications. Key features and capabilities provided with NetX™ include:

- Fast execution
- TraceX® system analysis support
- BSD sockets compatible API
- UDP Fast-Path lets basic UDP packets pass through NetX™ without copying or context switches
- Flexible packet management

NetX™ provides a complete set of protocol components that comprise the TCP/IP standard:

- RFC 791 Internet Protocol (IP)
- RFC 826 Address Resolution Protocol (ARP)
- RFC 903 Reverse Address Resolution Protocol (RARP)
- RFC 792 Internet Control Message Protocol (ICMP)
- RFC 3376 Internet Group Management Protocol (IGMP)
- RFC 768 User Datagram Protocol (UDP)
- RFC 793 Transmission Control Protocol (TCP)
- RFC 1112 Host Extensions for IP Multicasting.

In the SSP, both e^2 studio and IAR EW for Synergy have configurators to assist developers creating code for NetX™.

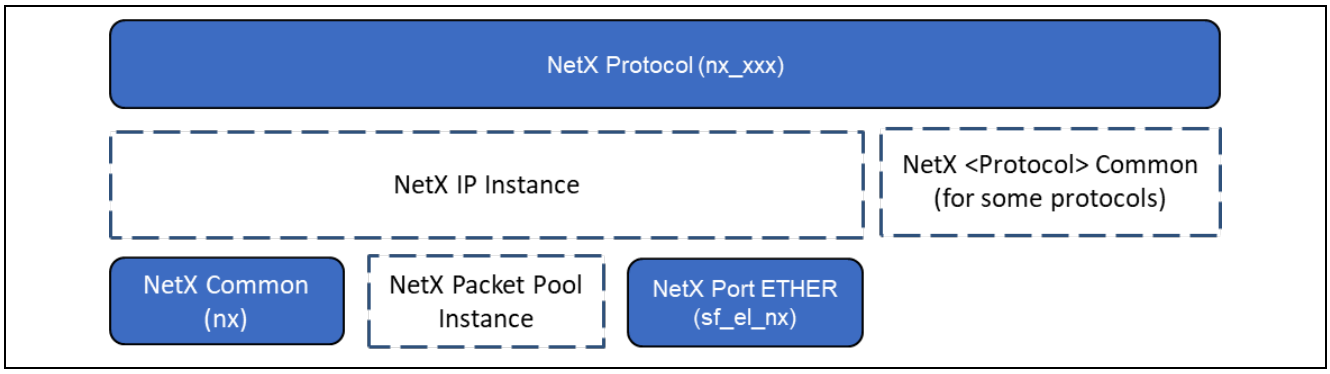


Figure 4.1 NetX™ Stack Configuration

5. NetX ™ Duo™ Dual IPv4/IPv6 Stack

For applications requiring IPv6 support, the SSP includes Express Logic's NetX Duo™, a dual IPv4 and IPv6 compliant TCP/IP protocol stack. NetX Duo™ is completely integrated with ThreadX® RTOS and includes all the features and capabilities available with NetX™. NetX Duo™ further extends the capabilities of SSP-based devices to auto-configure their interface addresses through the Stateless Address Auto configuration protocol. Devices can also use layered structures to enable devices to process IPv6 headers more efficiently. NetX Duo™ applications are individually selectable for each project offering flexibility to the system designer to incorporate only the applications necessary for the target application.

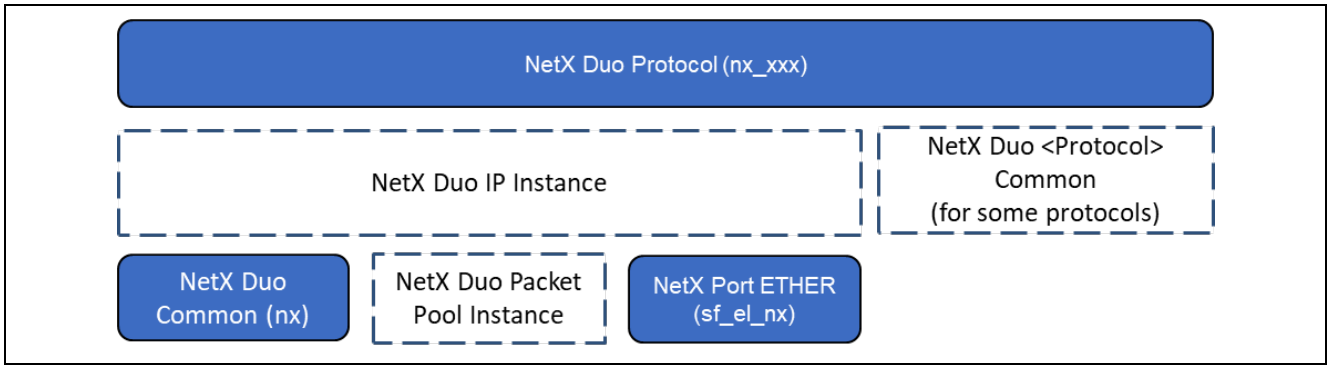


Figure 5.1 NetX Duo™ Stack Configuration

NetX Duo™ implements the following protocols:

- All IPv4 protocols available in NetX™
- Zero-copy API
- UDP Fast Path Technology
- BSD-compatible socket layer
- RFC 2460 IPv6 Specification
- RFC 4861 Neighbor Discovery for IPv6
- RFC 4862 IPv6 Stateless Address Auto configuration
- RFC 1981 Path MTU Discovery for IPv6
- RFC 4443 ICMPv6
- RFC 791 Internet Protocol (IP)
- RFC 826 Address Resolution Protocol (ARP)
- RFC 903 Reverse Address Resolution Protocol (RARP)
- RFC 792 Internet Control Message Protocol (ICMP)
- RFC 3376 Internet Group Management Protocol (IGMP)
- RFC 768 User Datagram Protocol (UDP)
- RFC 793 Transmission Control Protocol (TCP)
- RFC 1112 Host Extensions for IP Multicasting

NetX Duo™ has been accredited by the IPv6 Forum with Phase-II IPv6-Ready Logo certification. NetX Duo™ has been pre-certified by TUV and UL to IEC 61508 SIL 4, IEC 62304 Class C, ISO 26262 ASIL D, UL/IEC 60730, UL/IEC 60335, UL 1998, and EN 50128 SW-SIL 4.

6. NetX™ Applications Bundle

Included with SSP are additional application layer protocols that are frequently used in networking devices:

- Auto IP
  - RFC 3927 Dynamic Configuration of IPv4 Link-Local Addresses
- Dynamic Host Configuration Protocol for Servers (DHCP Server) and Client (DHCP Client)
  - RFC 2131 Dynamic Host Configuration Protocol
  - RFC 2132 DHCP Options and BOOTP Vendor Extensions
- Domain Name System (DNS Client)
  - RFC 1034 Domain Names – Concepts and Facilities
  - RFC 1035 Domain names – Implementation and Specification
  - RFC 1480 The US Domain
  - RFC 2782 A DNS RR for specifying the location of services (DNS SRV)
- HTTP Client and Webserver
  - RFC 1945 Hypertext Transfer Protocol/1.0
  - RFC 2581 TCP Congestion Control
  - RFC 1122 Requirements for Internet Hosts - Communication Layers
- RFC 959 FILE TRANSFER PROTOCOL (FTP) Client and Server
- TFTP Client and Server
  - RFC 1350 The TFTP Protocol (Revision 2)
- Telnet Client and Server
  - RFC 854 Telnet Protocol Specification
- RFC 1939 Post Office Protocol - Version 3 (POP3)
- RFC 1661 The Point-to-Point Protocol (PPP)
  - RFC 1332 The PPP Internet Protocol Control Protocol (IPCP)
  - RFC1334 PPP Authentication Protocols
  - RFC1994 PPP Challenge Handshake Authentication Protocol (CHAP)
- Simple Mail Transfer Protocol (SMTP)
  - RFC 2821 Simple Mail Transfer Protocol (SMTP)
  - RFC 2554 SMTP Service Extension for Authentication
- RFC4330 Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI

These implementations of core networking protocols are thread-safe, compliant with respective RFCs/standards, and have been optimized for memory footprint and CPU utilization. Networking applications are individually selectable for each project providing flexibility to system designer to incorporate only applications necessary for the target application.

7. NetX Duo™ Applications Bundle

Included with SSP are additional application layer protocols that are frequently used in networking devices:

- Auto IP
  - RFC 3927 Dynamic Configuration of IPv4 Link-Local Addresses
- Dynamic Host Configuration Protocol for Servers (DHCP Server) and Client (DHCP Client)
  - RFC 2131 Dynamic Host Configuration Protocol
  - RFC 2132 DHCP Options and BOOTP Vendor Extensions
- Domain Name System (DNS Client)
  - RFC 1034 Domain Names – Concepts and Facilities
  - RFC 1035 Domain names – Implementation and Specification
  - RFC 1480 The US Domain
  - RFC 2782 A DNS RR for specifying the location of services (DNS SRV)

- HTTP Client and Webserver
  - RFC 1945 Hypertext Transfer Protocol/1.0
  - RFC 2581 TCP Congestion Control
  - RFC 1122 Requirements for Internet Hosts - Communication Layers
- RFC 959 FILE TRANSFER PROTOCOL (FTP) Client and Server
- TFTP Client and Server
  - RFC 1350 The TFTP Protocol (Revision 2)
- Telnet Client and Server
  - RFC 854 Telnet Protocol Specification
- RFC 1939 Post Office Protocol - Version 3 (POP3)
- RFC 1661 The Point-to-Point Protocol (PPP)
  - RFC 1332 The PPP Internet Protocol Control Protocol (IPCP)
  - RFC1334 PPP Authentication Protocols
  - RFC1994 PPP Challenge Handshake Authentication Protocol (CHAP)
- Simple Mail Transfer Protocol (SMTP)
  - RFC 2821 Simple Mail Transfer Protocol (SMTP)
  - RFC 2554 SMTP Service Extension for Authentication
- RFC4330 Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI

These implementations of core networking protocols are thread-safe, compliant with respective RFCs/standards, and have been optimized for memory footprint and CPU utilization. Networking applications are individually selectable for each project providing flexibility to the system designer to incorporate only applications necessary for the target application.

8. NetX Secure™ and MQTT for NetX Duo

8.1 NetX Secure™

NetX Secure™ is a high-performance real-time implementation of cryptographic network security standards, including the TLS (Transport Layer Security) that is designed exclusively for embedded ThreadX-based Renesas Synergy applications. NetX Secure™ works with private or public clouds, such as Amazon Web Services™ (AWS) IoT interface and implements APIs similar in format and structure to other Renesas Synergy NetX™/NetX Duo™ APIs.

NetX Secure™ supports the following protocols related to TLS. The list is not necessarily comprehensive, as there are numerous RFCs pertaining to TLS and cryptography. NetX Secure™ follows all general recommendations and basic requirements within the constraints of a real-time operating system with small memory footprint and efficient execution:

- Transport Layer Security (TLS) Protocol
  - RFC 2246, The TLS Protocol Version 1.0
  - RFC 5246, The Transport Layer Security (TLS) Protocol Version 1.2
- X.509 PKI Certificates
  - RFC 5280, X.509 PKI Certificates Version 3
- TLS Cryptography
  - RFC 3268, Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)
  - RFC 3447, Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1
  - RFC 2104, HMAC: Keyed-Hashing for Message Authentication
  - RFC 623, US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)

8.1.1 NetX Secure™ Requirements

To function properly, the NetX Secure run-time library requires that a NetX IP instance has already been created. In addition, and depending on the application, one or more DER-encoded X.509 Digital Certificates are required, either to identify a TLS instance or to verify certificates coming from a remote host.

NetX Secure assumes the existence of ThreadX and NetX/NetX Duo. From ThreadX, it requires thread execution, suspension, periodic timers, and mutual exclusion facilities. From NetX/NetX Duo it requires the TCP/IP networking facilities and drivers

8.1.2 NetX Secure™ Constraints and Support

The NetX Secure™ protocol implements the requirements of the RFC 5246 and Standard(s) for TLS 1.2, as well as backwards-compatibility with RFCs 2246 (TLS 1.0). The following constraints and support levels are applicable to NetX Secure™:

1. Due to the nature of embedded devices, some applications may not have the resources to support the maximum TLS record size of 16 KB. NetX Secure can handle 16 KB records on devices with sufficient resources.
2. Minimal certificate verification: NetX Secure performs basic verification on a certificate to assure that the certificate is valid and signed by a trusted Certificate Authority and can validate the certificate Common Name against the Top-Level Domain Name of the remote host. However, the verification of certificate extensions and other data is the responsibility of the application implementer.
3. All versions of the official "SSL" protocol are considered obsolete and insecure and currently NetX Secure™ does not provide an SSL implementation.
4. The NetX or NetX Duo TCP/IP stack must be initialized prior to using NetX Secure TLS. Refer to the NetX or NetX Duo User Guide for information on how to properly initialize the TCP/IP stack.
5. If a server incorrectly implements TLS, then NetX Secure TLS only connects to the newest version of TLS (1.2) by default.
6. Server implementations that use `bad_record_mac` to indicate a padding error are supported. But whether or not padding errors exist, NetX Secure TLS supports computing the MAC.
7. Server implementations should support constant-time decryption, but on software cryptography library-based systems (in other words, no hardware decryption support). NetX Secure does not yet support constant-time decryption. On hardware supported cryptography, constant-time decryption is supported.
8. Currently, the server certificate and its chain are sent unconditionally.
9. Server implementations shall be able to terminate the connection with a "fatal handshake failure" alert when the client does not have a certificate or an acceptable certificate, and this is supported.
10. Certificate Revocation List (CRL) is supported, but Online Certificate Status Protocol (OCSP) is not yet supported.
11. Basic X.509 chain validation is supported, validating each incoming certificate against the trusted store. The X.509 special value "any-policy" is used for the user-initial-policy-set (this indicates that the application is not concerned about certificate policy).
12. Section 6 of RFC 5280 (see section 3.5.1 of NIST 800-52) defines the certificate path validation. Section 6.1 defines the basic path authentication for which NetX Secure implements a subset. Certificate processing will be expanded to include additional policy enforcement, as specified in section 6.1.3 and 6.1.4 of RFC 5280, to satisfy NIST 800-52. Basic policy enforcement needs to be added. X.509 validation failures terminate the TLS connection.
13. Client implementations should support name constraint checking—to ensure unauthorized certificates are properly rejected—but this is not currently supported.
14. Client implementations should check that the DNS name or IP addresses presented in the client TLS request matches a name or IP address contained in the server certificate's subject distinguished name field or subject alternative name extension. Currently this is application dependent—if a NetX Secure TLS application has DNS enabled, then name checking involves verifying the DNS name in the certificate matches the IP address of the remote server. Since some applications may not include DNS support, DNS and IP address checking will be done in the certificate verification callback.

8.1.3 Ciphers Implemented

NetX Secure currently supports the following ciphers:

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_PSK_WITH_AES_128_CBC_SHA
- TLS_PSK_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_NULL_MD5
- TLS_RSA_WITH_NULL_SHA

Cryptographic support exists for RSA, DES, 3DES, AES, MD5, SHA1, and SHA2. Hardware accelerated cryptography support exists for RSA, AES, SHA1, and SHA2.

8.2 MQTT for NetX Duo™

MQTT (Message Queue Telemetry Transport) is a lightweight, publish-subscribe model useful for low power sensors and other machine-to-machine (M2M) and/or Internet of Things (IoT) situations. The MQTT for NetX Duo implementation is compliant with OASIS MQTT Version 3.1.1 Oct 29th 2014. MQTT is an open standard specification that can be found at: http://mqtt.org/. MQTT for NetX is better suited to constrained environments than HTTP, provides methods for asynchronous communication, and runs over the Internet (IP).

8.2.1 MQTT for NetX Duo Requirements

To function properly, the MQTT for NetX Duo client package requires that a NetX Duo (version 5.10 or later) is installed. Before the MQTT can be used, application must properly configure and start the IP instance, and TCP must be enabled. In addition, if TLS security is required, TLS needs to be configured.

9. FileX® Embedded File System

The SSP provides a high performance and low memory footprint MS-DOS compatible file system, FileX®, for the embedded applications that require file operations. FileX® is implemented as a C library. Only the features used by the application are brought into the final image. The footprint of FileX® is as small as 6 KB. Additionally, FileX® has minimal function call layering, an internal logical sector cache, contiguous cluster allocation, and consecutive cluster reading and writing. All of these attributes make FileX® extremely fast and efficient.

FileX® provides many advanced features for embedded file applications, including the following key capabilities:

- Multiple media instances
- FAT12-, 16-, 32-bit support
- Long file name support
- Contiguous file support
- Consecutive cluster read/write
- Internal logical sector cache
- Fast seek logic
- Multiple partition support
- Fault-tolerant journaled file system

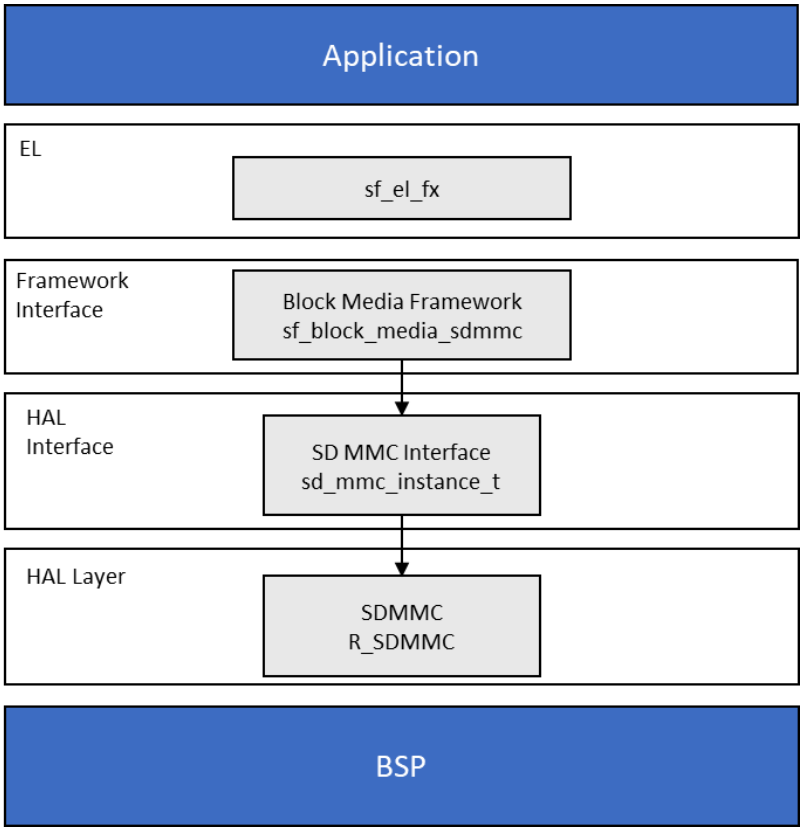


Figure 9.1 FileX® Embedded File System

10. GUIX™ GUI Development Toolkit

GUIX™ is a fully integrated SSP graphical user interface stack with a full-featured runtime UI library and a matching GUI design application for desktop PCs, GUIX™ Studio.

Key features and capabilities of GUIX™ include:

- High reliability, designed for use in fail-safe, safety critical applications
- GUIX™ objects (screens, windows, widgets), limited only by available memory
- Dynamic GUIX™ object creation/deletion
- Support for alpha blending and anti-aliasing at higher color depths
- Complete windowing support, including viewports and Z-order maintenance
- Support for multiple canvases and physical displays, window blending and fading, screen transitions, sprites, and dynamic animations
- Hardware accelerated JPEG and MJPEG Decoding
- Touchscreen and virtual keyboard support
- Multilingual support utilizing UTF-8 string encoding
- Support for 2D Graphics Acceleration in Hardware
- Flexible memory use
- Automatic scaling (object size)
- Small memory footprint
- Endian neutral
- Rotation
- 8-bit Color Lookup Table (CLUT) with either hardware or software rendering

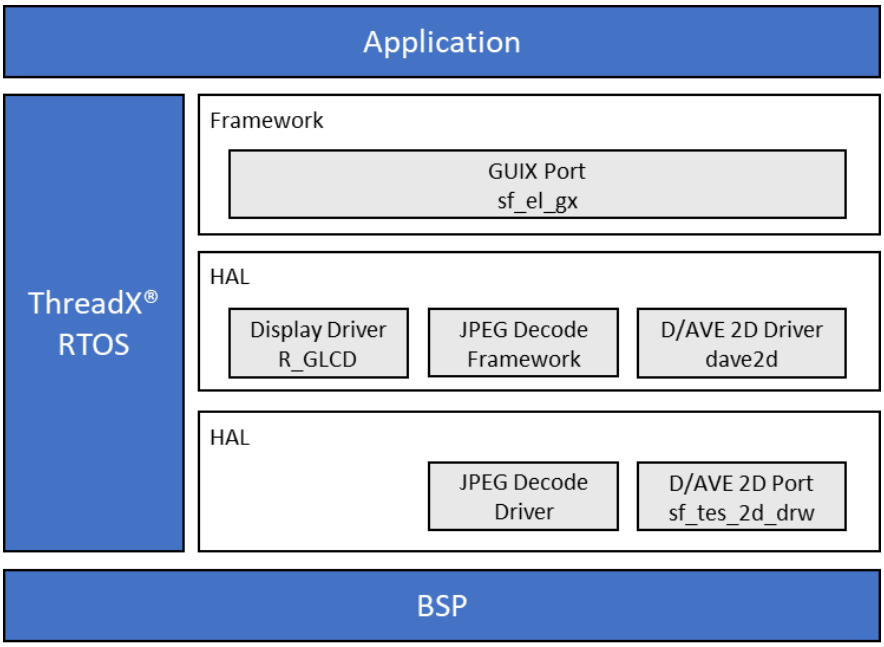


Figure 10.1 GUIX™ Runtime Library

11. USBX™

SSP includes an embedded USB stack fully integrated with ThreadX® and supporting high-performance USB Host and Device modes for embedded applications. USBX requires a small memory footprint and is modular, allowing for only the features used by the application to be included into the final image. This minimizes the footprint of the USB stack being built for the target device. USB Low Speed, Full Speed and High Speed modes are supported. Some of the key features include:

- USB Host and Device supports most of the standard USB class drivers including Mass Storage, HID, and CDC-ACM. UVC and HUB are additional classes supported for USB Host.
- Support for USB Isochronous mode
- Integrated with Express Logic components (FileX® and NetX™).
- Option to build in Device-only mode to reduce code size.
- Pre-build library for USBX for Synergy S124 Group MCU devices defaults to Device mode.

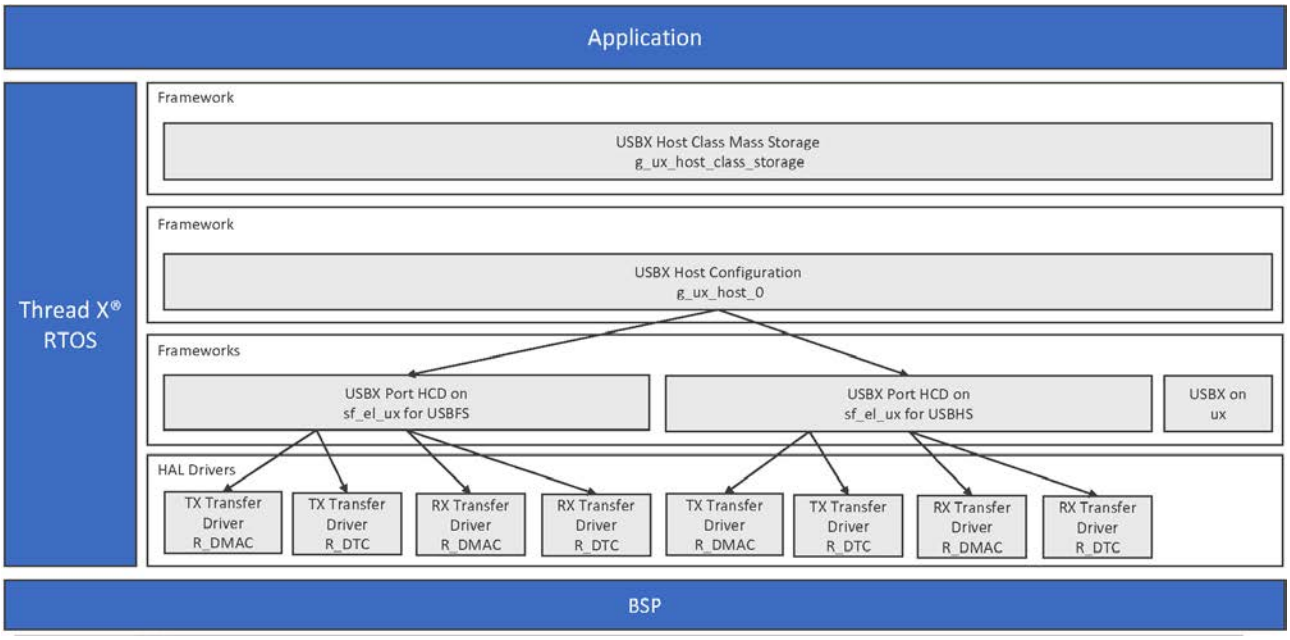


Figure 11.1 USBX™ Host Stacks for Mass Storage (MSC)

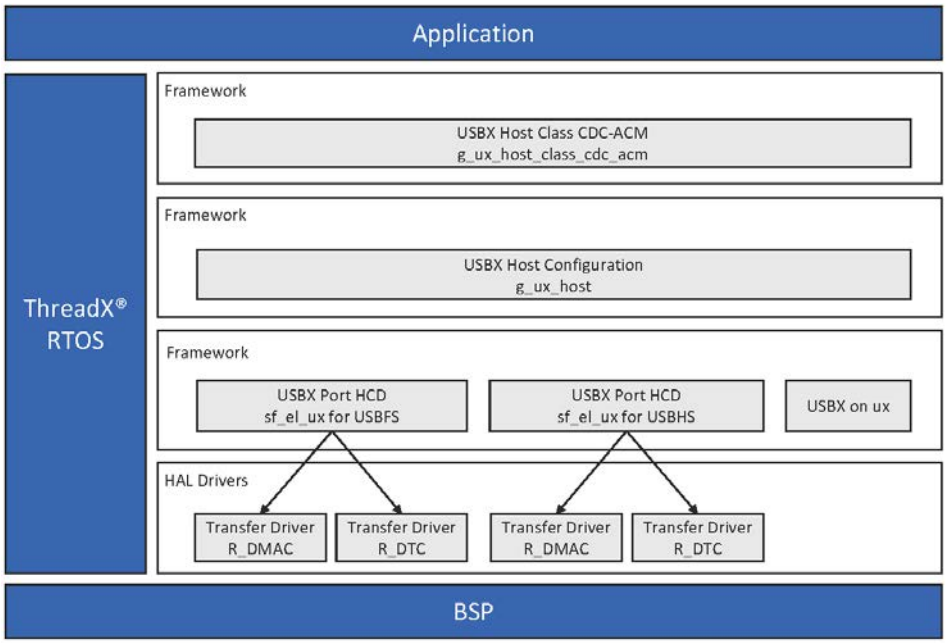


Figure 11.2 USBX™ Host Stacks for CDC ACM

	USBX Mass Storage Class				USBX HID Class			
	Host		Device		Host		Device	
	High Speed	Full Speed	High Speed	Full Speed	High Speed	Full Speed	High Speed	Full Speed
S1JA	N/A	N/A	N/A	✓	N/A	N/A	N/A	✓
S124	N/A	N/A	N/A	✓	N/A	N/A	N/A	✓
S128	N/A	N/A	N/A	✓	N/A	N/A	N/A	✓
S3A1	N/A	✓	N/A	✓	N/A	✓	N/A	✓
S3A3	N/A	✓	N/A	✓	N/A	✓	N/A	✓
S3A6	N/A	☒	N/A	✓	N/A	☒	N/A	✓
S3A7	N/A	✓	N/A	✓	N/A	✓	N/A	✓
S5D5	N/A	✓	N/A	✓	N/A	✓	N/A	✓
S5D9	✓	✓	✓	✓	✓	✓	✓	✓
S7G2	✓	✓	✓	✓	✓	✓	✓	✓

	USBX CDC/ACM				USBX Video Class		USBX HUB Class	
	Host		Device		Host		Host	
	High Speed	Full Speed	High Speed	Full Speed	High Speed	Full Speed	High Speed	Full Speed
S1JA	N/A	N/A	N/A	✓	N/A	N/A	N/A	N/A
S124	N/A	N/A	N/A	✓	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	✓	N/A	N/A	N/A	N/A
S3A1	N/A	✓	N/A	✓	N/A	☒	N/A	N/A
S3A3	N/A	✓	N/A	✓	N/A	☒	N/A	N/A
S3A6	N/A	✓	N/A	✓	N/A	☒	N/A	N/A
S3A7	N/A	✓	N/A	✓	N/A	☒	N/A	N/A
S5D5	N/A	✓	N/A	✓	N/A	☒	N/A	✓*
S5D9	✓	✓	✓	✓	✓	☒	✓	✓
S7G2	✓	✓	✓	✓	✓	☒	✓	✓

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

✓ * denotes the HUB class is tested with self-powered Hubs only.

In the SSP, both e^2 studio and IAR EW for Synergy have configurators to assist developers in creating code for USBX.

12. Application Frameworks

12.1 Introduction

Application Frameworks are a key subsystem in the SSP. Along with the Hardware Abstraction Layer (HAL), Application Frameworks abstract hardware peripherals, as well as link software features together to accomplish one or more tasks. With standardized Applications Programming Interfaces (APIs), frameworks provide a uniform and consistent programming interface for system designers and developers. This approach enables designers and developers to access and apply most of the major features in SSP without worrying about the complexity of the underlying low-level device interfaces in the platform.

Applications Frameworks give developers the flexibility to program at a higher level of abstraction (saving tedious programming work), while allowing direct access to the HAL layer or the board support package (BSP).

Tightly integrated with ThreadX®, Application Frameworks provide thread-safe APIs for accessing shared resources and managing access conflicts, and also provide mutual exclusion and synchronization services amongst application tasks. Application Frameworks help a Developer write code much faster, and more accurately, by combining tasks into logical APIs. Application Frameworks in the SSP link the RTOS with the HAL and provide high-level, C-callable interfaces for commonly used platform system services.

Framework APIs are controlled, engineered by Renesas and maintained to provide consistency. Moreover, they are thread-safe and re-entrant. If the Developer wishes to access HAL driver directly (peripheral drivers), they may do so. While HAL drivers do not use any RTOS objects, nor make any RTOS API calls, they can be used with, or without, the RTOS.

Framework layer modules make use of RTOS objects such as inter-thread communication tools such as semaphores, mutexes, and event flags. They can create their own objects when needed, and access underlying hardware via the HAL.

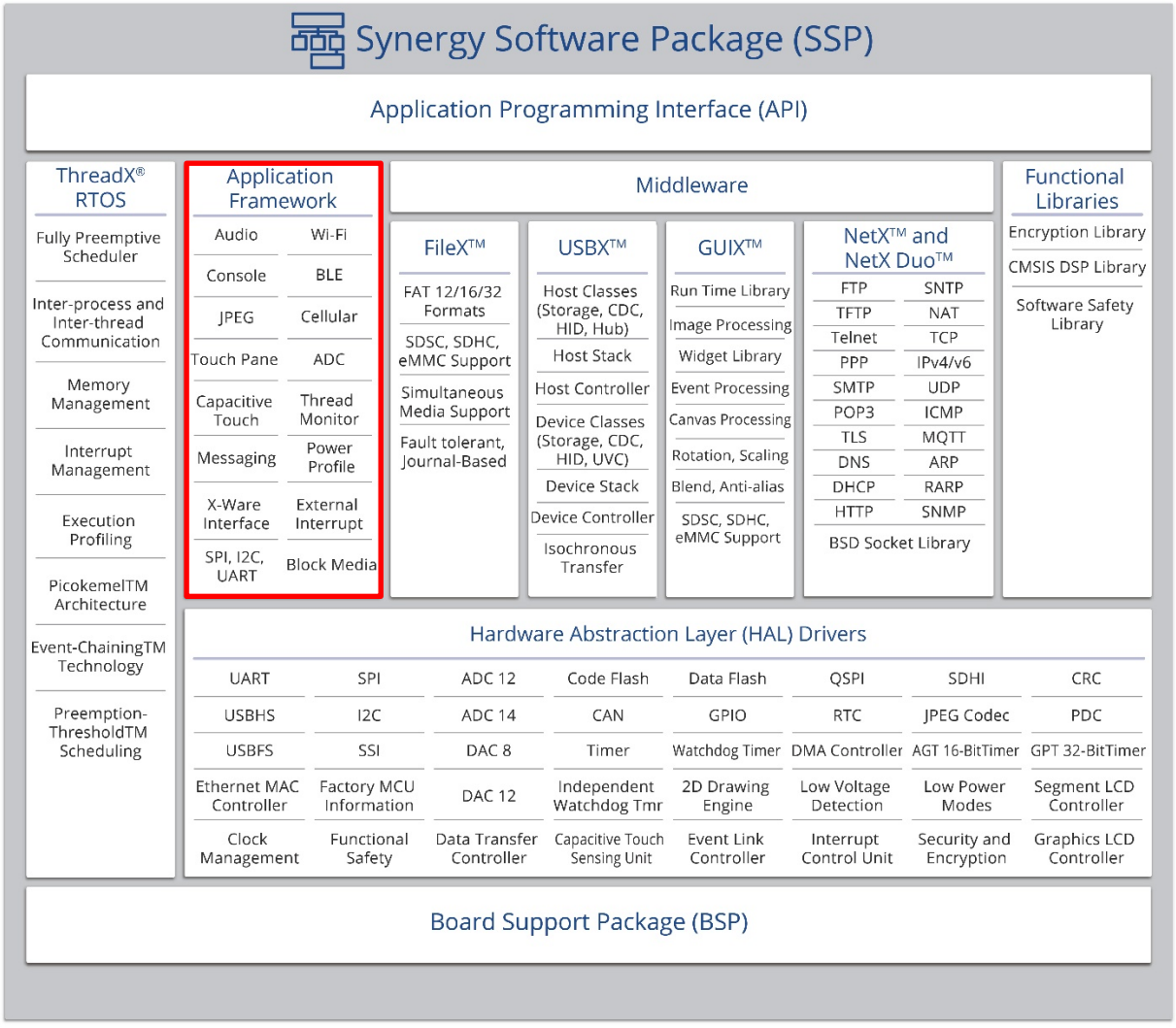


Figure 12.1 Application Frameworks

12.2 Complete List of Software Framework Modules in SSP v1.4.0

Modules are available for respective MCUs based on the following criteria:

1. If the core functionality of the module has been tested and works on a MCU, even if there are known bugs, the module is supported on the MCU.
2. If the core functionality is broken or not tested on a MCU, then that module is not supported on the MCU.
3. If a module has been tested on one of the Synergy MCUs, and is independent of the underlying MCU hardware or HAL drivers, the module is supported on all Synergy MCUs on which the underlying driver/framework/stacks the module depends have been completely tested on that MCU.

Synergy Software Framework	SSP Feature	Supported Synergy MCU Group
sf_adc_periodic	Periodic Sampling ADC	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_audio_playback	Audio Playback	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_audio_playback_hw_dac	Audio Playback HW DAC	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_audio_playback_hw_i2s	Audio Playback HW I2S	S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_audio_record_adc	Audio Record HW ADC	S124, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_audio_record_i2s	Audio Record I2S	S3A1, S3A3, S3A6, S3A7, S5D5,S5D9, S7G2
sf_ble_rl78g1d	BLE Framework	S124, S128, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_ble_rl78g1d_onboard_profile	BLE Framework Onboard Profiles	S124, S128, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_block_media_qspi	File System Support on QSPI	S3A3, S3A7, S5D5, S5D9, S7G2
sf_block_media_ram	File System Support on RAM	S3A3, S3A7, S5D9, S7G2
sf_block_media_sdmmc	Block Media Interface for SD Multi Media Card	S3A3, S3A7, S5D9, S7G2
sf_cellular_cat1	Cellular Framework CAT1	S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_cellular_cat1_socket	Cellular Framework CAT1 Socket	S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_cellular_cat3	Cellular Framework CAT3	S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_cellular_cat3_socket	Cellular Framework CAT3 Socket	S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_cellular_catm1	Cellular Framework CATM1	S5D9, S7G2
sf_cellular_catm1_socket	Cellular Framework CATM1 Socket	S5D9, S7G2
sf_comms_telnet	Synergy Telnet Communication Interface	S5D5, S5D9, S7G2
sf_console	Console	S124, S128, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_comms_telnet	Synergy Telnet Communication Interface	S5D5, S5D9, S7G2
sf_crypto#	Crypto Framework Common	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_crypto_cipher#	Crypto Framework Cipher	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_crypto_hash#	Crypto Framework Hash	S5D5, S5D9, S7G2
sf_crypto_key#	Crypto Framework Key	S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2

Synergy Software Framework	SSP Feature	Supported Synergy MCU Group
sf_crypto_key_installation[#]	Crypto Framework Key Installation	S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_crypto_key_signature[#]	Crypto Framework Key Signature	S5D5, S5D9, S7G2
sf_crypto_trng[#]	Crypto True Random Number Generator	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9,S7G2
sf_el_fx	Synergy FileX interface	S3A1, S3A3, S3A7, S5D9, S7G2
sf_el_gx	Synergy GUIX Interface	S7G2, S5D9
sf_el_nx	Synergy NetX Interface	S5D5, S5D9, S7G2
sf_el_nx_comms	Synergy NetX Communication Interface	S5D5, S5D9, S7G2
sf_el_ux	Synergy USBX Interface	S1JA, S124, S128, S3A1, S3A3, S3A7, S5D5, S5D9, S7G2
sf_el_ux_comms[†]	Synergy USBX Communication Interface	S1JA, S124, S128, S3A1, S3A3, S3A7, S5D5, S5D9, S7G2
sf_el_ux_comms_v2	Synergy USBX Communication Interface V2	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_external_irq	External Interrupt	S1JA, S124, S128, S3A1,S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_i2c	I2C Framework	S124, S128, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_jpeg_decode	JPEG Decode	S5D9, S7G2
sf_message	Inter-Thread Messaging	S3A1,S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_power_profiles†	Power Mode Profile	S124, S3A7, S7G2
sf_power_profiles_v2	Power Mode Profile V2	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_spi	SPI Framework	S1JA, S124, S128, S3A1,S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_tes_2d_drw	2D Drawing Engine Framework	S5D9, S7G2
sf_thread_monitor	Thread Monitor (Watchdog)	S1JA,S124, S128, S3A1,S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_touch_ctsu	Capacitive Touch Sensing Unit	S124, S128, S3A7, S7G2
sf_touch_ctsu_button	Capacitive Touch Sensing Unit Button	S124, S128, S3A7, S7G2
sf_touch_ctsu_slider	Capacitive Touch Sensing Unit Slider	S124, S128, S3A7, S7G2
sf_touch_panel_i2c	Touch Panel I2C	S5D9, S7G2
sf_uart_comms	UART Framework	S124, S128, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
sf_wifi_gt202	WiFi Framework	S3A3, S3A7, S5D5, S5D9, S7G2
sf_wifi_gt202_onchip	WiFi framework on Chip Stack	S3A3, S3A7, S5D5, S5D9, S7G2

[†] Indicates a module that is deprecated starting with SSP v1.3.0 and all subsequent versions. Deprecated modules will only be available to maintain compatibility with existing projects that may be using them. It is highly recommended that new projects use the recommended replacements and not use deprecated modules. For details, see the SSP User's Manual.For details, see the SSP User's Manual.

[#] **Cryptographic Functions**: See Table 14.1 which lists cryptographic functions available for each MCU in this release; these functions are accessible as part of r_sce/cryptographic library.

12.3 ADC Periodic Framework

The ADC Periodic Framework provides high-level APIs and is implemented on sf_adc_periodic. The module configures the ADC to sample any of the available channels (using the single-scan mode) at a configurable rate and buffers the data for a configurable number of sampling iterations before notifying the application. The ADC Periodic Framework uses the ADC, GPT or AGT, and DTC peripherals on the Renesas Synergy™ Microcontroller hardware. A user-defined callback can be created to process the data each time a new sample is complete.

ADC Periodic Framework Features

- Supports 14 and 12-bit A/D Converter on applicable Synergy MCUs
- Multiple Operation Modes
- Single Scan
- Group Scan
- Continuous Scan
- Multiple Channels
- 13 channels (unit 0), 12 channels (unit 1) for S7G2 and S5D9 MCU Groups
- 18 channels for S124
- 28 channels for S3A7

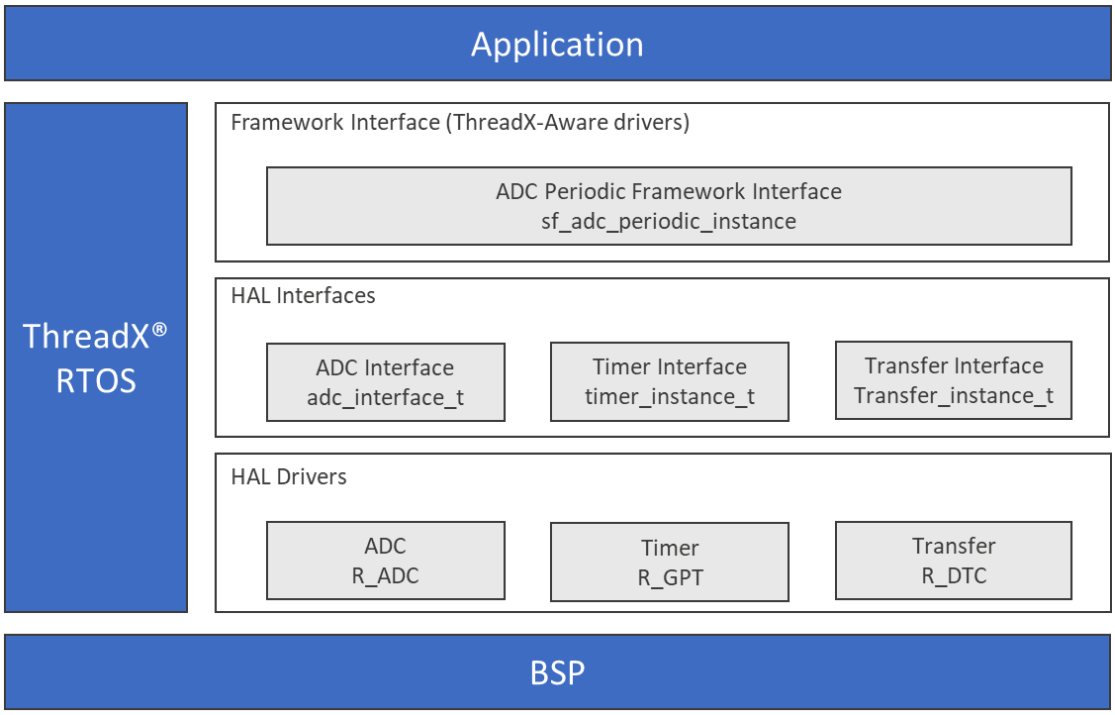


Figure 12.2 Periodic Sampling ADC Framework

12.4 Audio Playback Framework

The Audio Playback Framework module provides high-level APIs for audio playback applications and is implemented on either `sf_audio_playback_hw_dac` or `sf_audio_playback_hw_i2s`. It handles the synchronization needed to play 16 and 8-bit pulse-code modulation (PCM) samples. The Audio Playback Framework uses the DAC or I2S, timer (AGT or GPT) and data-transfer (DMA or DTC) peripherals on a Synergy MCU. A user-defined callback can be created to respond to additional data needs.

The Audio Playback Hardware DAC Framework module supports the following features:

- Plays long buffers by splitting the data into manageable amounts.
- Repeats playback until a ThreadX timeout (for repeated audio like sine wave tones or looped background music)
- Requests next data using callback after last buffer playback begins
- Software volume control
- Pause and resume functions
- Scaling, for example, to move signed 16-bit PCM data into range of the unsigned 12-bit DAC
- Basic mixing for multiple streams

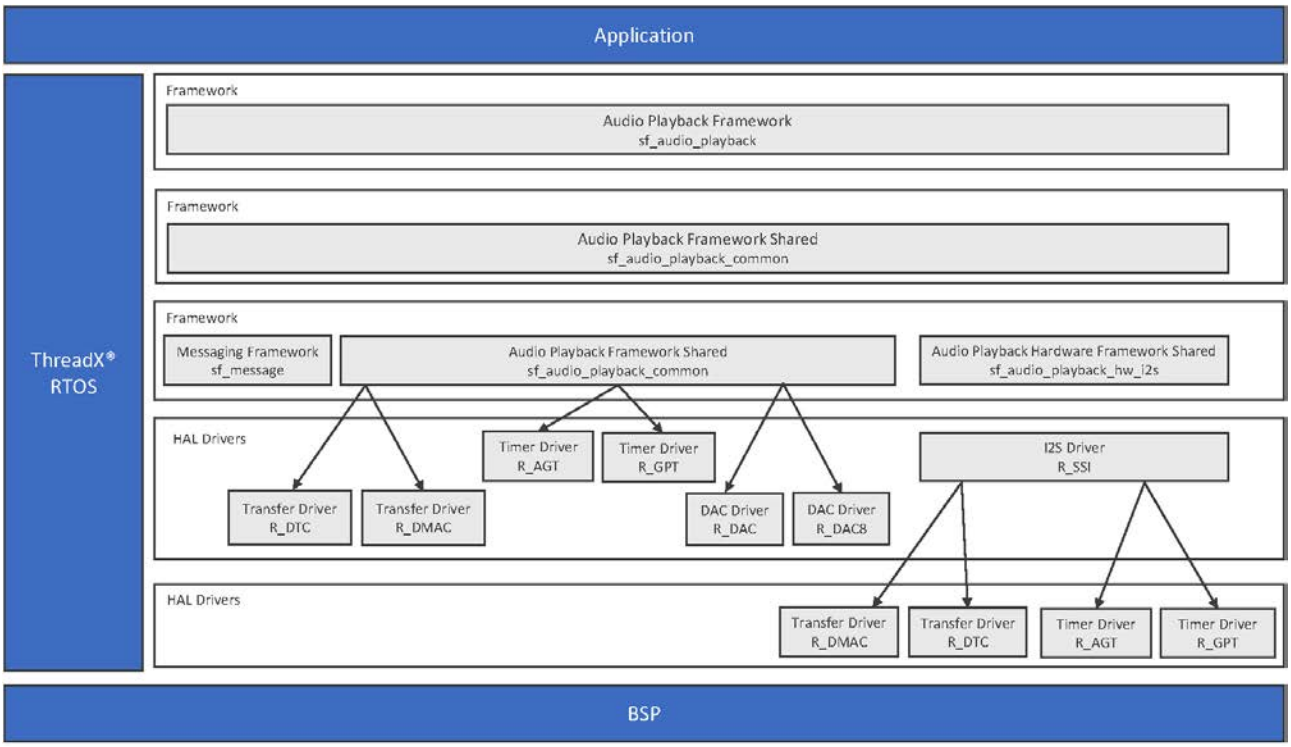


Figure 12.3 Audio Playback Stack

12.5 Audio Playback DAC Framework

The Audio Playback DAC Framework module provides high-level APIs for audio playback applications and is implemented on `sf_audio_playback_hw_dac`. It handles the synchronization needed to play 8 and 16-bit pulse-code modulation (PCM) samples. The Audio Playback DAC Framework uses the DAC, timer (AGT or GPT) and data transfer (DMA or DTC) peripherals on a Synergy MCU. A user-defined callback can be created to respond to the need for additional data.

The Audio Playback Hardware DAC Framework supports the following features:

- Plays long buffers by splitting the data into manageable amounts
- Repeats playback until a ThreadX timeout (for repeated audio like sine wave tones or looped background music)
- Requests next data using callback after last buffer playback begins
- Software volume control
- Pause and resume functions
- Scaling- for example, to move signed 16-bit PCM data into range of the unsigned 12-bit DAC
- Basic mixing for multiple streams

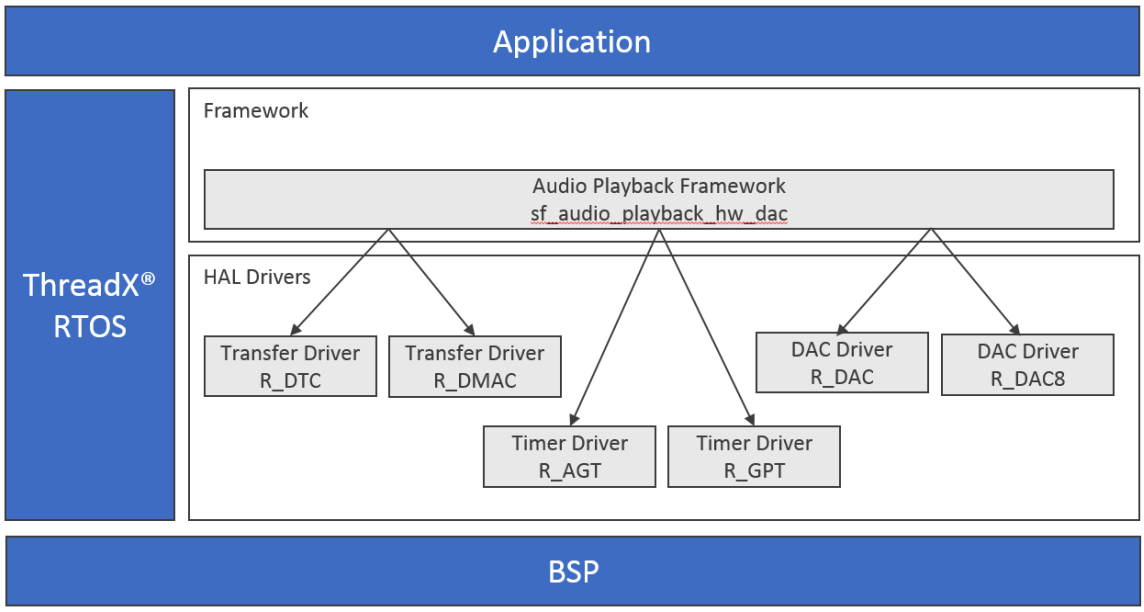


Figure 12.4 Audio Playback DAC Framework

12.6 Audio Playback I²S Framework

The Audio Playback I²S Framework module provides high-level APIs for Audio Playback applications and is implemented on sf_audio_playback_hw_i2s. It handles the synchronization needed to play 8 and 16-bit pulse-code modulation (PCM) samples. The Audio Playback Framework uses the I²S, Timer (AGT or GPT) and Data Transfer (DMA or DTC) peripherals on a Synergy MCU. A user defined callback can be created to respond to the need for additional data.

The Audio Playback I²S Hardware Framework supports the following features:

- Play long buffers by splitting the data into manageable chunks.
- Repeat playback until ThreadX timeout (for repeated audio like sine wave tones or looped background music).
- Request next data using callback after last buffer playback begins.
- Software volume control.
- Pause and resume functions.
- Basic mixing for multiple streams.

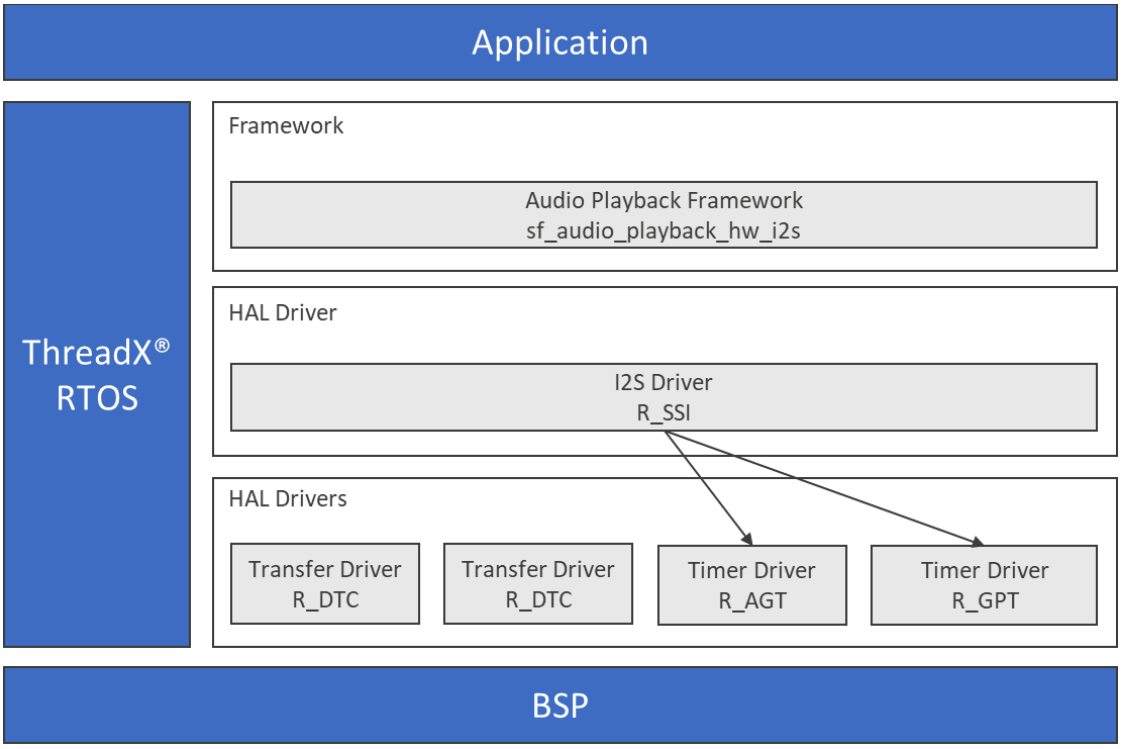


Figure 12.5 Audio Playback I²S Framework

12.7 Audio Record ADC Framework

The Audio Record ADC Framework module provides high-level API for audio recording applications and is implemented on sf_adc_periodic. The Audio Record ADC Framework module uses the sf_adc_periodic and its lower layer ADC, GPT and DTC peripherals on the Synergy MCU. A user-defined callback can be created to indicate that the sample count has been completed.

- Audio Record ADC Framework supports the following features:
- Records data in 8 or 12-bit PCM
- Uses ADC Periodic Framework to simplify configuration and integration
- Uses ThreadX object, like mutex, to protect hardware from improper access
- APIs for high-level functions simplify coding:
  - open, start
  - stop, infoGet
  - close

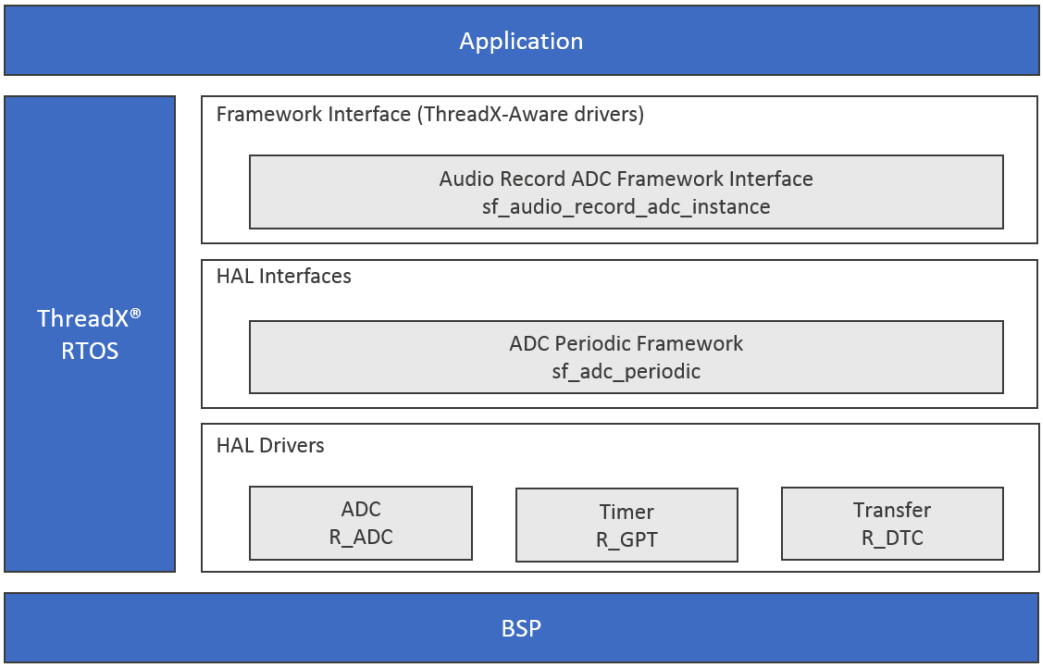


Figure 12.6 Audio Record ADC Framework

12.8 Audio Record I²S Framework

The Audio Record I²S Framework module provides high-level APIs for audio recording applications and uses the I2S interface. The Audio Record I2S Framework module uses the SSI, GPT and DTC peripherals on the Synergy MCU. A user-defined callback can be created to indicate that new samples are stored in the user buffer.

Audio Record I²S Framework supports the following features:

- Thread safe
- Records data in 8 or 16-bit PCM
- Periodic callback function when new samples are available
- Configurable number of samples (sample count) per callback

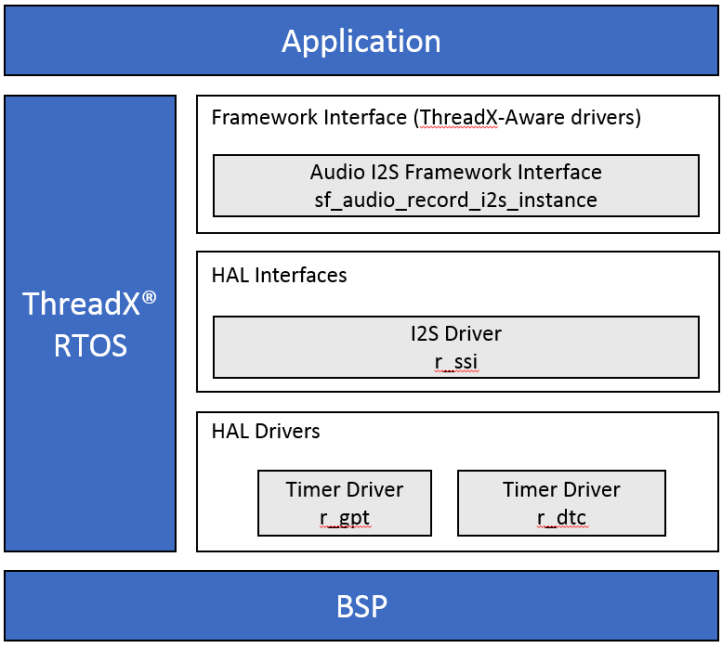


Figure 12.7 Audio Record I²S Framework

12.9 Bluetooth Low Energy (BLE) Framework

Bluetooth® Low Energy (BLE), sometimes referred to as Bluetooth Smart, is a light-weight subset of classic Bluetooth and was introduced as part of the Bluetooth 4.0 core specification. In contrast to classic Bluetooth, BLE is designed to provide significantly lower power consumption. This lower power consumption allows Internet of Thing (IoT) devices that have stricter power capacity to transfer small amounts of data between nearby devices.

Application developers access the functionality provided by the BLE stack by using its APIs. The BLE stack APIs provided by different vendors are not standardized. This results in application developers having to update their code when porting to different BLE stacks.

The Synergy BLE Framework handles this issue with a generic interface. The generic interface connects to the underlying BLE stack provided by various vendors and prevents coupling between application and vendor-specific BLE stack code. The use of generic APIs makes application development simpler and portable.

The BLE framework provides a high-level API for BLE applications and is implemented as sf_ble_rl78g1d. The BLE framework uses the Synergy Software Package (SSP) communication framework, enabling the UART driver for communication with the underlying BLE module. The SSP communication framework also integrates the generic BLE profile framework (g_sf_ble_onboard_profile) to provide a uniform interface to BLE profiles. For the RL78G1D BLE hardware module, the generic BLE profiles are implemented by the BLE module firmware.

The Synergy BLE framework supports the following features:

- ThreadX® RTOS Aware and thread safe
- Bluetooth v4.2 compliant framework.
- Generic Access Profile (GAP) Features
  - User-defined advertising data
  - Security modes 1 and 2
  - Peripheral and central roles
  - White list supports up to 6 devices
  - Bonding support
- Generic Attribute Profile (GATT) features
  - GATT client and server
- Generic Attribute Profile (GATT) APIs
- Generic Access Profile (GAP) APIs
- Generic On-board Profiles APIs

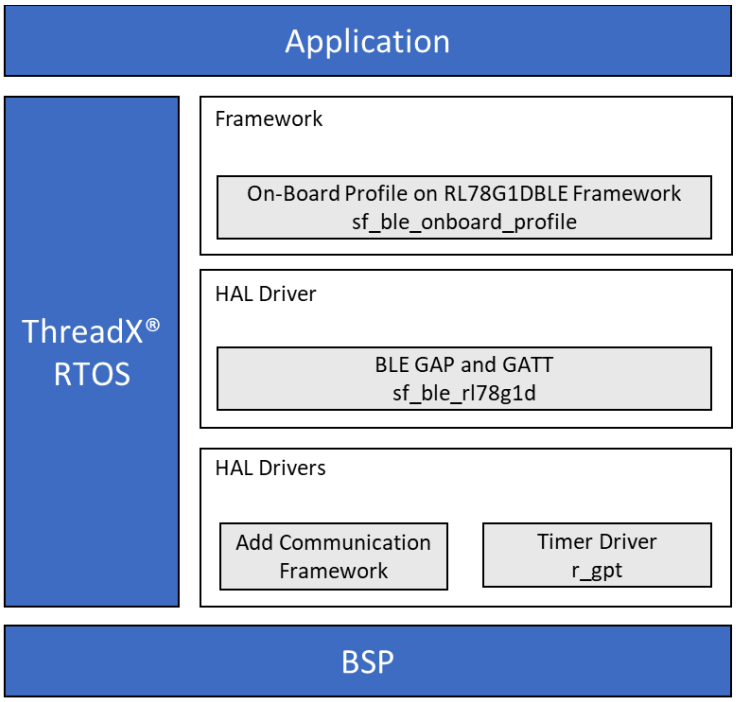


Figure 12.8 Blocks in BLE Framework

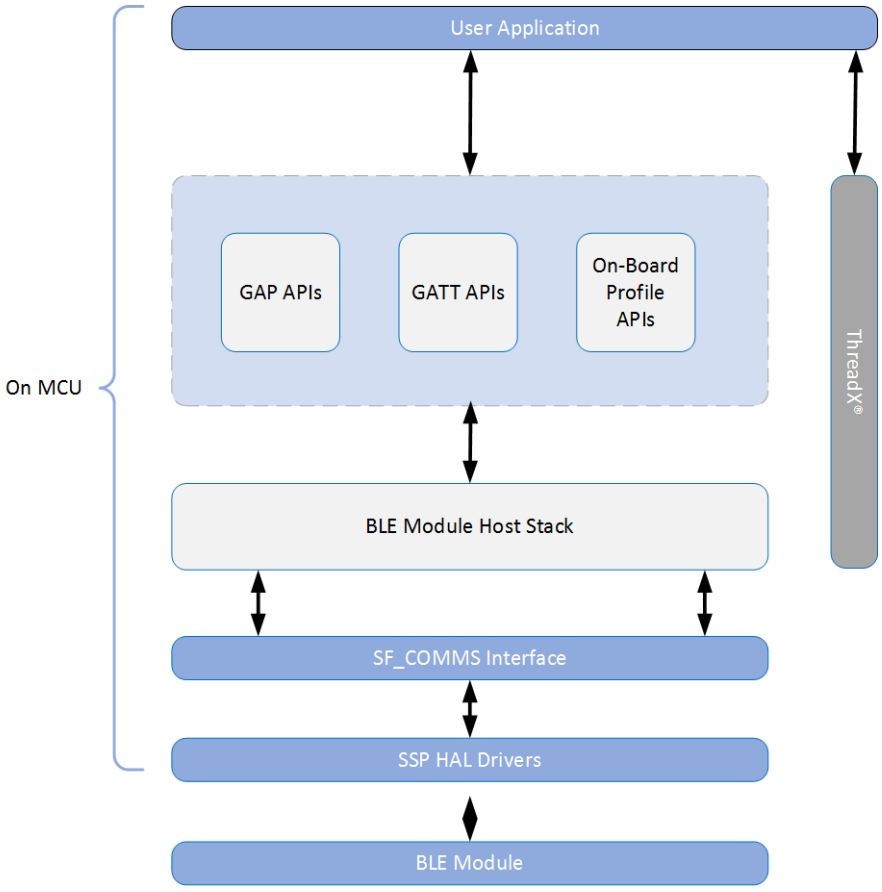


Figure 12.9 BLE Framework Layered Architecture

The BLE Framework consists of the following blocks:

1. **GAP and GATT APIs**
   The BLE Framework provides a generic interface for the application to configure and provision the BLE communication module. The BLE module has various configuration parameters as specified by the Bluetooth Smart standards. It is possible that individual device drivers and/or BLE communication modules may not support configuration of all parameters. At a bare minimum, the provisioning API provides a mechanism to set the operating mode of the BLE interface, security mode, security keys and bonding mode. It also provides an API for the GAP/GATT layers.
2. **On-board Profiles APIs**
   On-board Profiles APIs provide a uniform interface to BLE profiles implemented by the BLE communication module firmware.
3. **BLE Module Host Stack**
   The BLE communication module host stack is provided typically by the module vendor.
4. **BLE Architecture options**
   BLE communication hardware modules are typically available in three different configurations depending on the hardware and software partitioning between Host MCU and BLE communication module.
   A. BLE radio only mode: Link Layer and all the other BLE protocol stack layers plus profiles and application run on the host MCU. Physical Layer runs on BLE chipset.
   B. BLE controller implementation: Link layer runs on BLE chipset, L2CAP and higher BLE protocol layers, plus profiles and application run on the host MCU.
   C. Network controller implementation: Link layer, L2CAP, GATT, GAP layers and Generic Profiles run on the BLE chipset. Optional Profiles and Application run on the host processor.

Note: (A) BLE radio only mode and (B) BLE controller implementation are only shown for illustrative purposes. Only the RL78G1D network controller implementation is supported in the current release.

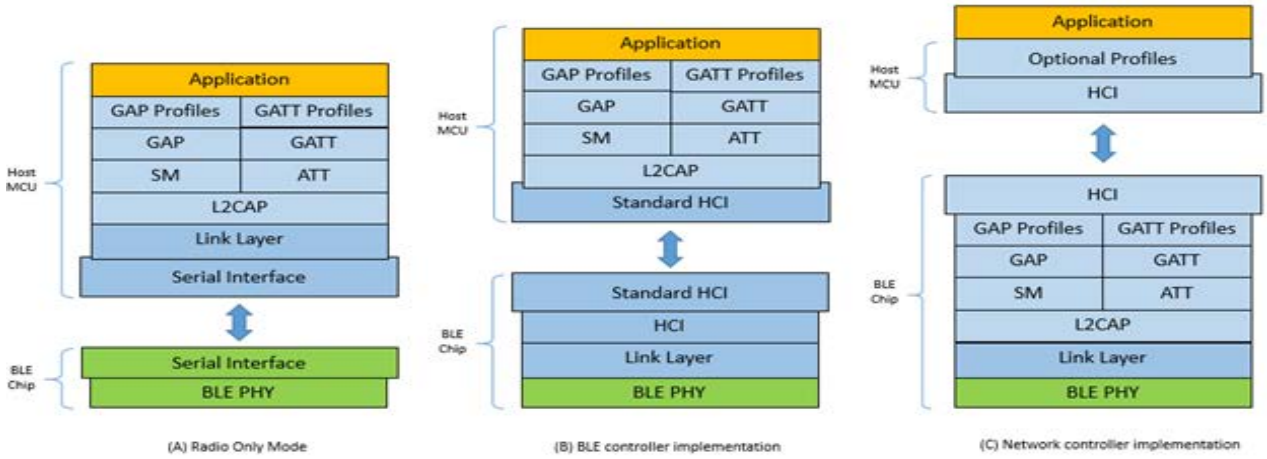


Figure 12.10 Supported BLE Communication Hardware Module Architecture Modes

12.10 Block Media QSPI Framework

The Block Media Framework Module implements APIs for a QSPI channel and supports read, write, and control of the QSPI Flash memory peripheral through the r_qspi driver. The driver has all the functionality needed to interface with a file system through a block media interface.

Block Media QSPI Framework Module supports the following features

- •Supports the QSPI channel interface for a QSPI flash memory device.
- •Supports a file system on QSPI flash memory.

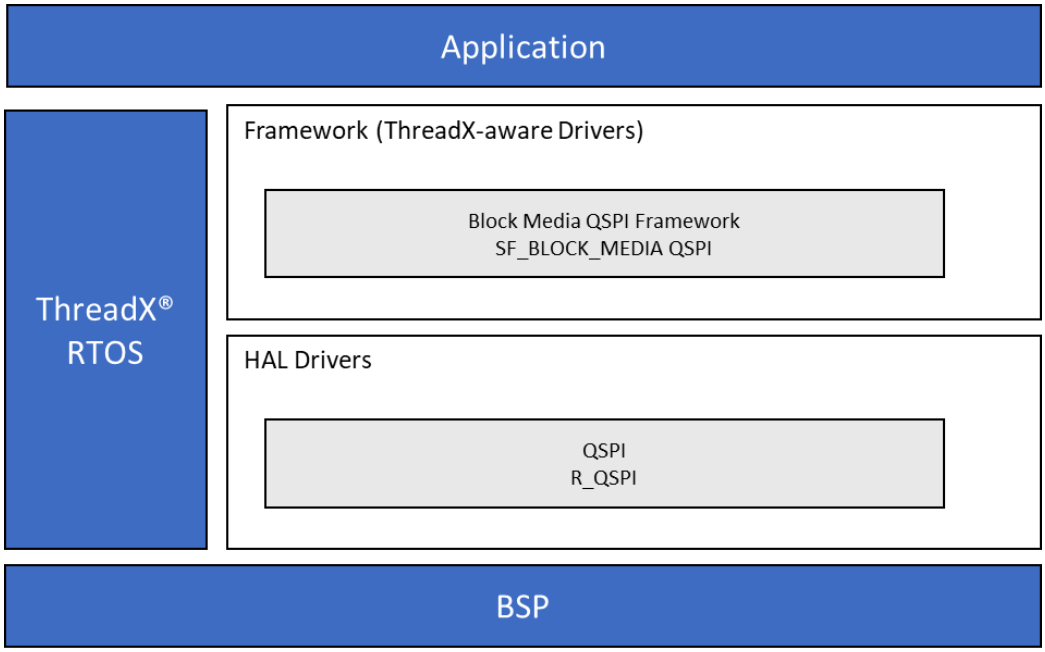


Figure 12.11 Block Media QSPI Framework

12.11 Block Media RAM Framework

The Block Media Framework Module implements a file system on RAM for read, write, and control of the read/write region of RAM memory. The framework has all the functionality needed to interface with a file system through a block media interface.

The Block Media RAM Framework supports the following features:

- Enable FileX to be run on linear memory-mapped devices.
- Temporary and fast storage of data on RAM

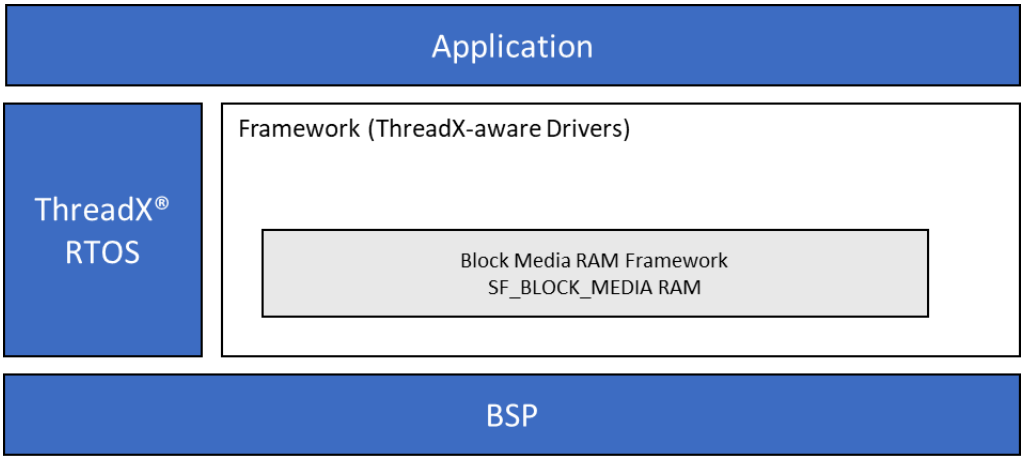


Figure 12.12 Block Media RAM Framework

12.12 Block Media SDMMC Framework

The Block Media SDMMC Framework module is covered in the FileX Port Block Media section. For information, see section 12.28, Synergy FileX® Port Block Media Interface Framework.

12.13 Capacitive Touch Sensing Unit (CTSU) Button Framework

The Capacitive Touch Button Framework module provides high-level ThreadX aware APIs for Capacitive Touch Button applications and is implemented on sf_touch_ctsu using the ThreadX RTOS. The Capacitive Touch Button Framework module uses the CTSU peripheral on the Synergy MCU. A user-defined callback can be created to respond to each button in the order in which they are present.

The Capacitive Touch Button Framework module is used to interpret the CTSU data for all the buttons that are present in the system. The Capacitive Touch Button Framework Module supports the following features:

- Works with the Capacitive Touch Workbench for Renesas Synergy (CTW) tool, which generates configuration data.
- Supports S124, S128, S3A3, S3A7, S5D9 and S7G2
- Provides a callback function to events.
  - Performs de-bouncing
  - Supports multiple types of events including Press, Release, and LongTouch
  - Calls the callback for each button in the order in which they are present in the button configuration table.

The Capacitive Touch Button Framework requires the Capacitive Touch Framework. In most cases, all the needed configuration information is automatically added to the modules.

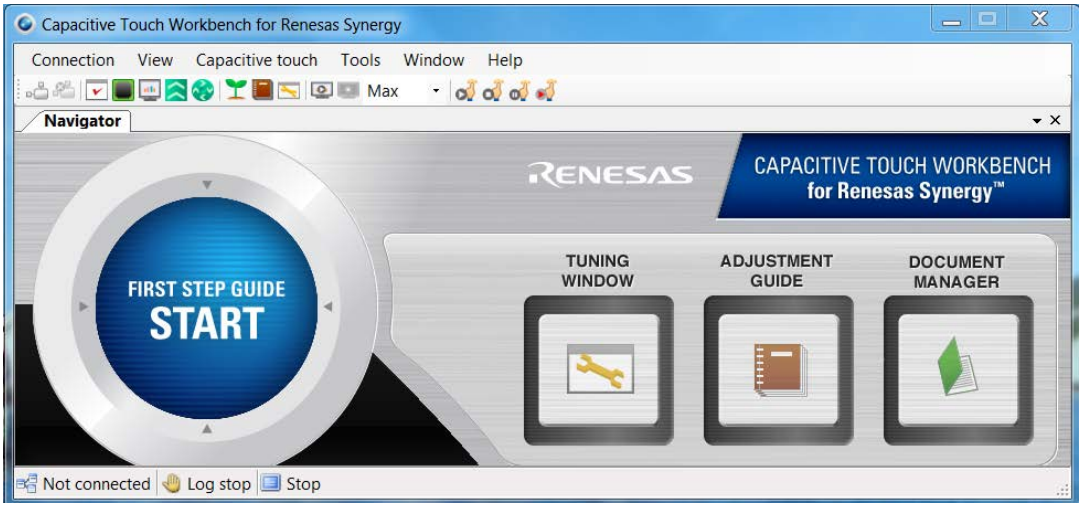


Figure 12.13 Tuning Tool for Capacitive Touch

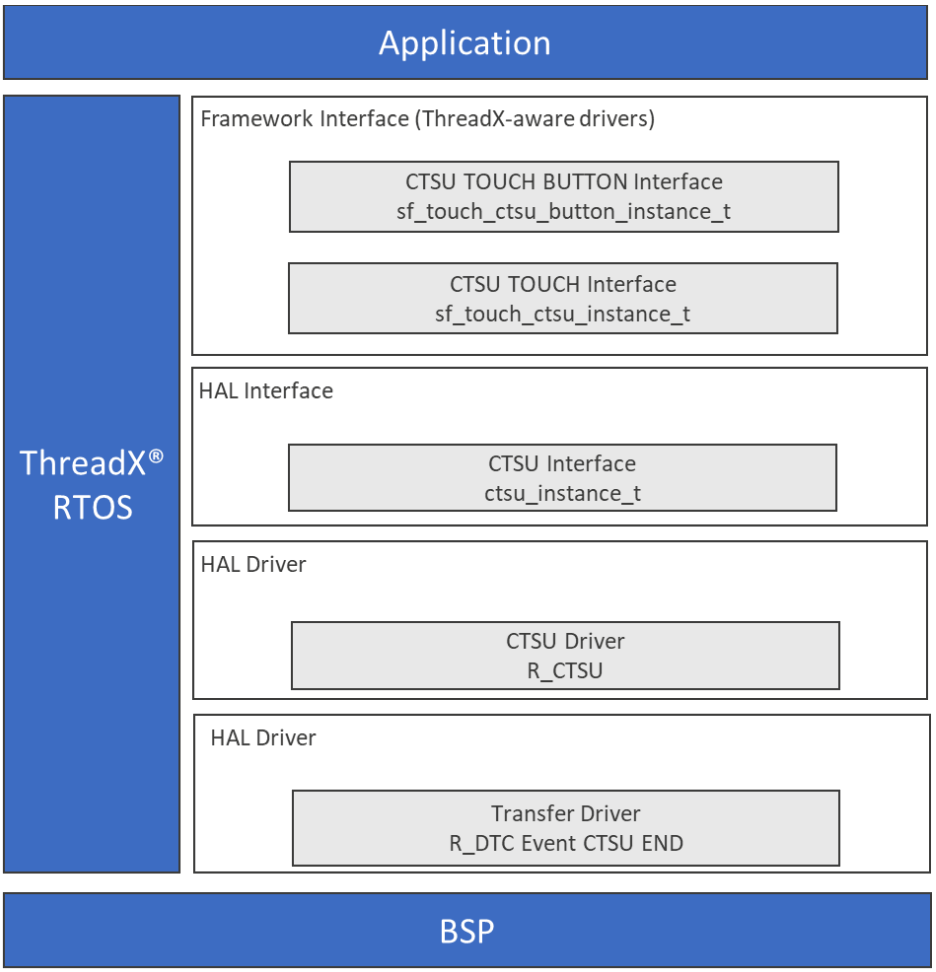


Figure 12.14 Capacitive Touch Sensing Unit Button Framework

12.14 Capacitive Touch Sensing Unit (CTSU) Framework

The Capacitive Touch Framework module provides high-level APIs for Capacitive Touch applications using the ThreadX RTOS and is implemented on sf_touch_ctsu. The Capacitive Touch Framework module essentially works as a framework version of the r_ctsu HAL driver, with an extra feature — it drives the scans autonomously. This framework can be used with other frameworks, like buttons and sliders. It can also be used by applications that only need the binary data on the status of each channel from the CTSU peripheral.

Capacitive Touch Framework Module supports the following features:

The CTSU Framework Interface creates a private thread which drives a hardware scan of a capacitive touch panel and updates the panel at a periodic rate specified in the configuration. It provides the following key functions:

- Allows more than one application layer to register a callback. This framework allows multiple widgets like sliders and buttons to use this layer.
- Autonomously drives the periodic CTSU scan and update process at the rate specified by the user.

This CTSU framework is designed to be used together with the configuration data generated by the Capacitive Touch Workbench for Renesas Synergy™ (CTW for Synergy) tool.

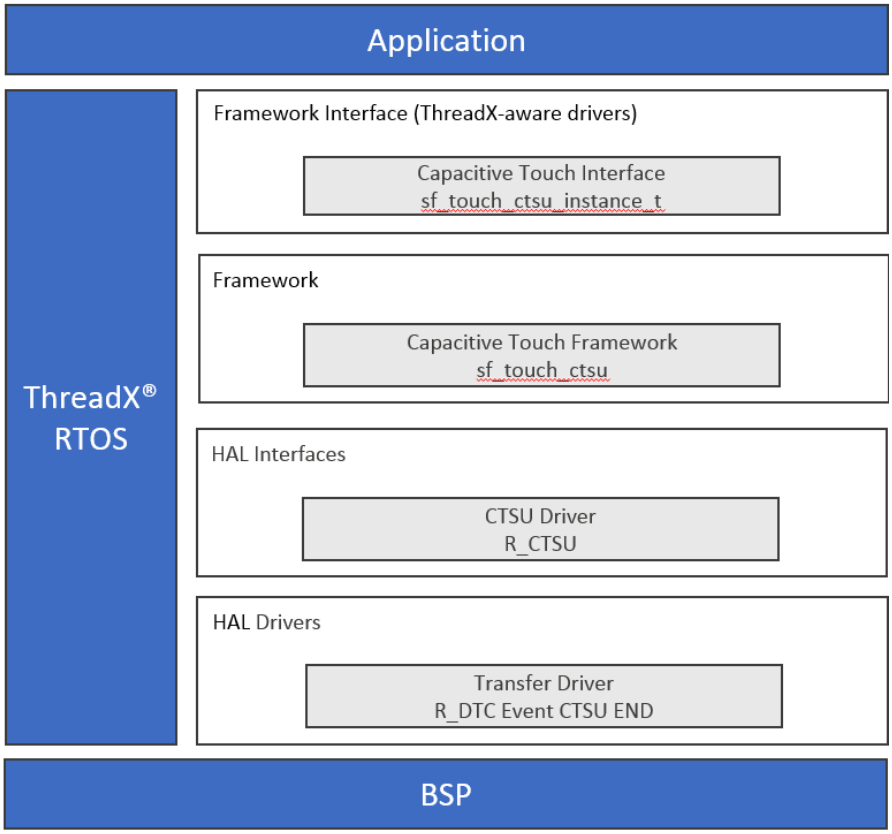


Figure 12.15 Capacitive Touch Sensing Unit Framework

12.15 Capacitive Touch Sensing Unit (CTSU) Slider Framework

The Capacitive Touch Slider Framework module provides high-level ThreadX-aware APIs for Capacitive Touch slider and wheel applications and is implemented on sf_touch_ctsu_slider. The Capacitive Touch Slider Framework module uses the CTSU on the Synergy MCU. This framework is designed to be used with the configuration data generated by the Capacitive Touch Workbench (CTW) for Synergy. Sliders, wheels, and channels used, are configured in the CTW tool. A user-defined callback can be created to process data when it is available from the hardware after a scan.

The Capacitive Touch Slider Framework module is used to interpret the CTSU data for all the slider configurations initialized by the system. Capacitive Touch Slider Framework Module supports the following features:

- Support for slider and wheel
- Support for multiple instances of sliders and wheels
- Callbacks that are used to simplify touch processing
  - Use configuration data generated from CTW for Synergy
  - When a state changes a callback is generated
  - Callbacks are associated with each slider and include the event and position
  - Callbacks are called in the order they appear in the configuration table
- Supports multi-touch detection (can be optionally disabled at build-time)

The framework is designed to be used together with configuration data generated by the CTW for Synergy tool.

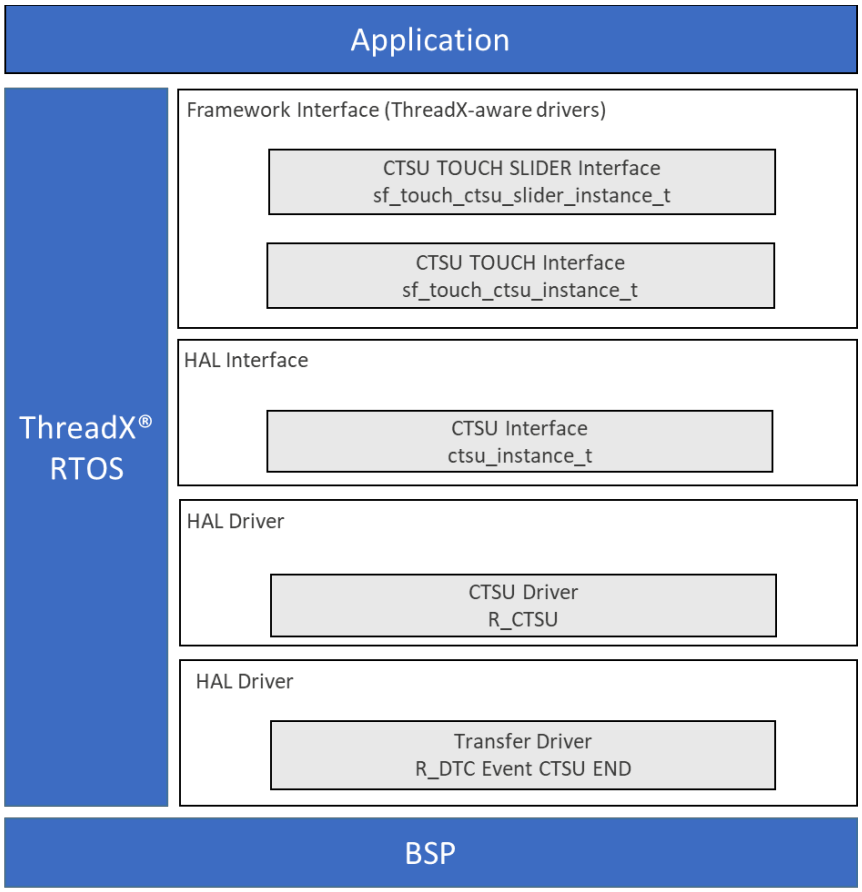


Figure 12.16 Capacitive Touch Sensing Unit Slider Framework

12.16 Cellular Framework

The Cellular Framework module provides a high-level application layer interface for the cellular modem integration in the SSP and a common interface that enables applications to interface with the Cellular modems from various vendors.

The framework provisions and configures the cellular modem for communicating with the cellular network for data communication. A Console Framework is used to communicate with Cellular modems over a serial interface using AT commands. The framework creates the serial data pipe over a serial interface for data communication, leveraging the PPP WAN protocol provided by NetX™. Data communication using TCP/IP can be established over this Wide Area Network (WAN) link using NetX application protocols, sockets, or IoT protocols such as MQTT.

The Cellular Framework also provides the framework level Socket APIs to communicate with the TCP/IP stack present on-chip (inside cellular hardware module) in certain cellular hardware modules, and with the TCP/IP link for the network using socket APIs.

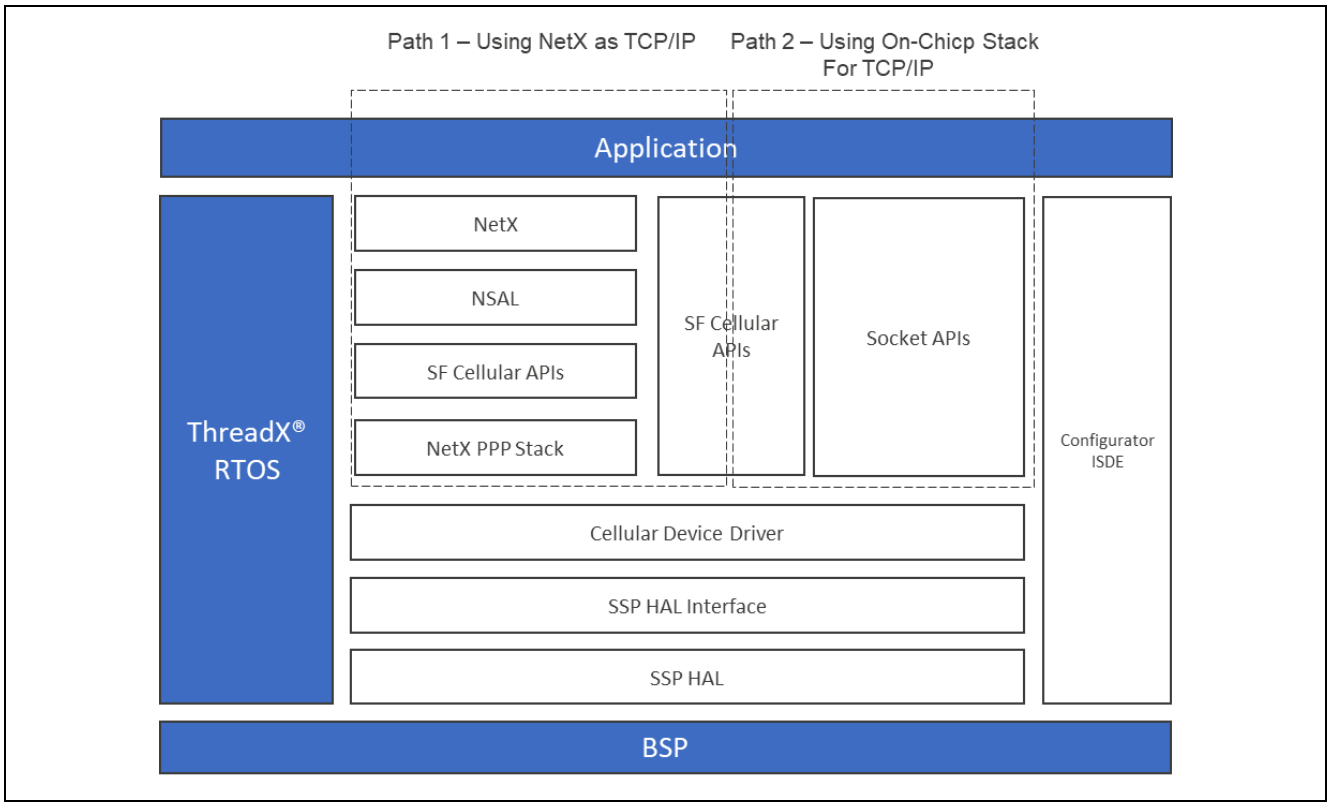


Figure 12.17 Cellular framework module organization and interface layers

The Synergy Cellular Framework consists of the following logical blocks:

- Synergy Cellular Framework Application Interface
- Network Stack Abstraction Layer (NSAL) for NetX TCP/IP stack
- Cellular Device Driver
- BSD Socket compatible APIs for interfacing with Cellular hardware module that supports on-chip networking stack
- Synergy Software Package (SSP) HAL Interface

SF Cellular Framework APIs

The application uses the SF Cellular Framework API interface to communicate with the cellular modem from various vendors without chaning the application. The module specific implementation provides the AT command set and overrides the generic driver by implementing the required functionality. For instance, the Generic driver implements the open, close, provision, etc. APIs. The AT commands required by these APIs is provided in the module specific driver. Should the module specific driver have to implement the open API differently, it can do so by overriding the open implementation with its own.

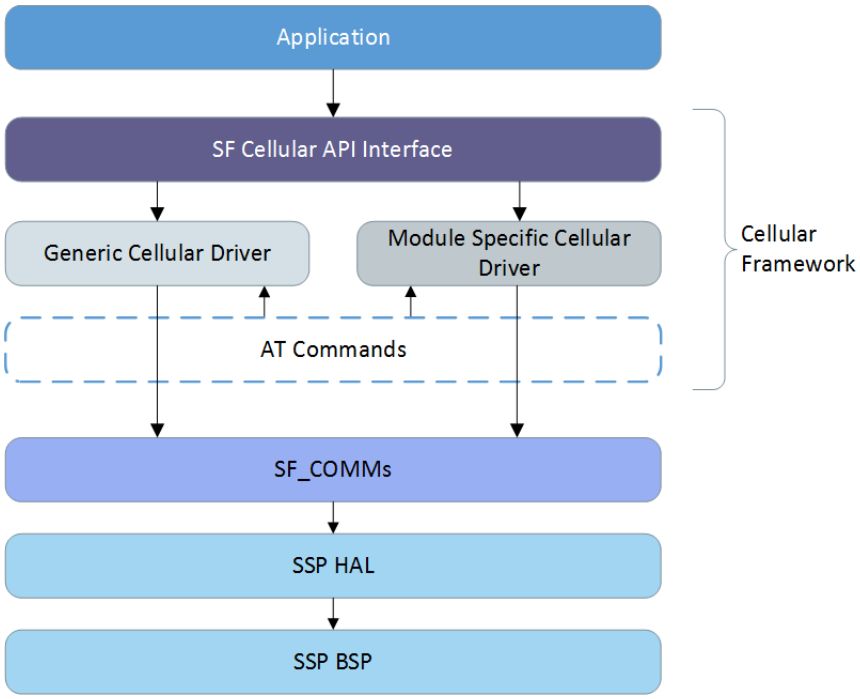


Figure 12.18 Synergy Cellular Framework APIs

Network Stack Abstraction Layer

The Cellular framework provides a network stack abstraction layer (NSAL). NSAL creates a PPP channel used for IP communication. It also handles authentication methods (such as PAP/CHAP) used by PPP. Although these authentication mechanisms are optional. NSAL makes use of framework APIs to send/receive data from Cellular module. NSAL allows the cellular device driver to be re-used without any changes specific to network stack. Adding support for a new network stack requires implementing the appropriate NSAL.

Cellular device driver

Cellular Framework uses the AT command set to interact with the Cellular modem using the serial driver. The serial interface used to interact with the modem is UART.

Socket APIs

The Socket API provides an interface to the application to enable the BSD Socket APIs to use on-chip networking stack present on the Cellular module. Usage requires the Cellular module/driver to support an on-chip networking stack and socket interface. When the application uses these APIs, it uses the on-chip networking stack present on Cellular module and does not use the NSAL. The application does not use the Networking stack running on the Synergy MCU.

SSP HAL Interface

The following figure shows the interface for SSP HAL components used by the Cellular device driver for lower level communication with Synergy MCU. This implementation is specific to Cellular device driver. Cellular modules make use of different HAL components such as UART, ICU, IOPORT, and so on.

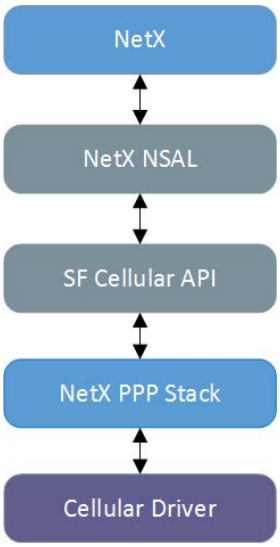


Figure 12.19 SSP HAL Interface

The Cellular Framework Module supports the following features:

- Supports connectivity using:
  - CAT1, CAT3 and CAT M1 Cellular Modems
  - BSD Socket interface for On-Chip stack present on the Cellular Module
  - NetX Stack on Synergy MCU (Host) using NSAL interface
- Supports a common a set of APIs, to interface to the networking stack and generic interface for the different Cellular hardware modules.
- Using generic APIs and abstraction, applications developed for the cellular hardware module can be easily migrated to work with another cellular hardware module.
- Supported Cellular modems:
  - NimbeLink CAT3 (NL-SW-LTE-TSVG) Verizon-US
  - NimbeLink CAT3 (NL-SW-LTE-TEUG) India and Europe
  - NimbeLink CAT1 (NL-SW-LTE-GELS3-B and NL-SW-LTE-GELS3-C) Verizon-US
  - Quectel CAT M1-BG96

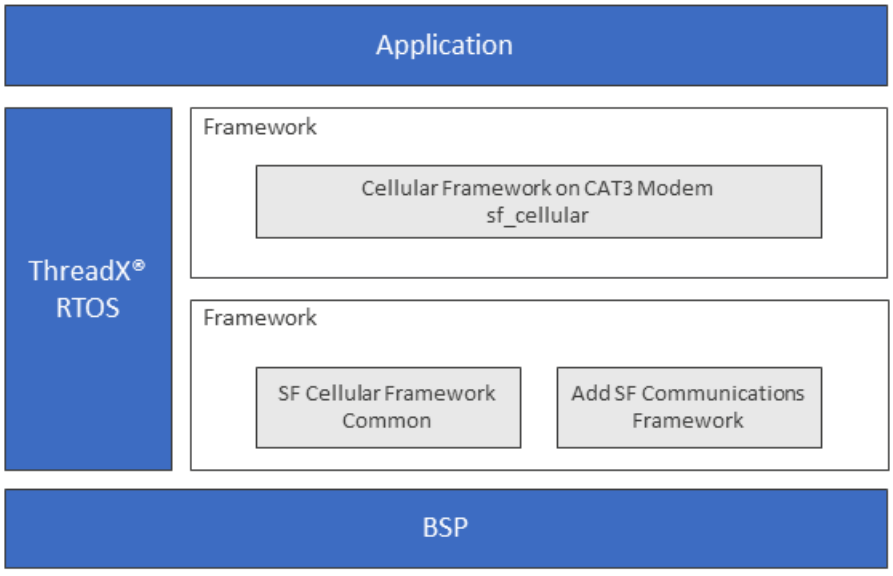


Figure 12.20 Cellular Framework on CAT3 Modem Module Organization, Options, and Stack Implementations

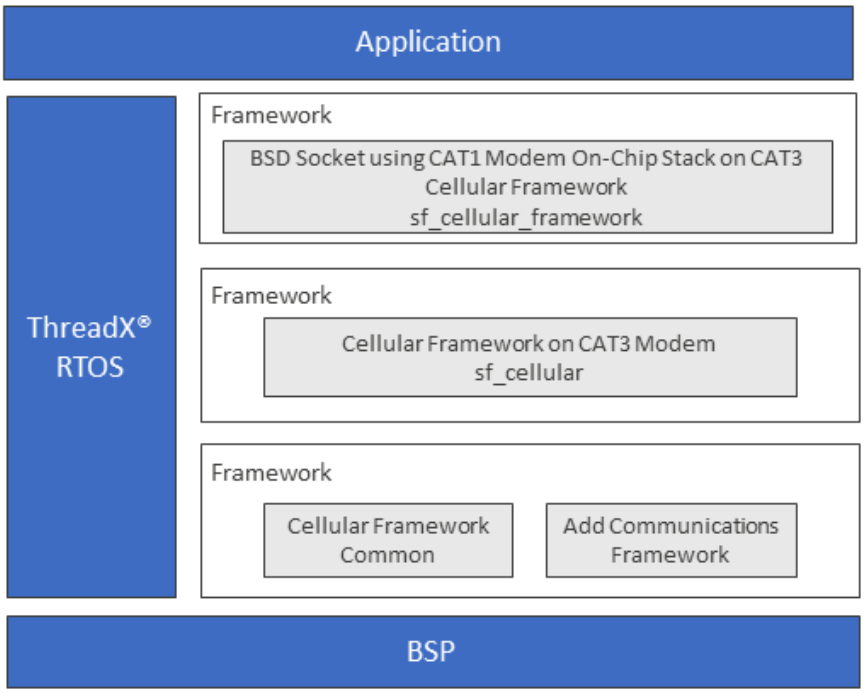


Figure 12.21 BSD Socket using CAT3 On-Chip Stack on CAT3 Cellular Framework Module

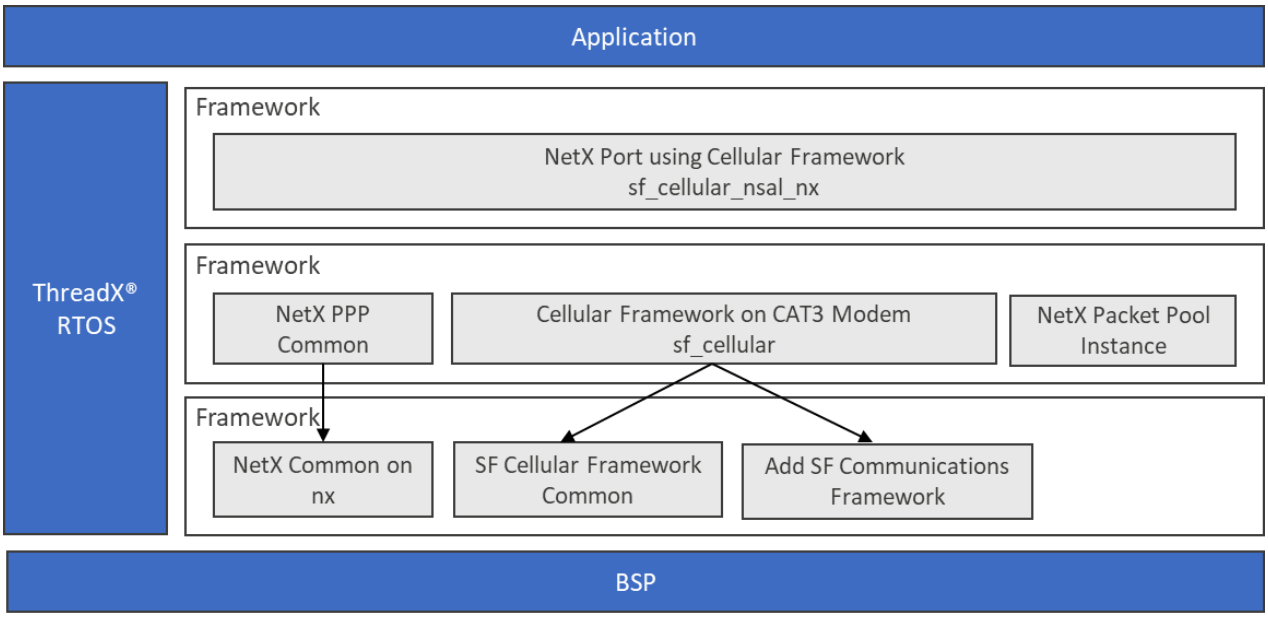


Figure 12.22 NetX Port using Cellular Framework Module Organization, Options and Stack Implementations

12.17 Communications Framework on NetX Telnet

The NetX Telnet Communications Framework module provides high-level APIs for telnet communications applications and is implemented on sf_comms_telnet. It uses the Ethernet peripheral on the Synergy MCU.

The NetX Telnet Communications Framework Module supports the following features:

- High level connectivity is supported on Ethernet but is easily changeable to UART and USB connectivity without API modification
- Supports channel locking for exclusive access
- Thread aware implementation uses mutex and event flags internally

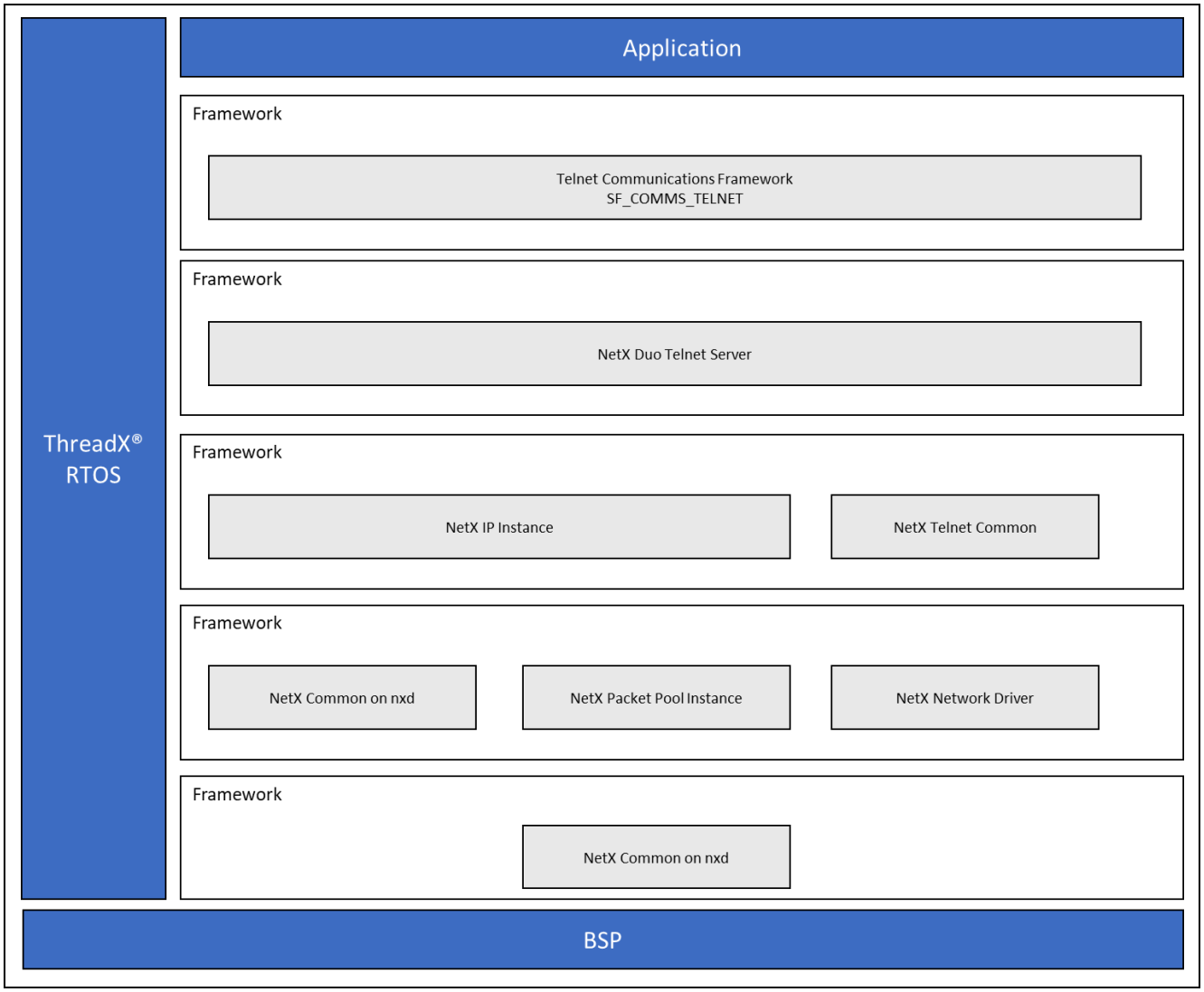


Figure 12.23 Communications Framework on NetX Telnet

12.18 Communications Framework on USBX

The Communications Framework on USBX™ provides high-level APIs for Communications Framework applications that is implemented on sf_el_ux_comms, the USBX Device Class CDC-ACM (Communications Device Class-Abstract Control Model). The Communications Framework uses the USB peripheral on the Synergy MCU device. This module has been deprecated in SSP v1.2.0 and later releases, however the module is being included in SSP releases to maintain backward compatibility.

The Communications Framework on USBX Module supports the following features:

- Implements Express Logic USBX in SSP- support USBX APIs
- Supports the Port Device Controller Driver (DCD) for the USBHS and USBFS peripherals
- Supports the Port Host Controller Driver (HCD) for the USBHS and USBFS peripherals
- Supports transfer module operation (optional)

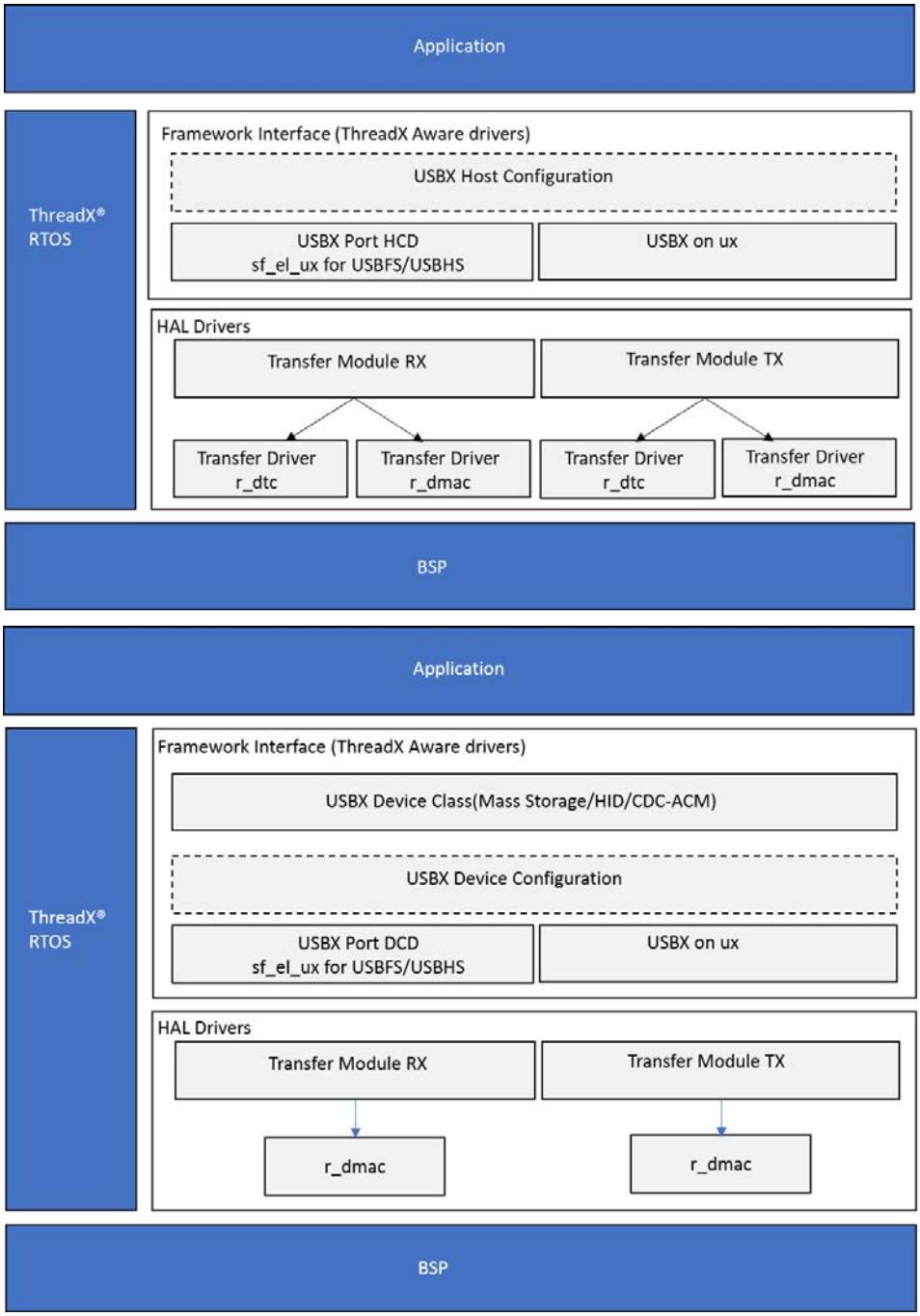


Figure 12.24 Communications Framework on USBX Module

12.19 Console Framework

The Console Framework provides high-level APIs implemented on sf_console for a menu-driven console command line interface (CLI) using the ThreadX RTOS. The Console Framework module uses a communications interface which connects to a hardware option for either UART, USB, or Ethernet Telnet connectivity. The Console Framework module has a user-defined menu of commands and various APIs to present a prompt, identify and issue a callback for menu commands, and read, write, and parse input strings.

Console Framework Module supports the following features:

- Creation of a menu-based command-line interface
- Submenus and navigation through multiple menus in a single call
- Menu navigation to go up to the parent menu or back to the root
- A help menu for each menu
- Writing NULL terminated strings and reading until return character is received
- An API to help parse arguments to the command line
- Case-insensitive inputs

The Console Framework module organization, as depicted in the thread stack window in the SSP configurator, is shown in the following figure. Each implementation choice, Ethernet, UART, and USB has its own lower-level modules that are added automatically based on the developer's implementation choice. In most cases, all the needed configuration information is automatically added to the modules leaving the developer with just a few important configuration settings that need to be selected.

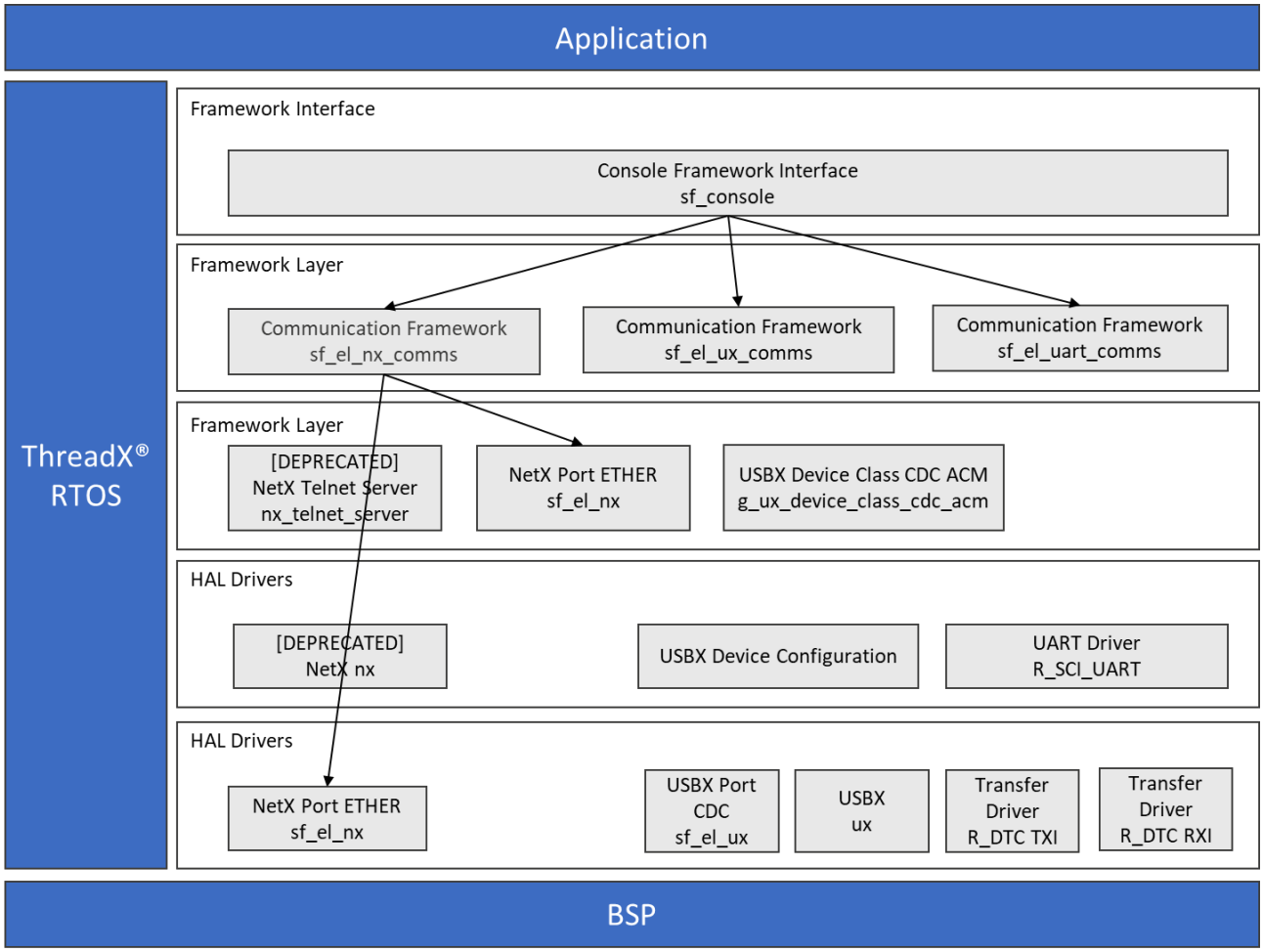


Figure 12.25 Console Framework

12.20 Crypto Framework

The Crypto Framework layer is composed of multiple Crypto modules providing varied cryptographic services. The Crypto Framework modules include SF_CRYPTO_TRNG for true random number generation, sf_crypto_hash for message digest generation and sf_crypto_key for RSA and AES Key Generation services. See section 14.38, Secure Cryptographic Engine (SCE) module and Table 14.1 for cryptographic functions supported on target MCUs.

The Crypto Framework module supports the following features:

- SF CRYPTO Framework
  - The SF CRYPTO Framework opens the underlying Secure Crypto Engine.
  - Provides services to access shared resources for other Crypto Framework modules SF_CRYPTO_TRNG, SF_CRYPTO_HASH , SF_CRYPTO_KEY, SF_CRYPTO_CIPHER, SF_CRYPTO_SIGNATURE, and SF_CRYPTO_KEY_INSTALLATION.
- SF CRYPTO TRNG Framework
  - The SF CRYPTO TRNG Framework module uses the TRNG HAL interfaces to the underlying Secure Crypto Engine.
  - Access to shared resources through SF CRYPTO Framework module.
- SF CRYPTO HASH Framework
  - The SF CRYPTO HASH Framework utilizes the underlying Secure Crypto Engine to provide HASH services.
  - This module provides enhancements over the HAL driver such as:
  - Initializing the HASH value
    - Processing chunks of data through hashUpdate API before finalizing the hash operation.
    - Formatting the final block for the final HASH operation.
- SF CRYPTO KEY Framework Module Features
  - RSA 2048-bit, 1024-bit plain text/raw keys.
  - RSA 2048-bit, 1024-bit standard format wrapped private keys (public keys in plain).
  - AES 128-bit, 192-bit and 256-bit wrapped keys for ECB, CBC, CTR and GCM chaining modes.
  - AES 128-bit and 256-bit wrapped keys for XTS chaining mode.
  - ECC 192-bit, 256-bit wrapped keys.
- SF CRYPTO SIGNATURE Framework Module Features
  - This module currently supports signature generation and signature-verification for RSA algorithm. There is support for both 1024-bits and 2048-bits key lengths for standard format plain-text, standard format wrapped private key and CRT plain-text keys.
  - RSASSA-PKCS1 v1.5 is the supported signature scheme. The input message can be passed as raw message to be signed/verified or can be PKCS1 v1.5 encoded and padded before sign/verify.
  - SHA1, SHA224, SHA256 are the supported hashing algorithms for PKCS1 v1.5 encoding /padding schemes.
  - This module allows signature generation/verification on data which is smaller than a block size.
  - If all of the data is not available at once, update APIs can be used to accumulate the incoming chunks of data and finally sign/verify the message only when all the message is gathered.
  - Allows switching between sign and verify operations with less expensive API calls which are not involved in allocating and deallocating memory.
- SF CRYPTO CIPHER Framework Module Features
  - This module currently supports encryption and decryption for AES and RSA algorithms.
  - This module allows encryption/ decryption of data when available in chunks through the cipherUpdate()API and final() when the last chunk /all the data is gathered.
  - Once the module is opened for a specific key type and key size, the encrypt and decrypt modes can be switched by calling the cipherInit() API.
- SF CRYPTO KEY INSTALLATION Framework Module Features
  - The SF CRYPTO KEY INSTALLATION Framework module uses the KEY INSTALLATION HAL interfaces to the underlying Secure Crypto Engine.
  - Access to shared resources through SF CRYPTO Framework module.
  - This module supports key installation for the following keys:
    - RSA:
      - 1024-bit and 2048-bit plain text keys.
    - AES:
      - 128-bit, 192-bit and 256-bit plain text keys for ECB, CBC, CTR and GCM chaining modes.

— 128-bit and 256-bit plain text keys for XTS chaining mode.

- ECC:
  — 192-bit and 256-bit plain text keys.
  — The user and install keys must be provided to the key installation API in the specified format.
  — Upon installation the key installation service returns a wrapped key to the caller for any future usage of installed key on that device.

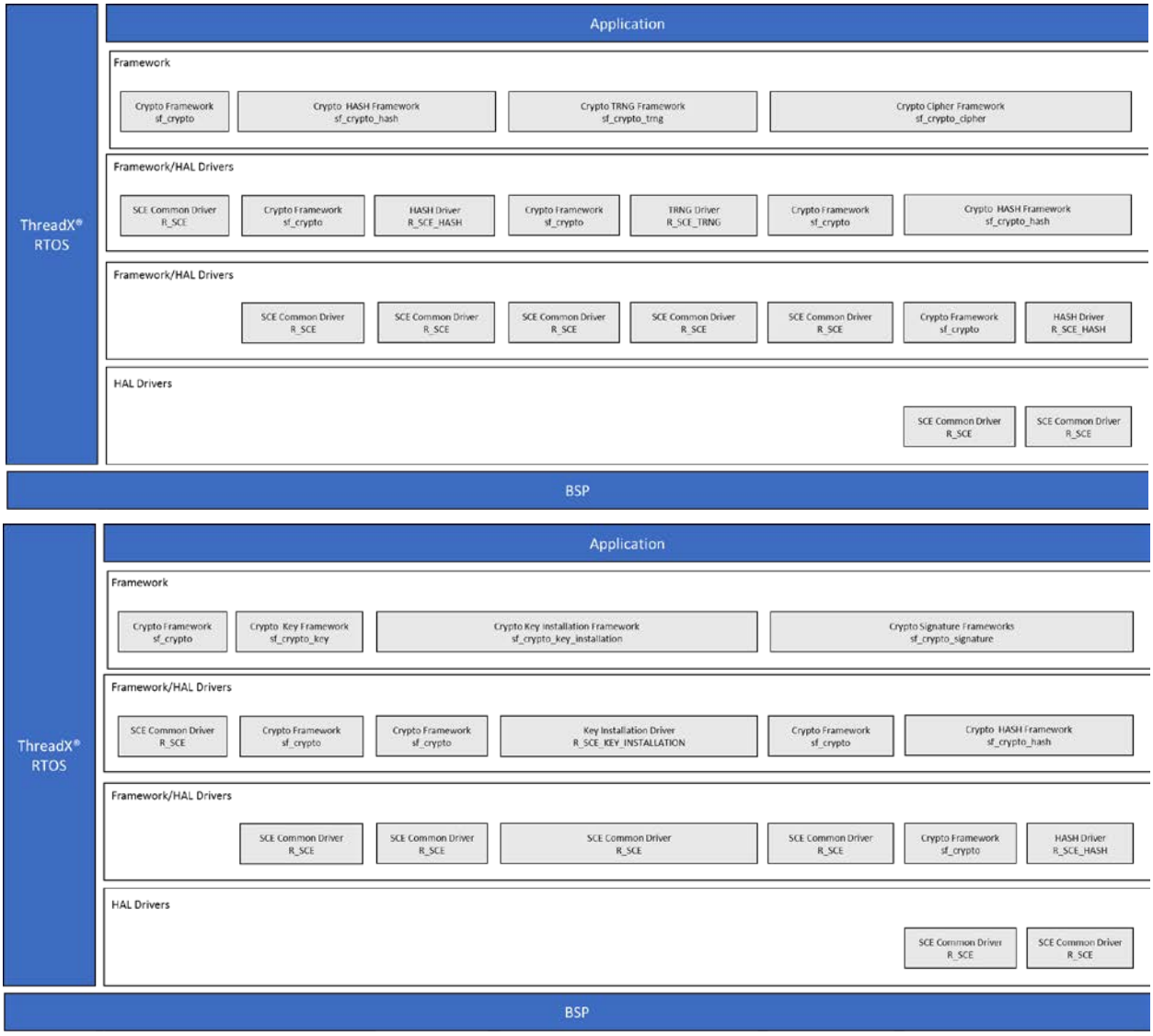


Figure 12.26 Crypto Framework Module Organization, Options, and Stack Implementations

12.21 External IRQ Framework

The External IRQ Framework provides high-level APIs for applications using the external pin interrupts with the ThreadX RTOS. The Framework is implemented on sf_external_irq and supports the external IRQ pins on the Synergy microcontroller. A callback function (sf_external_irq_callback) is available that will be called from the interrupt service routine (ISR) each time the IRQn triggers.

External IRQ Framework Module supports the following features:

- Responds to external interrupt inputs
- RTOS aware implementation using an internal semaphore for thread synchronization
- Can signal internal threads
- Can trigger transfers via the Event Link Controller (ELC)
- Uses the port pins available on Synergy MCUs
- Pins may differ between MCUs, see the applicable MCU User's Manual for specifics
- Supports several hardware features such as
  - Channel selection
  - Trigger conditions
  - Digital filtering
  - Auto start

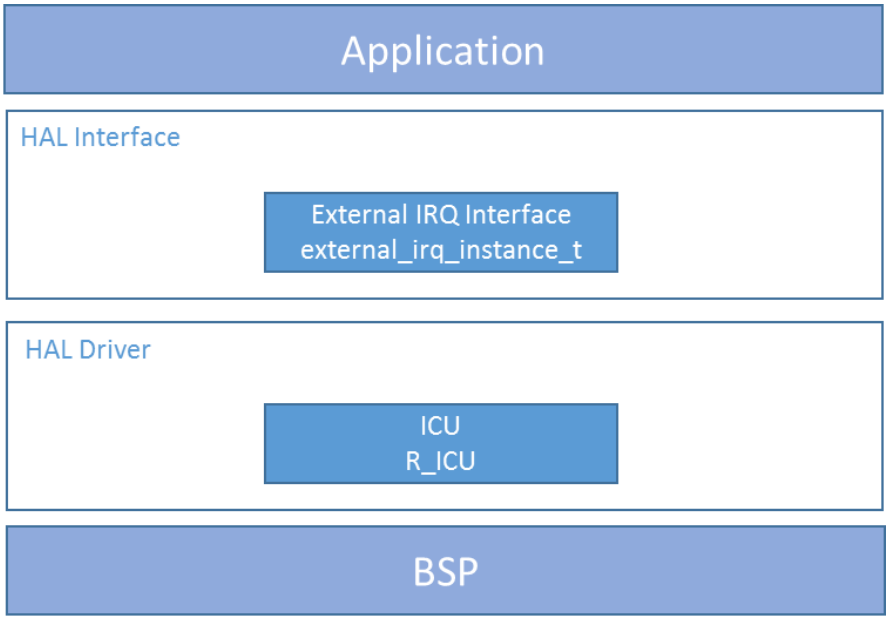


Figure 12.27 External Interrupt Framework

12.22 I²C Framework

The I²C Framework module is a ThreadX®-aware high-level API for I²C Framework applications and is implemented on sf_i2c. The I²C HAL module configures the I²C peripheral to enable serial communication to be used by the framework. The I²C Framework module uses the I²C and SCI peripherals on the Synergy MCU.

I²C Framework Module supports the following features:

- ThreadX-aware framework
- Handles integration and synchronization of multiple I²C peripherals on the I²C bus
- Provides a single interface to access both SCI I²C and RIIC drivers
- The I²C framework module configures I²C communication in master mode
- The I²C framework module supports three data rates: 100 kHz, 400 kHz, and 1 MHz
- The I²C framework module supports both 7-bit addressing and 10-bit addressing
- The I²C framework module also provides support for callbacks internally. User defined callback is not used. The callback functions are called with the following events i2c_event_t:
  - Transfer aborted
  - Transmit complete
  - Receive complete
- The callback structure `i2c_callback_args_t` also provides the number of bytes that were sent or received
- Implemented by:
  - Simple I²C on SCI
  - RIIC

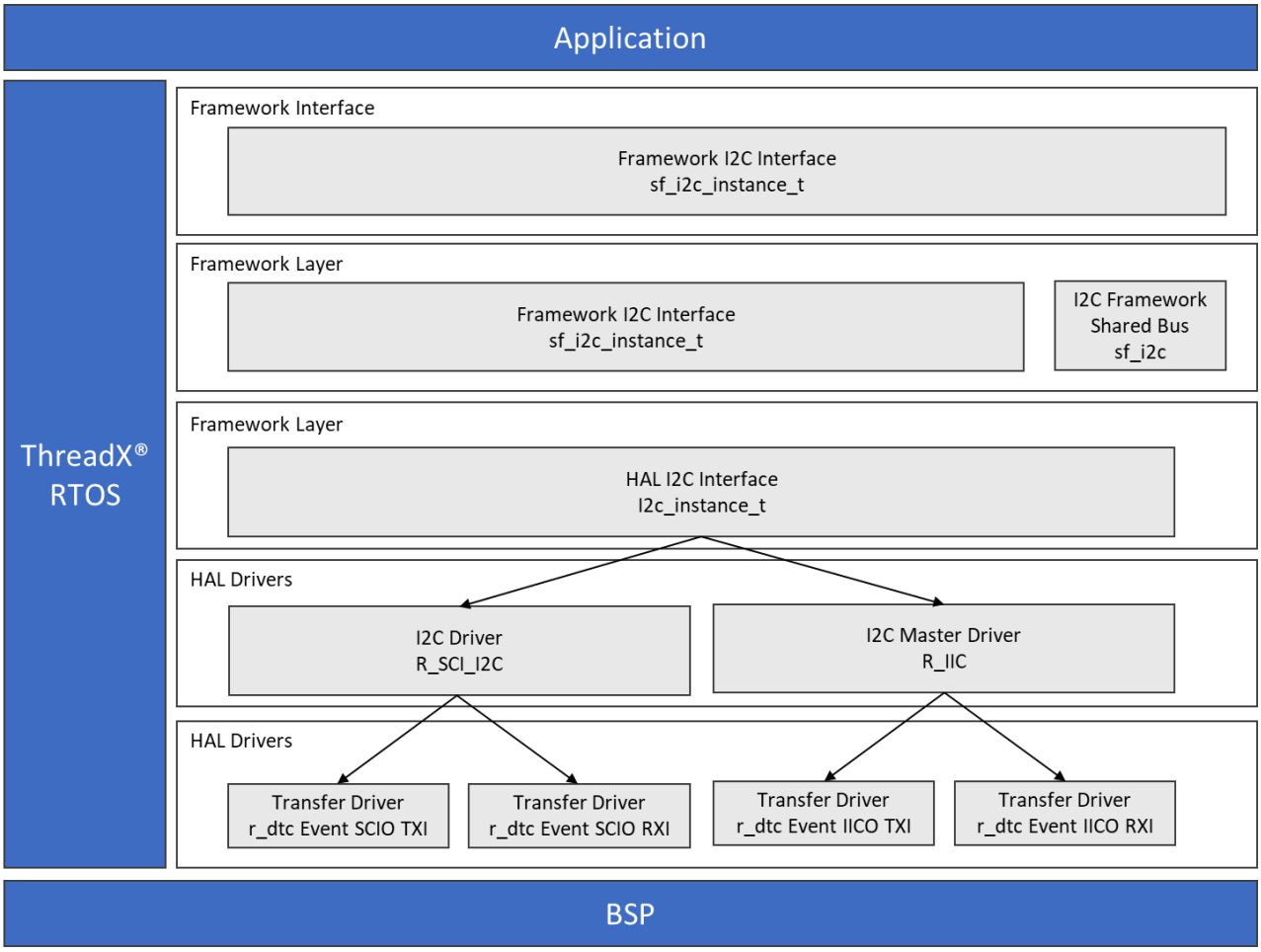


Figure 12.28 I²C Framework

12.23 JPEG Decode Framework

The JPEG Decode HAL module is a generic API for JPEG decode processing implemented on r_jpeg. (The JPEG Decode HAL module supports the JPEG Codec peripheral.) The JPEG Decode Framework Module is a ThreadX®-aware high-level API for JPEG framework module applications and is implemented on sf_jpeg_decode; it provides thread-safe access to the Synergy JPEG hardware on a Synergy MCU Group. A user-defined callback can be created to detect hardware supported events.

JPEG Decode Framework Module supports the following features:

- Provides thread-safe access to the Synergy JPEG hardware.
- Supports JPEG decompression using the JPEG Decode HAL module.
- Supports a polling mode that allows an application to wait for the JPEG Decoder to complete.
- Supports an interrupt mode with user-supplied callback functions.
- Configures parameters such as horizontal and vertical subsample values, horizontal stride, decoded pixel format, input and output data format, and color space.
- Obtains the size of the image prior to decoding it.
- Supports putting coded data in an input buffer and an output buffer to store the decoded image frame.
- Supports streaming coded data into the JPEG Decoder module. This feature allows an application to read a coded JPEG image from a file or from a network without buffering the entire image.
- Configures the number of image lines to decode. This feature enables the application to process the decoded image on the fly without buffering the entire frame.
- Supports the input decoded formats YCbCr444, YcbCr422, YcbCr420 and YcbCr411.
- Supports the output formats ARGB8888 and RGB565.
- Configurable for Motion JPEG (MJPEG) video decoding
- Returns an error when the JPEG image's size, height, and width don't meet the requirements.
- Supports the wait API function to suspend/resume the thread for synchronizing with JPEG hardware-supported events.

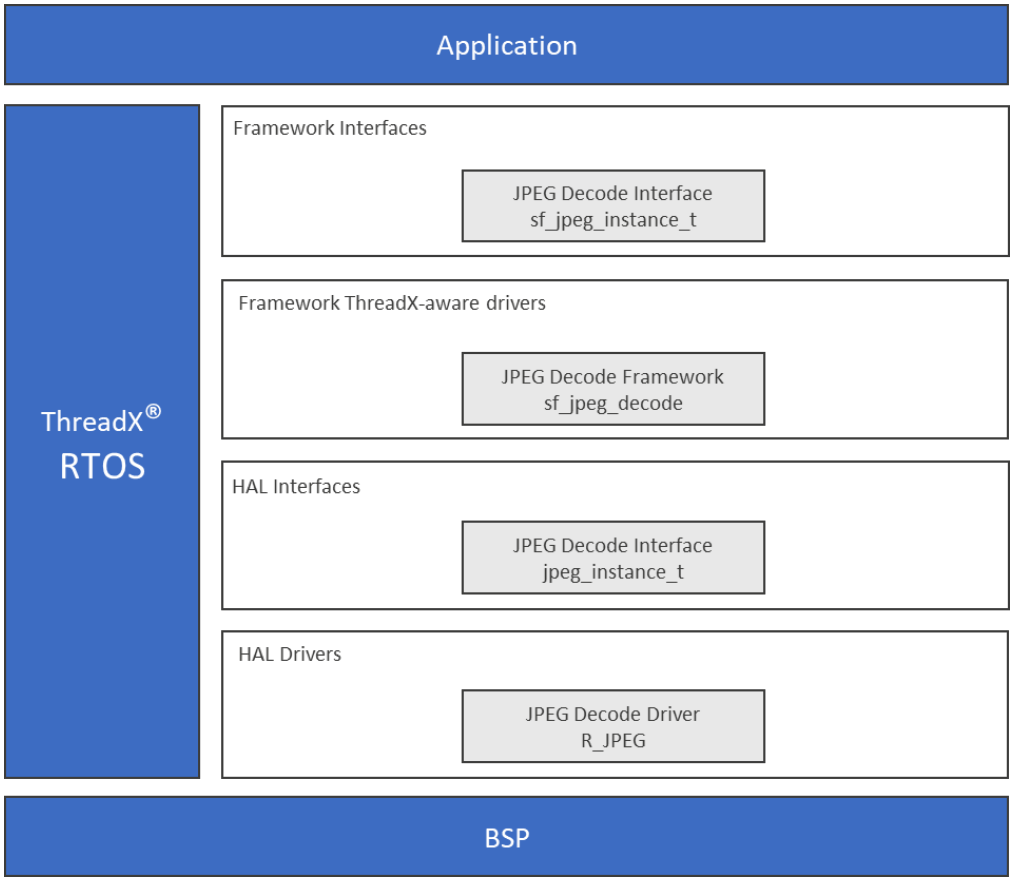


Figure 12.29 JPEG Decode Framework

12.24 Messaging Framework

The Messaging Framework module is implemented on sf_message and provides a lightweight and event-driven framework API for passing messages between threads. The Messaging Framework module allows applications to communicate messages between two or more threads. The framework uses the ThreadX® message-queue primitive for message passing and provides more benefits than the ThreadX RTOS message-queue services alone. The Messaging Framework API is purely a software API and does not access any hardware peripherals. The Messaging Framework callback is used to allow an event-producer thread and a message-subscriber thread to handshake after the message passing is done.

In the SSP configurator you can use the messaging tab to create your own custom event classes, events, and subscribers for the Messaging Framework module, or use it to customize preconfigured events, such as the touch event used by the Touch Panel Framework module.

The Messaging Framework Module supports the following features:

- **Inter-Thread Communication** — The framework allows application threads, that control disparate devices or manage subsystems, to communicate with each other.
- **Publishing/Subscribe scheme**—The framework design is based on the loosely-coupled messaging paradigm. The design allows multiple threads to listen to an event class. The message producer thread does not need to know who is subscribing to a message for the event class. Subscribers do not need to know who produces the message.
- **Message management** — The framework supports buffer control blocks to manage each message including flags to control the buffer and a callback function pointer for handshaking.
- **Message buffering** — The framework manages buffer allocation and release for messaging. An application can make use of the allocated buffer to write a message and discard the message if it is no longer needed.
- **Synchronous communication** — The framework supports asynchronous messaging by using the ThreadX message-queue but also provides an option to create a handshake between a message producer and a subscriber thread. The handshake is implemented by invoking a user-callback function of the producer thread from a subscriber thread.
- **Message formatting** — The framework provides a predefined common message header. It also provides some typical payload structure templates as examples.
- **Message Priority** — The framework can send a high-priority message so that a subscriber thread can retrieve the message prior to other messages which are located in the message queue.

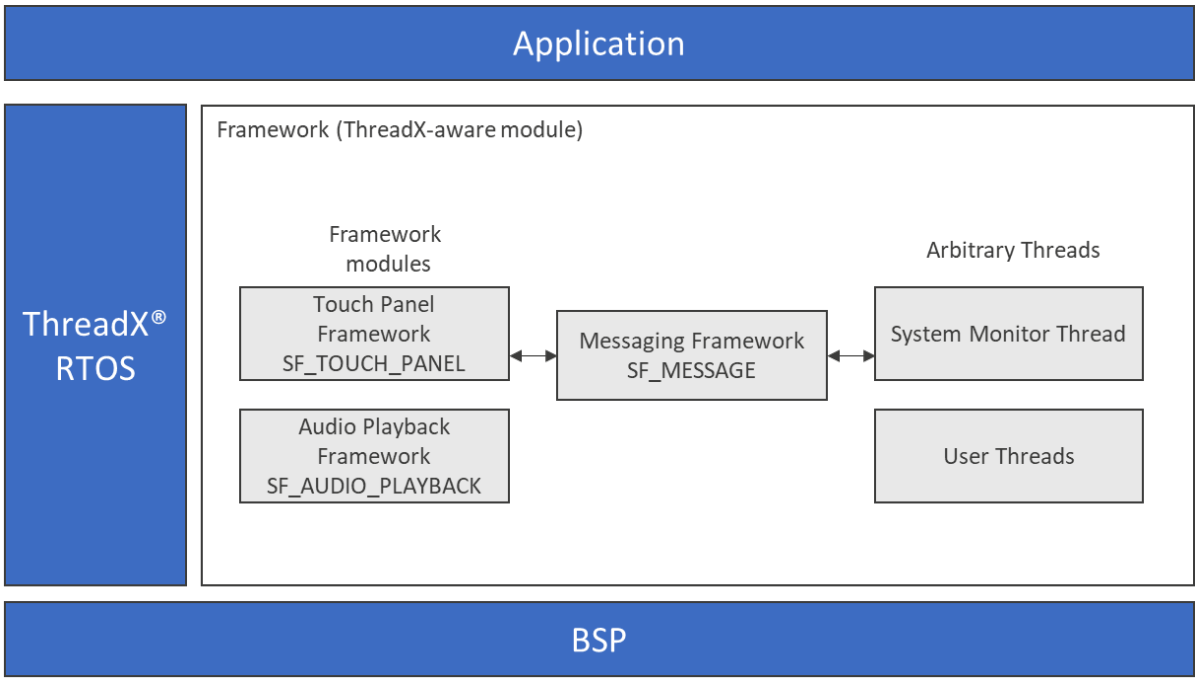


Figure 12.30 Inter-Thread Messaging Framework

12.25 Power Profile Framework Version 1

Important Note: Power Profile Framework Version 1 has been superseded by the Version 2 described in section 12.26, Power Profile Framework Version 2.

The Power Profile Framework Version 1 module has been deprecated and provided in this release for backward compatibility with existing projects developed during prior SSP releases. It is highly recommended to use the replacement module, Power Profiles Framework Version 2.

The Power Profiles Framework provides generic API that can be be sued by applications to place the MCU in lower power Software Standby mode. The module is implemented on sf_power_profiles.

Power Profiles Framework Module supports the following features:

- Supports Software Standby operations
- Supports multiple operating modes
  - Run
  - RTC
  - External Interrupt
- Selects clocks and peripherals to disable during Software Standby
- Selects output pin state prior to and after exiting Software Standby

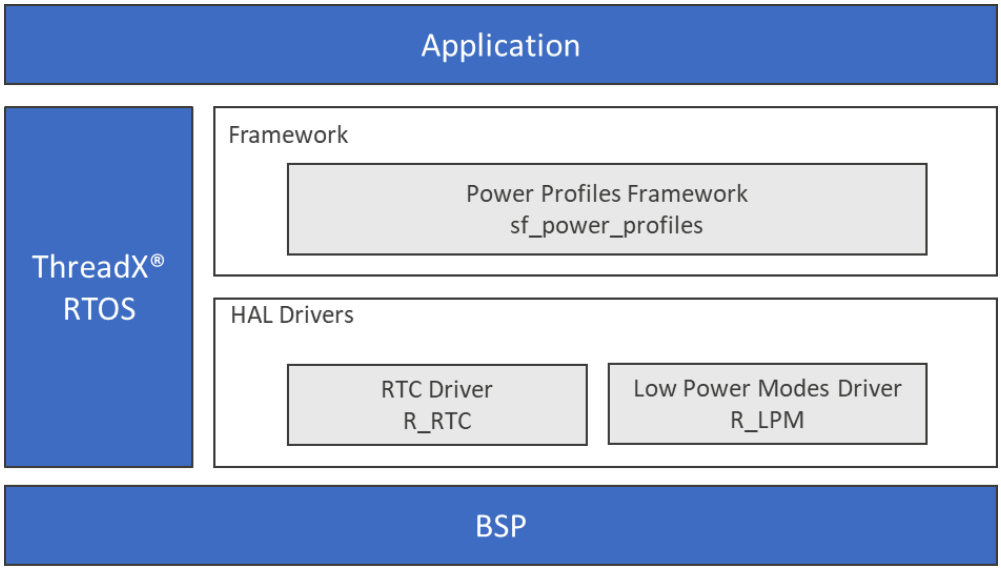


Figure 12.31 Power Mode Profile Framework (Version 1)

12.26 Power Profile Framework Version 2

The Power profiles V2 (PPv2) Framework provides more control for users to set a power control mode and the LPM mode in the Synergy MCU. The PPv2 Framework supports all the features of the Power Profiles v1 available in the current release of the Synergy Software Program (SSP) Framework using the LPM v1 driver. The Power Profiles v1 and PPv2 Framework are not compatible. Power Profiles v1 and PPv2 Framework cannot be used in the same project.

For all new projects, it is recommended that applications use the PPv2 Framework.

- Configurable options to set different MCU power control modes with customizable clock domains.
- Configurable options to set different MCU low power modes with different IO port or pin configurations.
- Supports both threaded and non-threaded operations

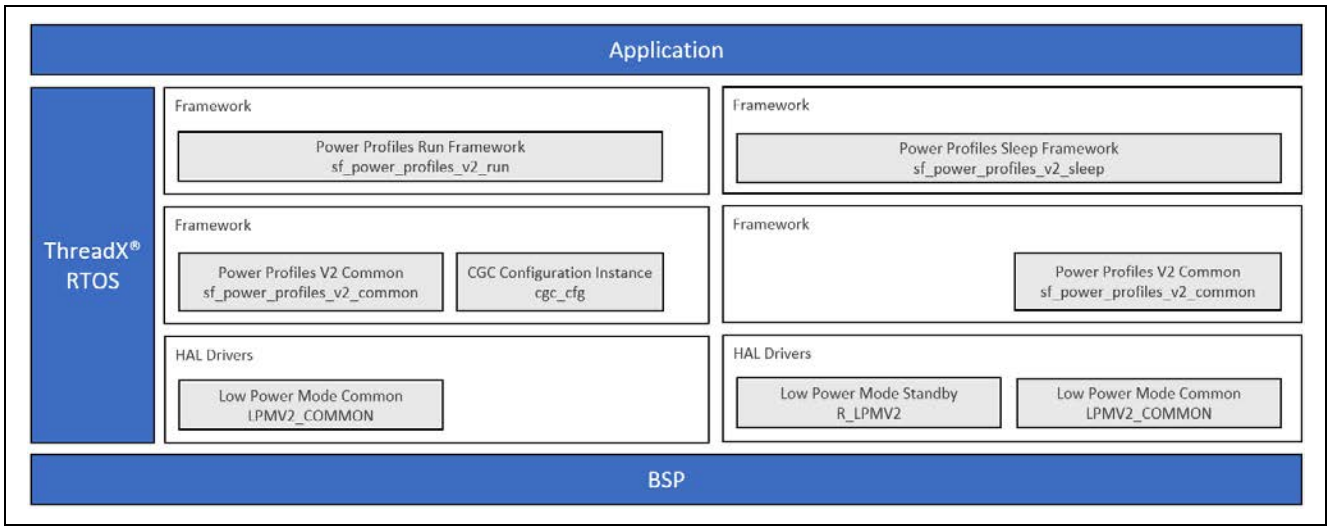


Figure 12.32 Power Mode Profile Framework Version 2

Note: There are many options for lower level LPMV2 HAL modules, since they are available for different MCUs. The figure does not show all available options. The short-hand <MCU> indicator is replaced with the CPU name (like S7G2) in the actual framework implementation.

12.27 Serial Peripheral Interface (SPI) Framework

The SPI Framework module provides set of ThreadX-aware framework APIs and is implemented on sf_spi. The SPI Framework module handles the integration and synchronization of multiple SPI peripherals on an SPI bus, including chip-select handling and its level activation. With the SPI Framework, one or more SPI buses can be created and multiple SPI peripherals can be connected to the SPI bus. The SPI Framework module uses a single interface to access both SCI SPI and RSPI drivers. The SPI Framework module uses the SCI and RSPI peripherals on the Synergy MCU.

SPI Framework Module Features:

The SPI Framework module uses either the SCI in SPI mode (together with the SCI common lower-level modules) or the RSPI lower-level driver module to communicate with the SPI peripherals on the Synergy microcontroller.

- Supports multiple devices on a bus
- Provides high-level APIs for initialization, transfers, and closing the module
- Supports synchronized transfers
- Supports chip-select operations
- Supports bus-locking

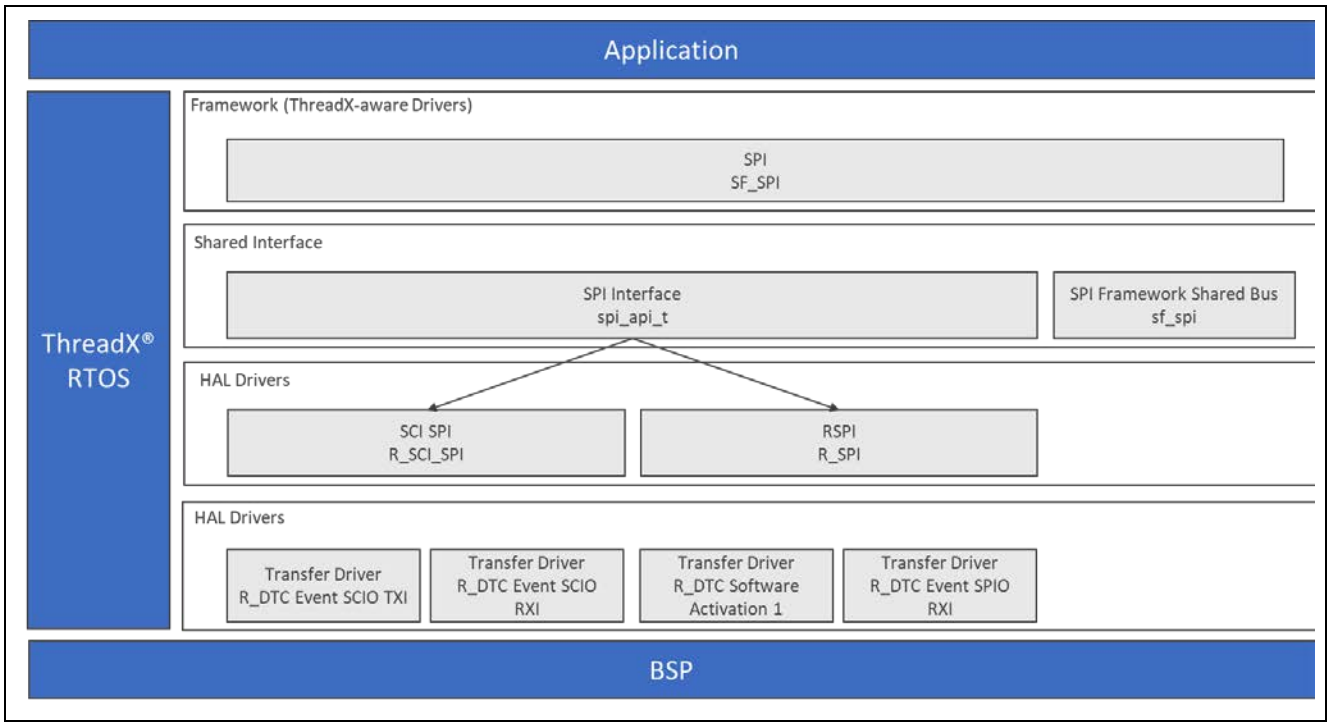


Figure 12.33 Serial Peripheral Interface (SPI) Framework

12.28 Synergy FileX® Port Block Media Interface Framework

The FileX Port Block Media Framework module, implemented on sf_el_fx, supports the Express Logic FileX system, a complete FAT format media and file management system for deeply embedded applications. The FileX Port Block Media Framework module provides I/O calls for FileX to access Synergy Media drivers through the Block Media Interface and adaptation layers. The SD/MMC HAL module and the SDIO HAL module are implemented on r_sdmmc and are used to read/write and control SD/MMC media devices as well as SDIO cards. The sf_el_fx module uses the SD/MMC peripherals on the Synergy MCU.

FileX Port Block Media Framework Features

- The FileX Port Block Media Framework module features include:
  - Support for FAT32, FAT16 and FAT12
  - Supports mounting the first FAT partition on a given media
  - Unlimited FileX objects (media, directories, and files)
  - Dynamic FileX object creation/deletion
  - Flexible memory usage
  - Size scales automatically
  - Small footprint
  - Complete integration with ThreadX
- SD card interface
  - Compatible with SD, SDHC, and SDXC formats
  - Supports 1-bit and 4-bit bus width
  - Card detect function when supported by the hardware
  - Write protect support
- eMMC interface
  - Supports 1-bit, 4-bit and 8-bit bus width
- SD bus interface
  - Compatible with SD memory card and SDIO card
  - Transfer bus mode selectable from 4-bit wide bus mode or 1-bit default bus mode
  - Compatible with SD, SDHC, and SDXC formats
- SD and MMC shared
  - DMAC and DTC triggerable by the SBFAI interrupt SD buffer is read and write accessible using the DMAC

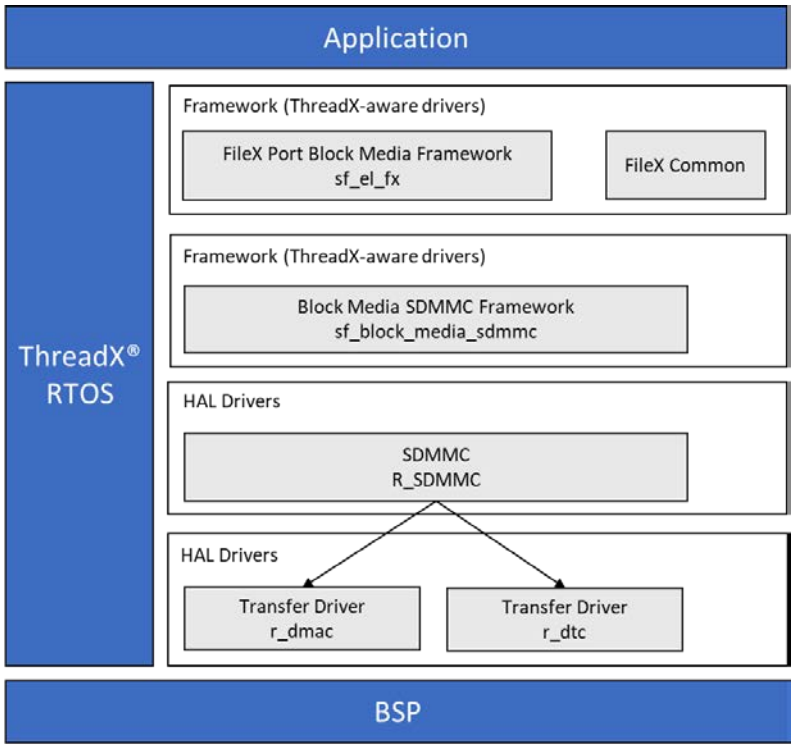


Figure 12.34 Synergy FileX® Interface Framework

12.29 Synergy GUIX™ Interface Framework

The Express Logic GUIX Synergy Port Module, sf_el_gx, is the Express Logic GUIX™ adaptation layer for Synergy MCU groups, which have graphics engines GLCDC, DRW (2DG engine), or a JPEG decode engine. The API supports graphics hardware engine setup for GUIX, graphics rendering, and displaying graphics accelerated by hardware engines. The module defines full-set of GUIX low-level display driver functions that draw graphics accelerated by the DRW (2DG engine), or JPEG, or displays graphics with GLCDC.. The module encourages the hardware acceleration for graphics rendering, but also allows software processing without hardware support.

GUIX Synergy Port Framework Module Features:

- Adapts GUIX to the SSP Framework
- Attaches the SSP Display Interface driver to GUIX Display Driver Interface
- Allows GUIX to draw widgets accelerated by the Synergy D2W (2DG) engine
- Allows GUIX to draw widgets accelerated by the Synergy JPEG engine
- Supports double-buffer toggling control for screen transitions without tearing
- Supports screen rotation (90/180/270 degree)
- Supports various output color formats
  - 32bpp (ARGB8888, RGB-888)
  - 16bpp (RGB565)
  - 8bpp; 8-bit palette color look-up table (CLUT)
- Support for user callback functions

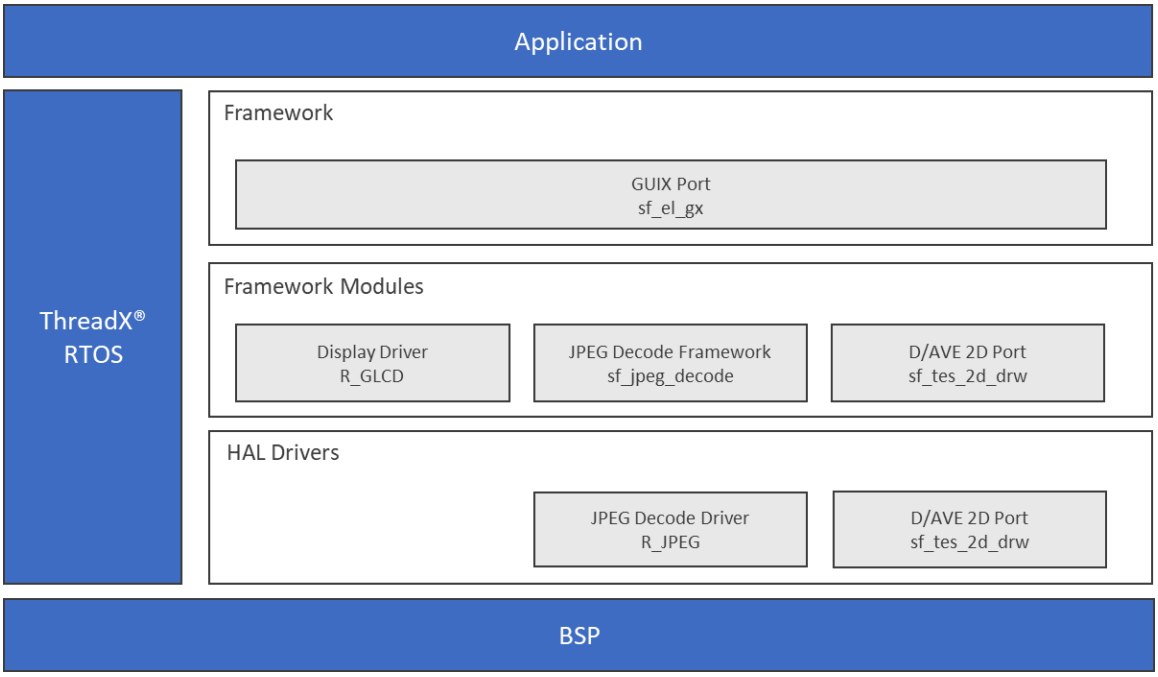


Figure 12.35 Synergy GUIX™ Interface Framework

12.30 Synergy NetX™ Port Ethernet Module

The Synergy NetX Port Ethernet module (sf_el_nx) for NetX and NetX Duo is integrated into the SSP. Its function is to interface the generic NetX and NetX Duo software with the hardware. This module includes the MAC driver, the PHY driver, additional glue logic and utility functions.

Note: Unless otherwise stated there is no difference is how this module works in NetX or NetX Duo projects.

NetX Port Ether Module Features

- NetX services are implemented as a library, so only code that is needed is added to the project
  - For most applications, this results in an instruction image from between 5k bytes and 30k bytes, with IPv6 and ICMPv6 enabled, the size is from 30k bytes to 45k bytes.
- NetX supports a variety of RFCs including the following:
  - RFC 1112 Host Extensions for IP Multicasting (IGMPv1)
  - RFC 1122 Requirements for Internet Hosts - Communication Layers
  - RFC 2236 Internet Group Management Protocol, Version 2
  - RFC 768 User Datagram Protocol (UDP)
  - RFC 791 Internet Protocol (IP)
  - RFC 792 Internet Control Message Protocol (ICMP)
  - RFC 793 Transmission Control Protocol (TCP)
  - RFC 826 Ethernet Address Resolution Protocol (ARP)
  - RFC 903 Reverse Address Resolution Protocol (RARP)
  - RFC 2460 Internet Protocol v6 (IPv6) Specification (NetX Duo only)
  - RFC 4443 Internet Control Message Protocol (ICMPV6) (NetX Duo only)
  - RFC 4861 Neighbor Discovery for IPv6 (NetX Duo only)
  - RFC 4862 IPv6 Stateless Address Auto Configuration (NetX Duo only)
- Packet-based zero-copy implementation of TCP/IP. Buffers are not copied inside NetX as they travel across the stack or from the stack to the user application, and this frees up processing cycles and frees up memory, to improve transmission rates significantly.
- Fast UDP processing

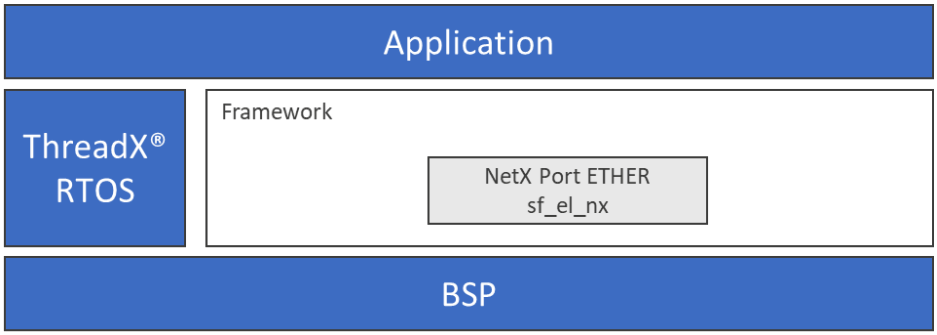


Figure 12.36 Net X Port Ether Module Organization, Options and Stack Implementations

12.31 Synergy USBX™ Port Framework

The Express Logic USBX™ Synergy Port framework module (sf_el_ux) is integrated into the SSP. This driver is meant to be used with the Express Logic USBX.

Express Logic USBX Synergy Port Framework Module Features

- Implements Express Logic USBX in SSP- support USBX APIs
- Supports the Port Device Controller Driver (DCD) for the USBHS peripheral
- Supports the Port Device Controller Driver (DCD) for the USBFS peripheral
- Supports the Port Host Controller Driver (HCD) for the USBHS peripheral
- Supports the Port Host Controller Driver (HCD) for the USBFS peripheral
- Supports transfer module operation (optional)

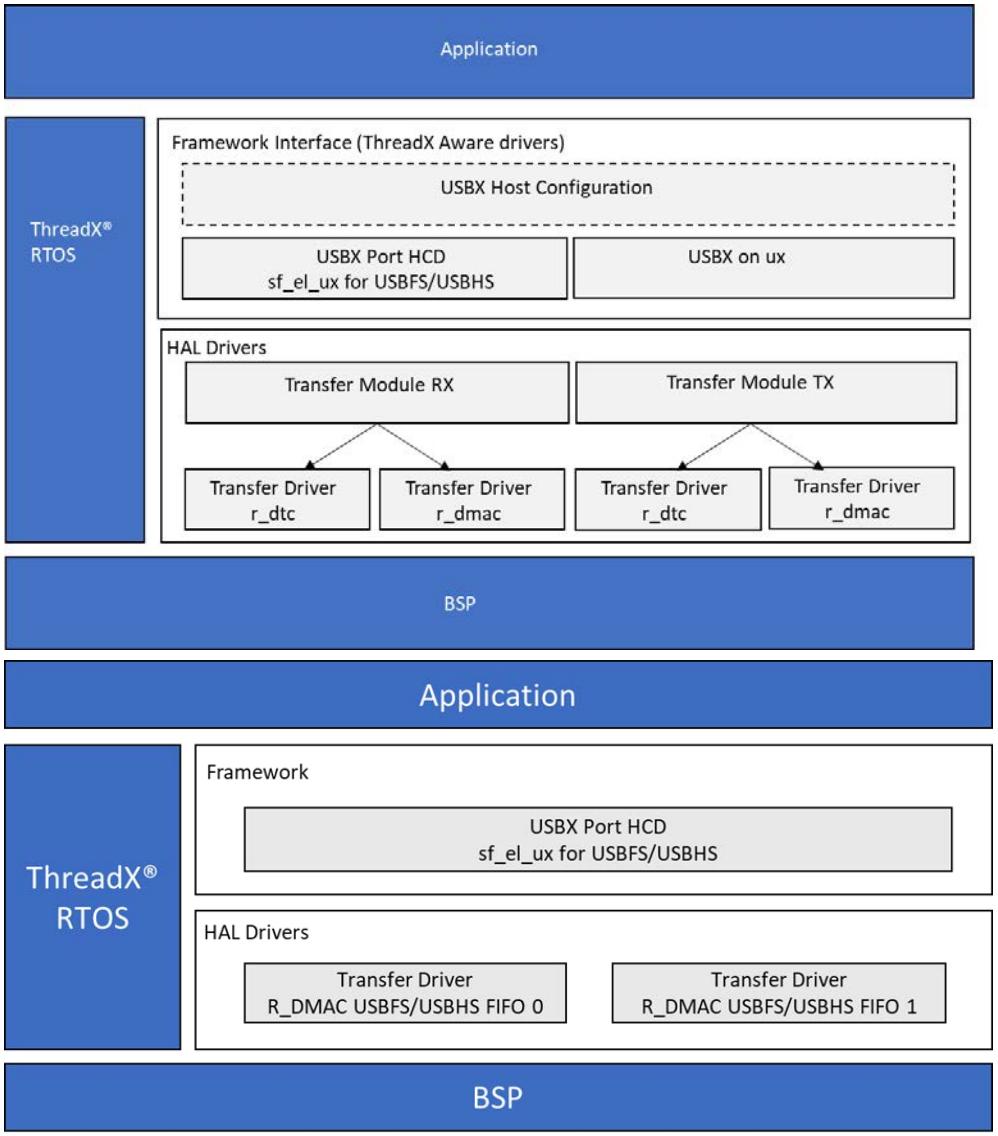


Figure 12.37 **Express Logic USBX Synergy Port Framework Organization, Options and Stack Implementation for DCD and HCD**

12.32 Thread Monitor Framework

The Thread Monitor Framework provides high-level APIs for thread monitor framework applications and is implemented on sf_thread_monitor. The framework configures the watchdog timer (WDT) or independent watchdog timer (IWDT). The Thread Monitor Framework uses WDT or IWDT peripherals on the Synergy MCU device.

Thread Monitor Framework Module supports the following features:

- The Thread Monitor Framework interface monitors RTOS threads using a watchdog timer. The Thread Monitor forces a watchdog reset of the microcontroller when any of the monitored threads do not behave as expected.
- The Thread Monitor is designed to support any Synergy device with either a WDT or IWDT peripheral, and a HAL module with no changes to the API.
- In profiling mode, the minimum and maximum counter values for registered threads can be determined. When in profiling mode, the watchdog timer is always refreshed and does not reset the device.
- Both the WDT and IWDT HAL modules are supported by this framework module.

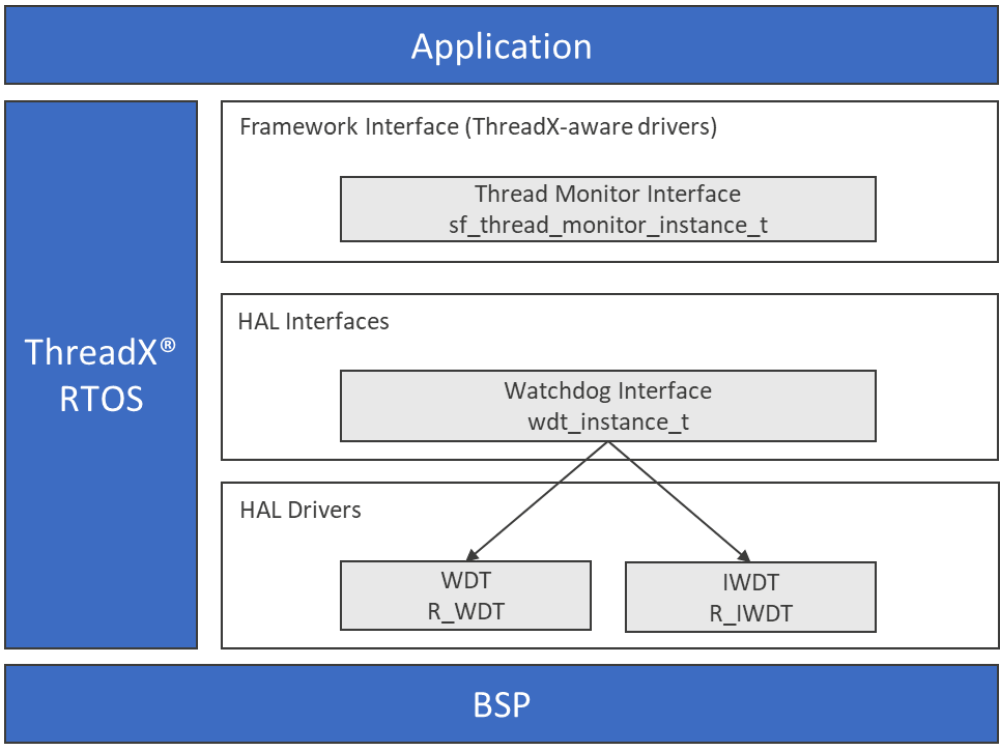


Figure 12.38 Thread Monitor Framework

12.33 Touch Panel I2C Framework

The Touch Panel Framework module provides a high-level API for reading messages from a touch controller and it is implemented using an I²C port. The Touch Panel Framework uses the IIC or SCI peripherals on the Synergy MCU.

Touch Panel Framework Module supports the following features:

- Reads data from a touch controller and publishes touch messages to any queue subscribed to touch events in the messaging framework
- Provides position data (X and Y coordinates)
- Provides the touch event type (down, up, move, hold, or invalid)
- Supports I²C-based touch panel implementations with external interrupts

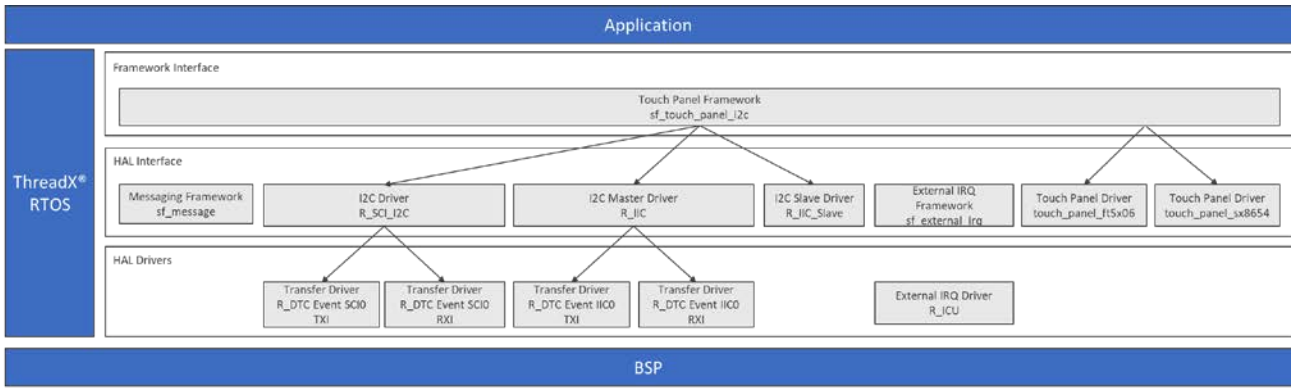


Figure 12.39 Touch Panel I²C Framework

12.34 UART Communications Framework

The UART communications framework provides an instance of the synergy communication framework APIs on an UART compliant synergy MCU peripheral. It uses r_sci_uart HAL driver, which configures and operates the SCI peripheral of Synergy MCUs to communicate using the UART protocol.

The frameworkis ThreadX-aware and and uses ThreadX objects to ensure the operations are thread safe.

UART Communications Framework Module supports the following features:

- UART Communications Protocol
- Locking a channel to reserve exclusive access
- ThreadX aware implementation

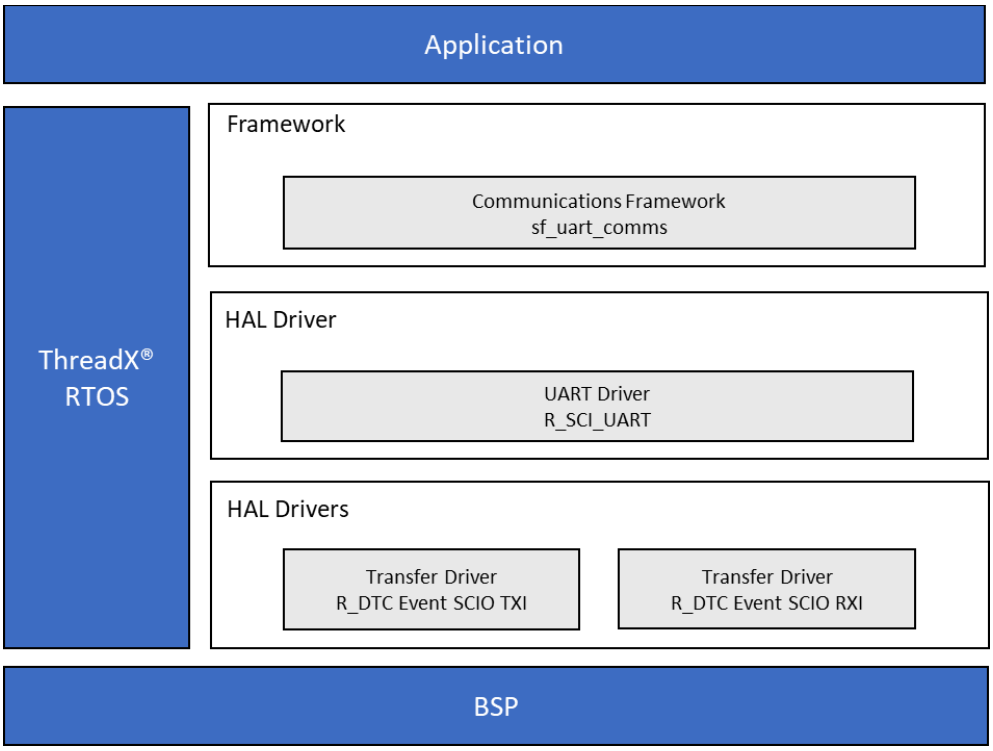


Figure 12.40 UART Framework

12.35 Wi-Fi Framework

The Wi-Fi framework provides high-level APIs to configure and provision Wi-Fi modules, as well as perform data transfers with or without on-chip networking capability. Currently, only the Qualcomm GT202 module is supported. The Wi-Fi framework communicates via the SPI with the underlying GT202 module.

The WiFi framework module has the following key features:

- Provides high-level APIs to configure and provision a WiFi module
- Provides four different implementations for:
  - A Wi-Fi device driver stack using the sf_wifi_gt202 framework
  - An on-chip stack using the sf_wifi_onchip_stack framework
  - A BSD socket stack using the sf_wifi_onchip_stack framework
  - A NetX and NetX Duo port using the sf_wifi_nsal_nx framework
- Using NetX and NASL:
  - Allows the same application code to be used across different Wi-Fi modules
  - Allows easy migration of the Ethernet-based application to a Wi-Fi based application
  - Allows for debugging and fine-tuning the application and TCP/IP stack as required by the application.
  - The current NSAL implementation only provides NetX NSAL. Adding support for a new network stack requires implementing the appropriate NSAL.
- Using the On-chip networking stack:
  - Is beneficial when using MCUs with a small memory footprint
  - Provides BSD sockets interface to create socket-based applications with the On-chip TCP/UDP
  - Provides an option to integrate 3rd-party application protocols on top of the TCP/IP such as MQTT and COAP without using the NetX stack.

Note: This on-chip networking stack is supported only when supported by the Wi-Fi module or WiFi module driver.

The following figures show the Wi-Fi Framework Module Organization, Options and Stack Implementations.

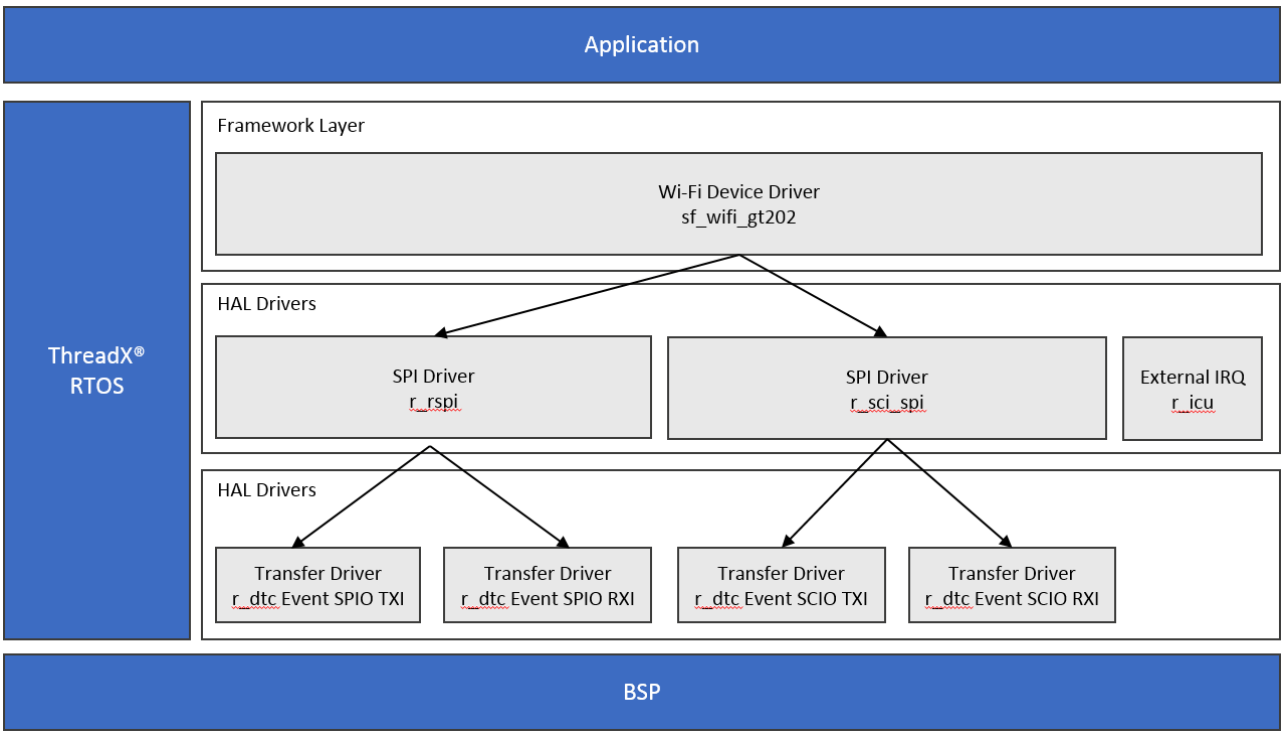


Figure 12.41 Wi-Fi Device Driver Implementation

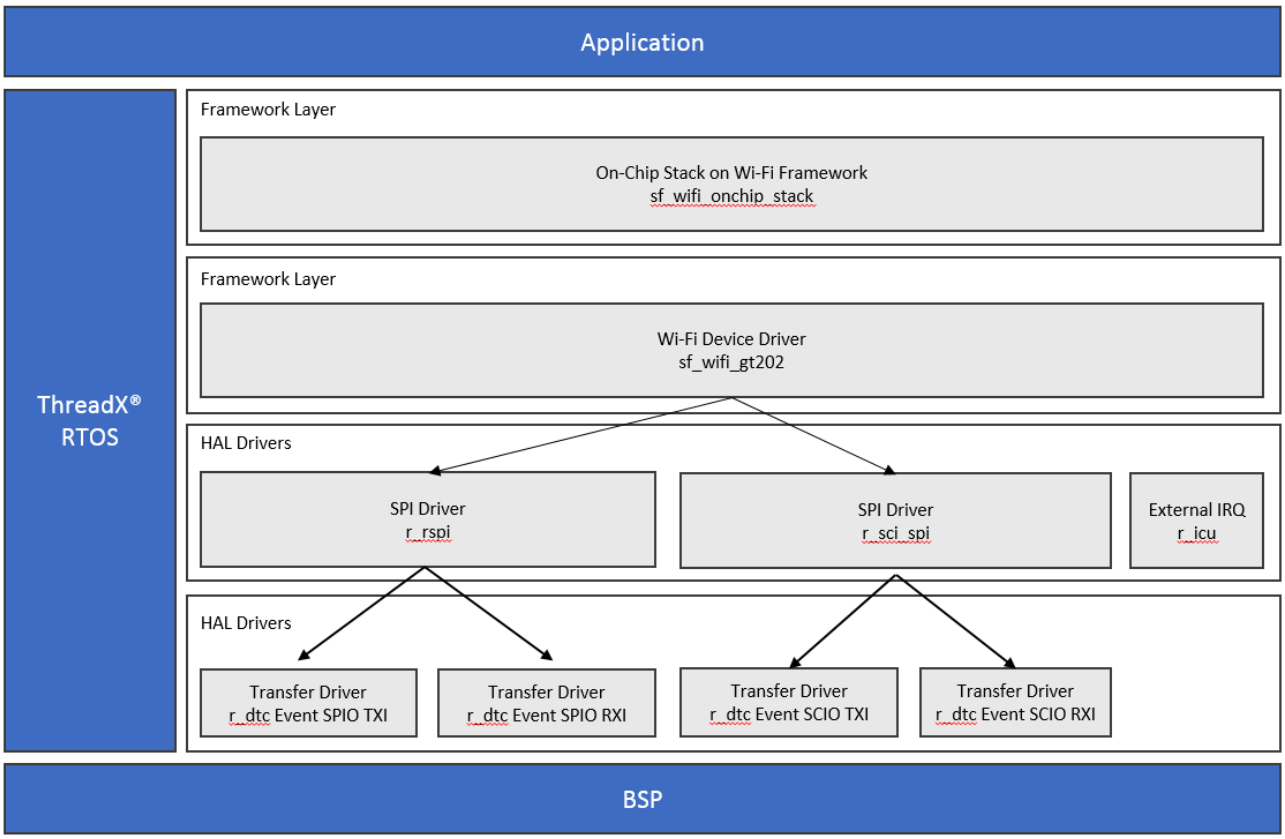


Figure 12.42 Wi-Fi On-Chip Stack Implementation

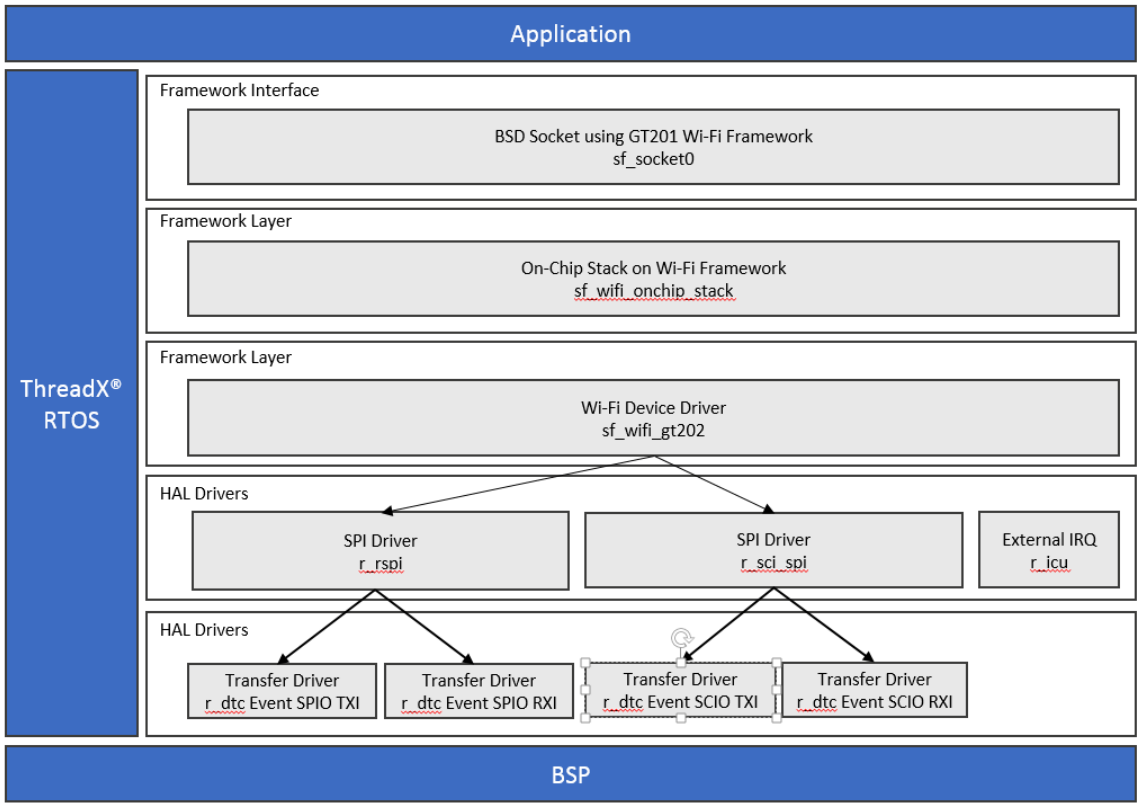


Figure 12.43 Wi-Fi BSD Socket Implementation

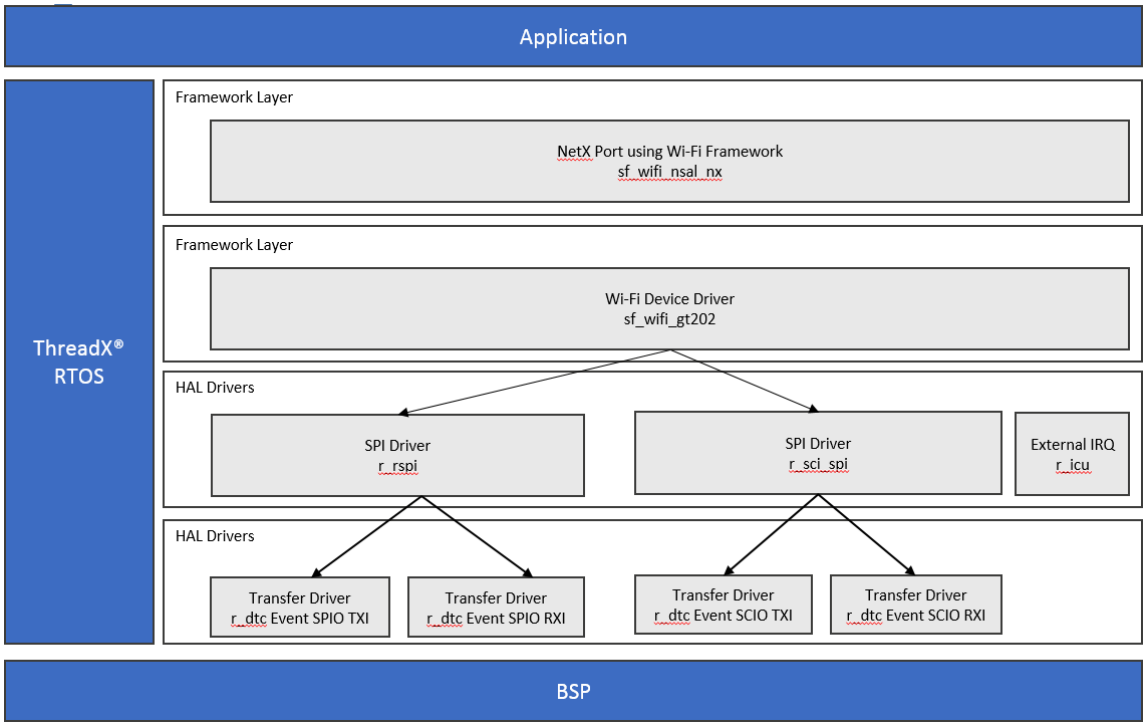


Figure 12.44 Wi-Fi NetX Port Implementation with NSAL

13. CMSIS DSP Library

The Arm Cortex® Microcontroller Software Interface Standard DSP hardware block (CMSIS-DSP) in the Cortex®-M4 processor core-based Synergy family of MCUs provides a suite of common digital signal processing functions.

The CMSIS-DSP library is a hardware abstraction layer included in SSP for Synergy MCUs that includes a collection of over 60 completely optimized signal processing functions commonly used in digital signal control applications. The library supports key arithmetic formats such as fixed-point/fractional (Q7, Q15, Q31) and single precision floating-point (32-bit) arithmetic for DSP operations. The combination of high-efficiency signal processing functions in SSP with the low-power, low-cost, and high-performance benefits of Synergy MCUs having underlying SIMD architecture and FPU provide a compelling solution for diverse applications in IoT and M2M markets.

The CMSIS-DSP library covers operations under the following major categories:

Category	Functions
Basic Math	Absolute Value, Addition, Dot Product, Multiplication, Negate, Offset, Scale, Shift, Subtract
Common Tables	FFT twiddle factors, Bit reverse, Reciprocal
Complex Math	Conjugate, Dot Product, Magnitude, Magnitude Squared, Complex by Complex Multiplication, Complex by Real Multiplication
Fast Math	Cosine, Sine, Square Root
Filter	Q31, IIRs, FIRs, LMS, Convolution, Correlation
Matrix Math	Addition, Multiplication, Initialization, Inverse, Scale, Subtraction, Transpose
Statistics	Maximum, Mean, Minimum, Power, RMS, Standard deviation, Variance
Transform	Complex FFT, Radix-8 Complex FFT, DCT Type IV, Real FFT, Complex FFT Tables, Real FFT

Figure 13.1 CMSIS-DSP Library Features

14. Hardware Abstraction Layer (HAL) Modules

14.1 Introduction

HAL modules in SSP are device-independent drivers for peripherals available on Synergy MCUs. The HAL modules provide abstracted and well-defined interfaces. The underlying functionality of these interfaces can be implemented by multiple device drivers. The HAL drivers use system services like timers and provide generic, high-level, C-callable interfaces which are functional but device independent. The Application Framework in SSP uses the HAL drivers for interfacing with the low-level device-specific drivers. HAL drivers can also be used by application programs to interface directly with the respective peripheral, bypassing the SSP Framework. However, these modules are RTOS independent (not ThreadX® aware) and are not thread-safe.

In this Software Datasheet, Renesas also identifies the key features supported by Synergy MCUs and the level of support for these peripherals and features in SSP HAL software.

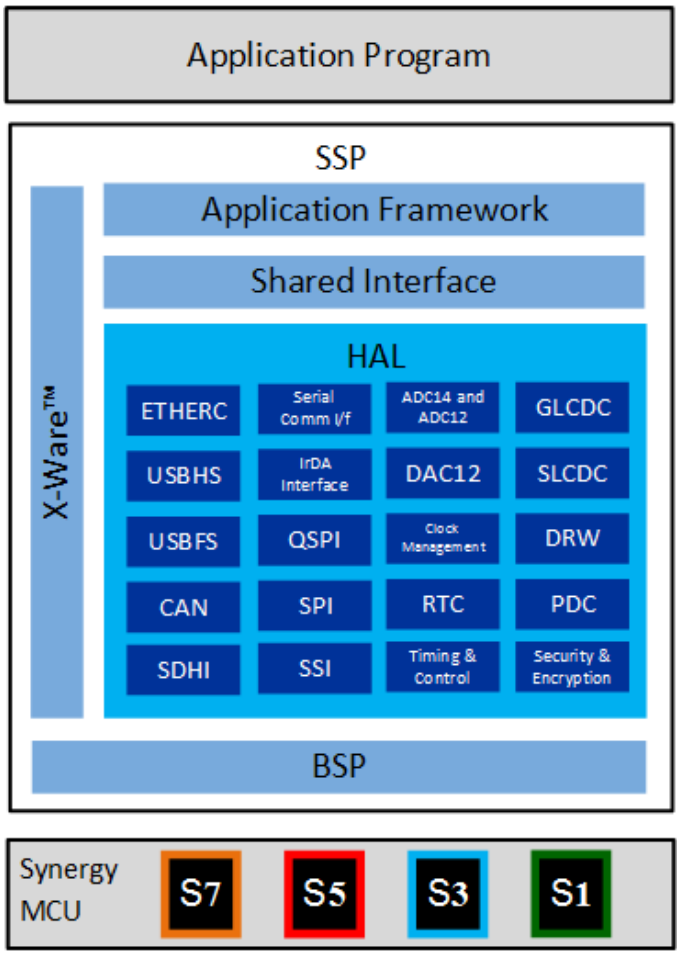


Figure 14.1 Hardware Abstraction Layer

14.2 Complete List of HAL Modules in SSP v1.4.0

These HAL modules are available for respective MCUs based on the following criteria:

1. If the core functionality of the module has been tested and works on a MCU, even if there are known bugs, the module is then supported on the MCU.
2. If the core functionality is broken or not tested on a MCU, then that module is not supported on the MCU.
3. If a module has been tested on one of the Synergy MCUs, and is independent of the underlying MCU hardware or HAL drivers, the module is supported on all Synergy MCUs on which the underlying driver/framework/stacks the module depends have been completely tested on that MCU.

HAL Driver	SSP Feature	Supported Synergy MCU Group
r_acmphs	Analog Comparator High Speed	S1JA
r_acmplp	Analog Comparator Low Power	S1JA
r_adc	A/D Converter	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_agt	Asynchronous General Purpose Timer	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_cac	Clock Frequency Accuracy Measurement Circuit	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_can	Controller Area Network	S1JA, S124, S128, S3A1,S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_cgc	Clock Generation Circuit	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_crc	Cyclic Redundancy Check Calculator	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_ctsu	Capacitive Touch Sensing Unit	S124, S128, S3A3, S3A7, S5D5, S5D9, S7G2
r_dac	Digital to Analog Converter	S124, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_dac8	8-bit Digital to Analog Converter	S1JA, S128, S3A3
r_dmac	Direct Memory Access Controller	S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_doc	Data Operation Circuit	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_dtc	Data Transfer Controller	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_elc	Event Link Controller	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_flash_hp	Flash Memory, High Performance	S5D5, S5D9, S7G2
r_flash_lp	Flash Memory, Low Power	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7
r_fmi	Factory Microcontroller Information	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_glcd	Graphics LCD Controller	S5D9, S7G2
r_gpt	General Purpose Timer	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_gpt_input_capture	General Input Capture	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_icu	Interrupt Controller Unit	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_ioport	General Purpose I/O Ports	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_iwdt	Independent Watchdog Timer	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_jpeg_decode	JPEG Decoder	S5D9, S7G2
r_jpeg_encode	JPEG Encoder	S5D9, S7G2

HAL Driver	SSP Feature	Supported Synergy MCU Group
r_kint	Keyboard Interrupt Interface	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_lpm†	Low Power Mode	S124, S3A7, S7G2
r_lpmv2_s1ja	Low Power Mode V2 for S1JA	S1JA
r_lpmv2_s124	Low Power Mode V2 for S124	S124
r_lpmv2_s128	Low Power Mode V2 for S128	S128
r_lpmv2_s3a1	Low Power Mode V2 for S3A1	S3A1
r_lpmv2_s3a3	Low Power Mode V2 for S3A3	S3A3
r_lpmv2_s3a6	Low Power Mode V2 for S3A6	S3A6
r_lpmv2_s3a7	Low Power Mode V2 for S3A7	S3A7
r_lpmv2_s5d5	Low Power Mode V2 for S5D5	S5D5
r_lpmv2_s5d9	Low Power Mode V2 for S5D9	S5D9
r_lpmv2_s7g2	Low Power Mode V2 for S7G2	S7G2
r_lvd	Low Voltage Detection Driver	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_opamp	Operational amplifier	S1JA, S128
r_pdc	Parallel Data Capture Unit	S5D5, S5D9, S7G2
r_qspi	Quad Serial Peripheral Interface	S3A1, S3A3, S3A7, S5D5, S5D9, S7G2
r_riic_master	IIC Master	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_riic_slave	IIC Slave	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_rspi	Serial Peripheral Interface	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_rtc	Real-time Clock	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_sce# See table note on Cryptographic Functions	Cryptographic Library (HAL interfaces)	S1JA, S124,S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_sci_i2c	Serial Communication Interface I2C	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_sci_spi	Serial Communication Interface SPI	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_sci_uart	Serial Communication Interface UART	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2
r_sdadc	Sigma Delta ADC	S1JA
r_sdmmc	SDHI Driver for SDIO and SD/MMC Memory Devices	S3A1, S3A3, S3A7, S5D5, S5D9, S7G2
r_slcdc	Segment LCD Controller	S3A1, S3A3, S3A6, S3A7
r_ssi	(Inter-IC Sound) Interface [old: Serial Sound Interface] or r_i2s	S3A1, S3A3,S3A6, S3A7, S5D5, S5D9, S7G2
r_wdt	Watchdog Timer	S1JA, S124, S128, S3A1, S3A3, S3A6, S3A7, S5D5, S5D9, S7G2

† Indicates a module that is deprecated starting with SSP v1.3.0 and all subsequent versions. Deprecated modules will only be available to maintain compatibility with existing projects that may be using them. It is highly recommended that new projects use the recommended replacements and not use deprecated modules. For details, see the SSP User's Manual. For details, see the SSP User's Manual.

Cryptographic Functions: See Table 14.1 which lists cryptographic functions available for each MCU in this release; these functions are accessible as part of r_sce/cryptographic library.

14.3 Analog Comparator High-Speed (ACMPHS)

The high-speed analog comparator (ACMPHS) HAL module implements the comparator API for signal processing applications on r_acmphs. It supports the ACMPHS peripheral available on the Synergy microcontroller hardware. A callback is available to signal the user application on transition events.

ACMPHS HAL Module Features

- •Callback on rising edge, falling edge, or both
- •Configurable debounce filter
- •Option to include comparator output on VCOUT pin

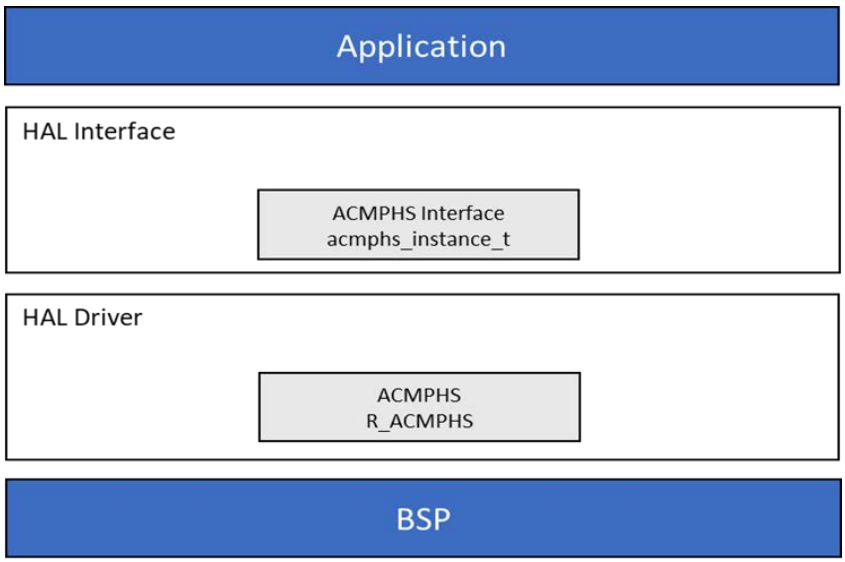


Figure 14.2 ACMPHS HAL Module Block Diagram

The following hardware features are, or are not, supported by the SSP for the ACMPHS:

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Query HS Comparator result	Callback on rising edge, falling edge, or both	Configurable debounce	Option to include comparator output on VCOUT
S124	N/A	N/A	N/A	N/A
S128	☒	☒	☒	☒
S1JA	✓	✓	✓	✓
S3A1	N/A	N/A	N/A	N/A
S3A3	N/A	N/A	N/A	N/A
S3A6	N/A	N/A	N/A	N/A
S3A7	☒	☒	☒	☒
S5D5	☒	☒	☒	☒
S5D9	☒	☒	☒	☒
S7G2	☒	☒	☒	☒

14.4 Analog Comparator Low-Power (ACMPLP)

The ACMPLP HAL module implements the Comparator API for signal processing applications on r_acmplp. It supports the ACMPLP peripheral available on the Synergy microcontroller hardware. A callback is available to signal the user application on transition events.

ACMPLP HAL Module Features

- Normal mode or window mode
- Callback on rising edge, falling edge, or both
- Configurable debounce filter
- Option to include comparator output on VCOUT pin

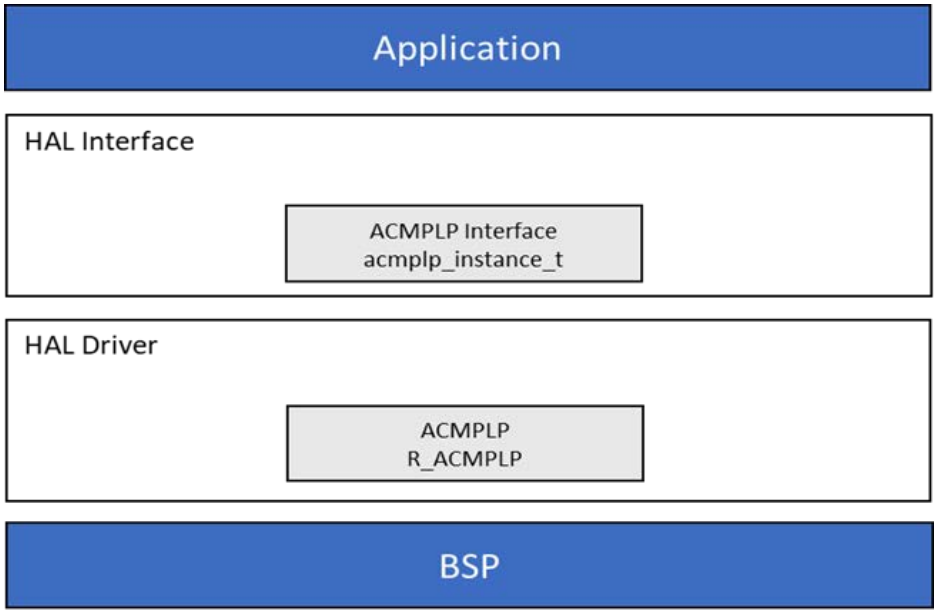


Figure 14.3 ACMPLP HAL Module Block Diagram

The following hardware features are, or are not, supported by the SSP for the ACMPLP:

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Normal or window mode	Callback on rising edge, falling edge, or both	Configurable debounce	Option to include comparator output on VCOUT
S124	☒	☒	☒	☒
S128	☒	☒	☒	☒
S1JA	✓	✓	✓	✓
S3A1	☒	☒	☒	☒
S3A3	☒	☒	☒	☒
S3A6	☒	☒	☒	☒
S3A7	☒	☒	☒	☒
S5D5	N/A	N/A	N/A	N/A
S5D9	N/A	N/A	N/A	N/A
S7G2	N/A	N/A	N/A	N/A

14.5 Analog to Digital Converter (ADC)

The ADC HAL module implements APIs for analog-to-digital conversions on r_adc. It supports the ADC12, ADC14 and ADC16 for the associated peripherals available on Synergy MCUs. A user-defined callback can be used to process the data each time a new sample is complete.

ADC HAL Module Features

- 16-Bit A/D Converter (S1JA), 14-Bit A/D Converter (S3A7,S3A3,S3A6,S3A1,S128,S124) and 12-Bit A/D Converter (S7G2,S5D9,S5D5)
- Multiple Operation Modes
  - — Single Scan
  - — Group Scan
  - — Continuous Scan
- Multiple Channels
  - — All analog channels on MCU
    - 13 channels (unit 0) or 12 channels (unit 1) for S7G2
    - 17 channels for S1JA
    - 18 channels for S124
    - 28 channels for S3A7
  - — Temperature sensor channel
  - — Voltage sensor channel

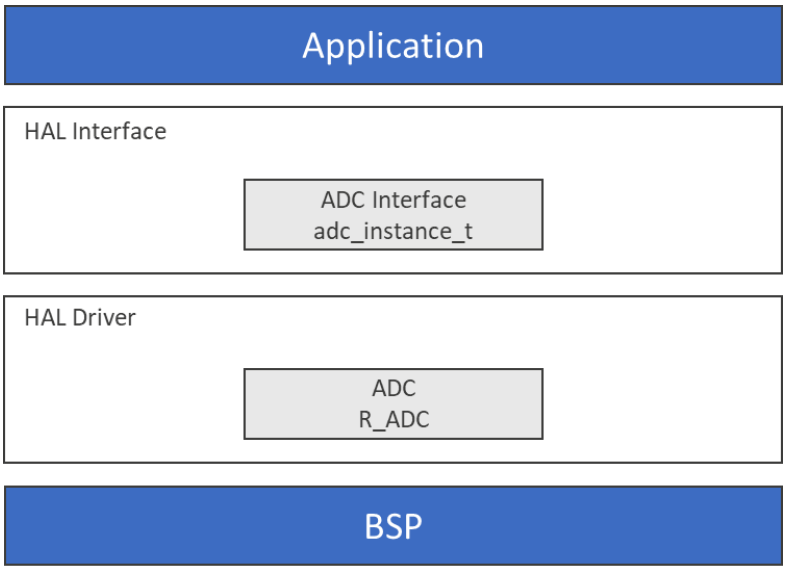


Figure 14.4 Analog to Digital Converter

The following hardware features are, or are not, supported by the SSP for the ADC:

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Support for all Analog Channels (Unit 0 and Unit 1)	8-Bit	10-bit	12-Bit	14-bit	16-bit	Single-scan Mode
S124	✓	N/A	N/A	✓	✓	N/A	✓
S128	✓	N/A	N/A	✓	✓	N/A	✓
S1JA	✓	N/A	N/A	N/A	N/A	✓	✓
S3A1	✓	N/A	N/A	✓	✓	N/A	✓

MCU Group	Support for all Analog Channels (Unit 0 and Unit 1)	8-Bit	10-bit	12-Bit	14-bit	16-bit	Single-scan Mode
S3A3	✓	N/A	N/A	✓	✓	N/A	✓
S3A6	✓	N/A	N/A	✓	✓	N/A	✓
S3A7	✓	N/A	N/A	✓	✓	N/A	✓
S5D5	✓	✓	✓	✓	N/A	N/A	✓
S5D9	✓	✓	✓	✓	N/A	N/A	✓
S7G2	✓	✓	✓	✓	N/A	N/A	✓

MCU Group	Continuous Scan Mode	Group Scan Mode	Programmable Gain Amplifier	Event link function through ELC HAL driver*
S124	✓	✓	☒	☒
S128	✓	✓	☒	☒
S1JA	✓	✓	☒	☒
S3A1	✓	✓	☒	☒
S3A3	✓	✓	☒	☒
S3A6	✓	✓	☒	☒
S3A7	✓	✓	☒	☒
S5D5	✓	✓	☒	☒
S5D9	✓	✓	☒	☒
S7G2	✓	✓	☒	☒

* Note: ELC is supported but only for Group Mode. This must be set up manually by programming the Event Link Controller.

14.6 Asynchronous General Purpose Timer (AGT)

The AGT HAL module provides high-level APIs for timing applications and is implemented on r_agt. The AGT HAL module uses the AGT peripheral on the Synergy MCU. A user-defined callback can be created to respond to a timer event.

The AGT HAL module configures a timer to a user-specified period. When the period elapses, any of the following events can occur:

- Interrupt the CPU, which calls a user-callback function (if provided)
- Toggle a port pin
- Transfer data using DMAC/DTC (if configured with transfer Interface)
- Start another peripheral (if configured with events and peripheral definitions)

AGT HAL Module Features:

- Multiple Channels: 16-bit x 2 channels
  - Channel 1 can be clocked by the channel 0 underflow, creating a cascaded 32-bit timer
- Core Clock: Can be clocked using PCLKB, LOCO, or Fsub. When clocked by LOCO or Fsub, it can be used to wake up the MCU from sleep modes

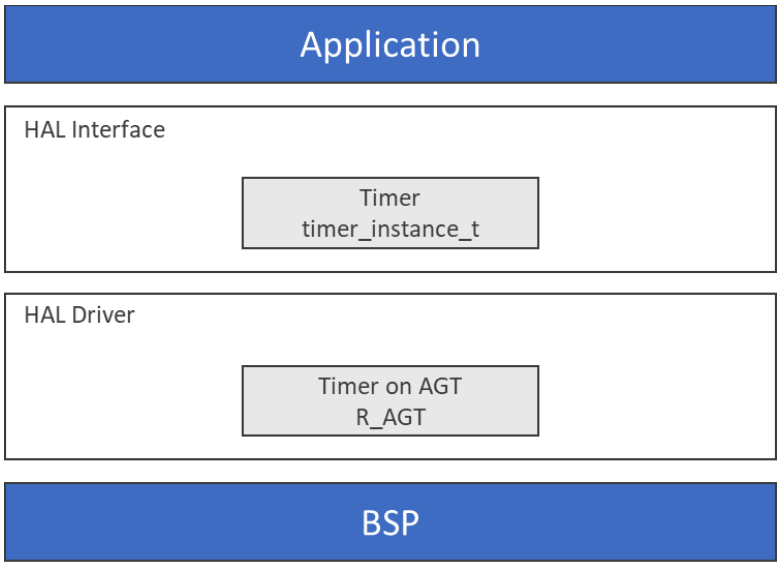


Figure 14.5 Asynchronous General Purpose Timer

The following hardware features are, or are not, supported by the SSP for the AGT.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Timer Mode	Pulse Output Mode	Event Counter Mode	Pulse width measurement mode
S124	✓	✓	☒	☒
S128	✓	✓	☒	☒
S1JA	✓	✓	☒	☒
S3A1	✓	✓	☒	☒
S3A3	✓	✓	☒	☒
S3A6	✓	✓	☒	☒
S3A7	✓	✓	☒	☒
S5D5	✓	✓	☒	☒
S5D9	✓	✓	☒	☒
S7G2	✓	✓	☒	☒

MCU Group	Pulse period measurement mode	Event link function through ELC HAL driver	Compare/Match Function
S124	☒	☒	✓
S128	☒	☒	✓
S1JA	☒	☒	✓
S3A1	☒	☒	✓
S3A3	☒	☒	✓
S3A6	☒	☒	✓
S3A7	☒	☒	✓
S5D5	☒	☒	✓
S5D9	☒	☒	✓
S7G2	☒	☒	✓

14.7 Clock Frequency Accuracy Measurement (CAC)

The CAC HAL module provides high-level APIs for clock accuracy-control applications and is implemented on r_cac. The module uses the Clock Frequency Accuracy Measurement Circuit (CAC) peripheral on the Synergy MCU, which is useful in implementing a fail-safe mechanism for reliability-oriented applications. A user-defined callback can be created to respond to various error indications.

The CAC HAL module API interfaces with a clock frequency-measurement circuit capable of monitoring the clock frequency based on a reference-signal input. The reference signal may be an externally supplied clock source or one of several available internal clock sources. An interrupt request may optionally be generated by a completed measurement, a detected frequency error, or a counter overflow. A digital filter is available for an externally supplied reference clock, and dividers are available for both internally supplied measurement and reference clocks. Edge-detection options for the reference clock are configurable as rising, falling, or both.

The frequency of the following clocks can be measured:

- Clock output from main clock oscillator (main clock)
- Clock output from sub-clock oscillator (sub-clock)
- Clock output from high-speed on-chip oscillator (HOCO clock)
- Clock output from low-speed on-chip oscillator (LOCO clock)
- Clock output from mid-speed on-chip oscillator (MOCO clock)
- Clock output from IWDT-dedicated on-chip oscillator (IWDTCLK clock)
- Peripheral module clock (PCLKB)

The measurement clock is monitored using a reference clock. The reference clock may be an external clock, supplied on the CACREF input pin, or one of the following internal clocks:

- Clock output from main clock oscillator (main clock)
- Clock output from sub-clock oscillator (sub-clock)
- Clock output from high-speed on-chip oscillator (HOCO clock)
- Clock output from mid-speed on-chip oscillator (MOCO clock)
- Clock output from low-speed on-chip oscillator (LOCO clock)
- Clock output from IWDT-dedicated on-chip oscillator (IWDTCLK clock)
- Peripheral module clock (PCLKB)

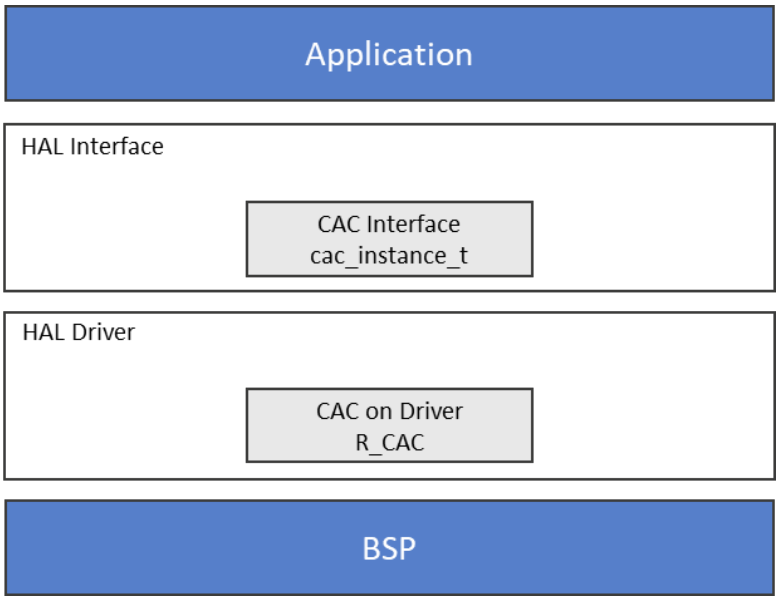


Figure 14.6 Clock Frequency Accuracy Measurement

The following hardware features are, or are not, supported by SSP for the CAC.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Measurement Clock Targets: Main Clock Oscillator	Measurement Clock Targets: Sub-Clock Oscillator	Measurement Clock Targets: HOCO clock	Measurement Clock Targets: MOCO Clock	Measurement Clock Targets: LOCO Clock
S124	✓	✓	✓	✓	✓
S128	✓	✓	✓	✓	✓
S1JA	✓	✓	✓	✓	✓
S3A1	✓	✓	✓	✓	✓
S3A3	✓	✓	✓	✓	✓
S3A6	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓
S5D5	✓	✓	✓	✓	✓
S5D9	✓	✓	✓	✓	✓
S7G2	✓	✓	✓	✓	✓

MCU Group	Measurement Clock Targets: IWDTCLK	Measurement Clock Targets: Peripheral Module Clock B	Measurement Reference Clocks: External clock input to the CACREF pin	Measurement Reference Clocks: Main Clock Oscillator	Measurement Reference Clocks: Sub-Clock Oscillator
S124	✓	✓	✓	✓	✓
S128	✓	✓	✓	✓	✓
S1JA	✓	✓	✓	✓	✓
S3A1	✓	✓	✓	✓	✓
S3A3	✓	✓	✓	✓	✓
S3A6	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓
S5D5	✓	✓	✓	✓	✓
S5D9	✓	✓	✓	✓	✓
S7G2	✓	✓	✓	✓	✓

MCU Group	Measurement Reference Clocks: HOCO Clock	Measurement Reference Clocks: MOCO Clock	Measurement Reference Clocks: LOCO Clock	Measurement Reference Clocks: IWDTCLK Clock	Measurement Reference Clocks: Peripheral Module Clock B
S124	✓	✓	✓	✓	✓
S128	✓	✓	✓	✓	✓
S1JA	✓	✓	✓	✓	✓
S3A1	✓	✓	✓	✓	✓
S3A3	✓	✓	✓	✓	✓
S3A6	✓	✓	✓	✓	✓

MCU Group	Measurement Reference Clocks: HOCO Clock	Measurement Reference Clocks: MOCO Clock	Measurement Reference Clocks: LOCO Clock	Measurement Reference Clocks: IWDTCLK Clock	Measurement Reference Clocks: Peripheral Module Clock B
S3A7	✓	✓	✓	✓	✓
S5D5	✓	✓	✓	✓	✓
S5D9	✓	✓	✓	✓	✓
S7G2	✓	✓	✓	✓	✓

MCU Group	Selectable Function: Digital Filter	Interrupt Sources: Measurement End	Interrupt Sources: Frequency Error	Interrupt Sources: Overflow	Module-stop function to reduce power consumption
S124	✓	✓	✓	✓	✓
S128	✓	✓	✓	✓	✓
S1JA	✓	✓	✓	✓	✓
S3A1	✓	✓	✓	✓	✓
S3A3	✓	✓	✓	✓	✓
S3A6	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓
S5D5	✓	✓	✓	✓	✓
S5D9	✓	✓	✓	✓	✓
S7G2	✓	✓	✓	✓	✓

14.8 Controller Area Network (CAN)

The CAN HAL module provides high-level APIs for a CAN network. The CAN HAL module is implemented on r_can and supports the CAN peripherals available on the Synergy microcontroller hardware. A user-callback function must be defined, which the driver will invoke when transmit, receive or error interrupts are received. The callback returns with a parameter which indicates the channel, mailbox and event.

The CAN HAL Module supports the following features:

- Both standard (11-bit) and extended (29-bit) messaging format
- Bit-timing configuration. as defined in the CAN specification
- Up to 32 transmit or receive mailboxes with standard or extended ID frames
- Receive mailboxes can be configured to capture either data or remote CAN frames
- A user-callback function when transmit, receive or error interrupts are received

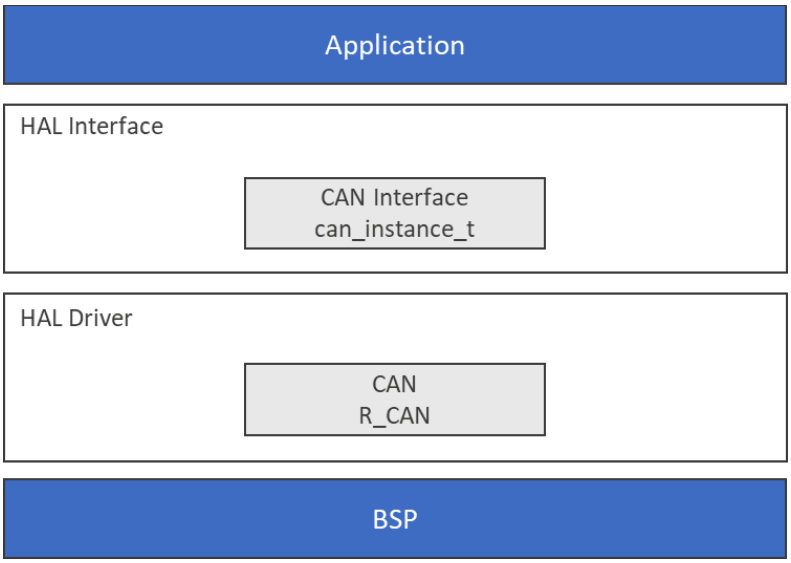


Figure 14.7 Controller Area Network

The following hardware features are, or are not, supported by SSP for CAN:

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Programmable Bit Rate	Support up to 32 Mailboxes	Mailbox Normal Mode	Mailbox FIFO Mode	Support for Data Frame Reception	Support for Remote Frame Reception
S124	✓	✓	✓	☒	✓	✓
S128	✓	✓	✓	☒	✓	✓
S1JA	✓	✓	✓	☒	✓	✓
S3A1	✓	✓	✓	☒	✓	✓
S3A3	✓	✓	✓	☒	✓	✓
S3A6	✓	✓	✓	☒	✓	✓
S3A7	✓	✓	✓	☒	✓	✓
S5D5	✓	✓	✓	☒	✓	✓
S5D9	✓	✓	✓	☒	✓	✓
S7G2	✓	✓	✓	☒	✓	✓

MCU Group	Programmable one-shot reception	Overwrite mode Reception	Overrun mode Reception	Support all 8 Acceptance Masks	Support Masks independently enabled or disabled for each Mailbox
S124	☒	✓	✓	✓	✓
S128	☒	✓	✓	✓	✓
S1JA	☒	✓	✓	✓	✓
S3A1	☒	✓	✓	✓	✓
S3A3	☒	✓	✓	✓	✓
S3A6	☒	✓	✓	✓	✓
S3A7	☒	✓	✓	✓	✓
S5D5	☒	✓	✓	✓	✓
S5D9	☒	✓	✓	✓	✓
S7G2	☒	✓	✓	✓	✓

MCU Group	Support for transmission request abort	Mode transition for bus-off: ISO11898-1 specification-compliant	Mode transition for bus-off: Automatic invoking of CAN halt mode on bus-off entry	Mode transition for bus-off: Automatic invoking of CAN halt mode on bus-off end	Mode transition for bus-off: Invoking of CAN halt mode through software	Mode transition for bus-off: Transition to error-active state through software.
S124	☒	✓	☒	☒	☒	☒
S128	☒	✓	☒	☒	☒	☒
S1JA	☒	✓	☒	☒	☒	☒
S3A1	☒	✓	☒	☒	☒	☒
S3A3	☒	✓	☒	☒	☒	☒
S3A6	☒	✓	☒	☒	☒	☒
S3A7	☒	✓	☒	☒	☒	☒
S5D5	☒	✓	☒	☒	☒	☒
S5D9	☒	✓	☒	☒	☒	☒
S7G2	☒	✓	☒	☒	☒	☒

MCU Group	Monitoring of all CAN bus errors {Stuff, Form, ACK, 15-bit CRC, Bit error, ACK Delimiter}	Detection of all transition to error states {error-warning, error-passive, bus-off entry, and bus-off Recovery}	Support Reference clock selectable from 1-, 2-, 4- and 8-bit time periods	Support all 5 Interrupt Sources {Reception complete. Transmission complete, Receive FIFO, Transmit FIFO Error interrupts}	Support CAN sleep mode 1 {stop CAN clock}	Support all 3 software support units {Acceptance filter support, Mailbox search support, including receive mailbox search, transmit mailbox search, and message lost Search, Channel search support}
S124	✓	✓	☒	✓	☒	☒
S128	✓	✓	☒	✓	☒	☒
S1JA	✓	✓	☒	✓	☒	☒
S3A1	✓	✓	☒	✓	☒	☒
S3A3	✓	✓	☒	✓	☒	☒
S3A6	✓	✓	☒	✓	☒	☒
S3A7	✓	✓	☒	✓	☒	☒
S5D5	✓	✓	☒	✓	☒	☒
S5D9	✓	✓	☒	✓	☒	☒
S7G2	✓	✓	☒	✓	☒	☒

MCU Group	CAN Source Clock: PCLKB	CAN Source Clock: CANMCLK	Support all 3 Test Modes {Listen-only, Self-Test 1 (external loopback), Self-Test 2 (internal loopback)}	Module-stop Function	Support Standard CAN (11-bit)	Support Extended CAN (29 bit)
S124	N/A	✓	✓	✓	✓	✓
S128	N/A	✓	✓	✓	✓	✓
S1JA	N/A	✓	✓	✓	✓	✓
S3A1	✓	✓	✓	✓	✓	✓
S3A3	✓	✓	✓	✓	✓	✓
S3A6	✓	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓	✓
S5D5	✓	✓	✓	✓	✓	✓
S5D9	✓	✓	✓	✓	✓	✓
S7G2	✓	✓	✓	✓	✓	✓

14.9 Clock Generation Circuit (CGC)

The CGC HAL module provides high-level APIs for clock-control applications and is implemented on r_cgc. The CGC HAL module configures and controls the clock-control functions of a Synergy MCU using the clock-control peripheral. Since every project requires a clock function, the CGC HAL module is added to a project by default. (The module is configured in the ISDE.) A user-defined callback can be created to signal when the main oscillator has stopped.

CGC HAL Module Features

- Select the system clock source
  - HOCO (high-speed on-chip oscillator), MOCO (middle-speed on-chip oscillator)
  - LOCO (low-speed on-chip oscillator), Main Clock, PLL, or Sub-Oscillator
- Configure internal clocks and turn them on or off
- Configure the output clocks
- Set up the Oscillation Stop Detection feature
- Set up clock divisors on each of the up to six clock domains
- Some Synergy MCUs also support controllable external clock outputs, which may have independent divisors

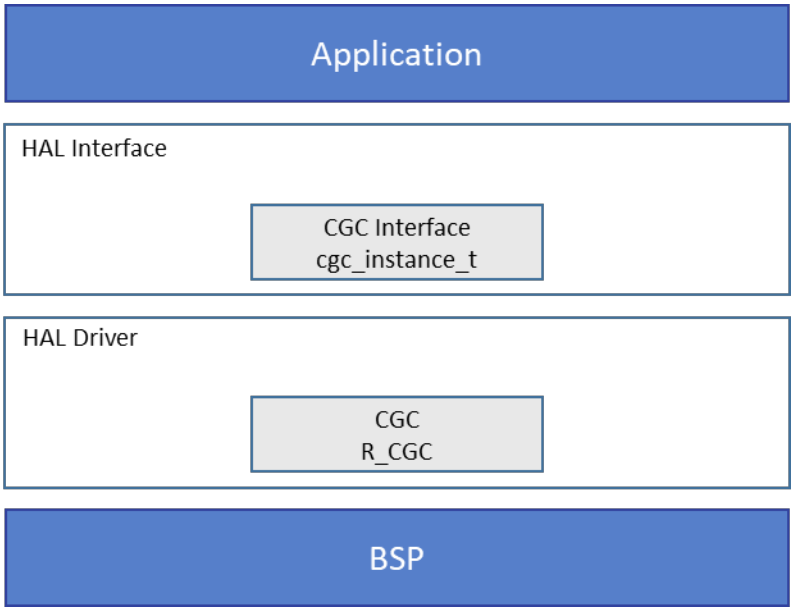


Figure 14.8 Clock Generation Circuit (CGC)

The following hardware features are, or are not, supported by SSP for the Clock Generation Circuit specifications for the clock sources.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	MOSC	SOSC	PLL Circuit	HOCO	MOCO	LOCO
S124	✓	✓	N/A	✓	✓	✓
S128	✓	✓	N/A	✓	✓	✓
S1JA	✓	✓	N/A	✓	✓	✓
S3A1	✓	✓	✓	✓	✓	✓
S3A3	✓	✓	✓	✓	✓	✓
S3A6	✓	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓	✓
S5D5	✓	✓	✓	✓	✓	✓
S5D9	✓	✓	✓	✓	✓	✓
S7G2	✓	✓	✓	✓	✓	✓

MCU Group	IWDTLOCO	JTAG External clock input	SWD External clock input
S124	✓	N/A	☒
S128	✓	N/A	☒
S1JA	✓	N/A	☒
S3A1	✓	✓	✓
S3A3	✓	✓	✓
S3A6	✓	✓	✓
S3A7	✓	✓	✓
S5D5	✓	✓	✓
S5D9	✓	✓	✓
S7G2	✓	✓	✓

14.10 Cyclic Redundancy Check calculator (CRC)

The CRC HAL module provides high-level APIs used to calculate 8, 16, and 32-bit CRC values on a block of data in memory or a stream of data over a Serial Communication Interface (SCI) channel using various types of industry-standard polynomials. The CRC HAL module is implemented on r_crc and uses the CRC peripheral on the Synergy MCU.

- CRC HAL module can calculate CRC on a block of data in memory.
- CRC HAL module can calculate CRC on a stream of data being transmitted or received over a serial communication Interface (SCI) channel (snoop mode).
- CRC HAL module supports the following 8 and 16 bit CRC polynomials which operates on 8-bit data in parallel
  - $X^8 + X^2 + X + 1$ (CRC-8)
  - $X^{16} + X^{15} + X^2 + 1$ (CRC-16)
  - $X^{16} + X^{12} + X^5 + 1$ (CRC-CCITT)
- CRC HAL module supports the following 32 bit CRC polynomials which operates on 32-bit data in parallel
  - $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$ (CRC-32)
  - $X^{32} + X^{28} + X^{27} + X^{26} + X^{25} + X^{23} + X^{22} + X^{20} + X^{19} + X^{18} + X^{14} + X^{13} + X^{11} + X^{10} + X^9 + X^8 + X^6 + 1$ (CRC-32C)
- CRC HAL module can calculate CRC with LSB first or MSB first bit order.

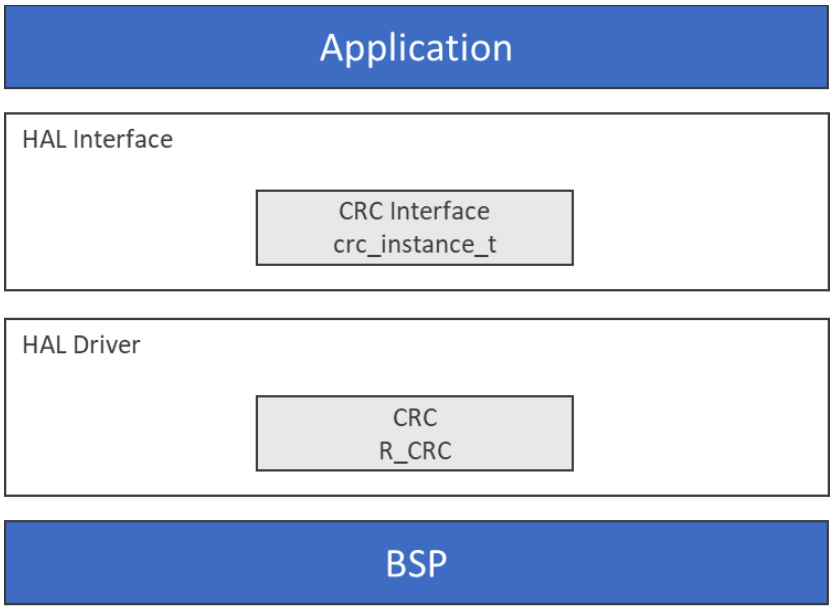


Figure 14.9 Cyclic Redundancy Check calculator

The following hardware features are, or are not, supported by SSP for the CRC.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Data Size 8 bit	Data Size 32 bit	CRC calculation switching	Module Stop Function	CRC Snoop
S124	✓	✓	✓	✓	✓
S128	✓	✓	✓	✓	✓
S1JA	✓	✓	✓	✓	✓
S3A1	✓	✓	✓	✓	✓
S3A3	✓	✓	✓	✓	✓
S3A6	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓
S5D5	✓	✓	✓	✓	✓
S5D9	✓	✓	✓	✓	✓
S7G2	✓	✓	✓	✓	✓

14.11 Capacitive Touch Sensing Unit (CTSU)

The Capacitive Touch Sensing Unit (CTSU) driver supports the CTSU peripheral on the Synergy Microcontrollers. The CTSU driver provides the functionality necessary to open, close, run and control the CTSU peripheral depending upon the configuration passed as arguments. The driver initializes the CTSU peripheral to detect a change in capacitance on any of the configured (and enabled) channels, perform requisite filtering, and generate a variety of data that can be used by higher level framework layers for CTSU button, wheel, and slider. To support the different types of data required by these layers, the implementation provides a function that allows upper level layers to read different types of processed data based on their need. The driver will scan the configured channels, move data using the DTC, perform filtering, drift compensation, auto-tuning and notify the user via a callback once each iteration is completed and new processing data is available. These callbacks can be used by upper layers to read the data.

This module has been designed to be used together with the CTW for Synergy tool, which generates the required structures for initialization and operation. The driver also allows the user to provide their own filtering and auto-tuning algorithms and integrate it into the process. The driver can only support one configuration at a time, but the user can reopen the driver with multiple channel configurations as required by the application.

The CTSU driver allows the user to configure the CTSU channels for all the supported operation modes including Mutual and Self Capacitance.

The CTSU driver implements the CTSU interface in SSP:

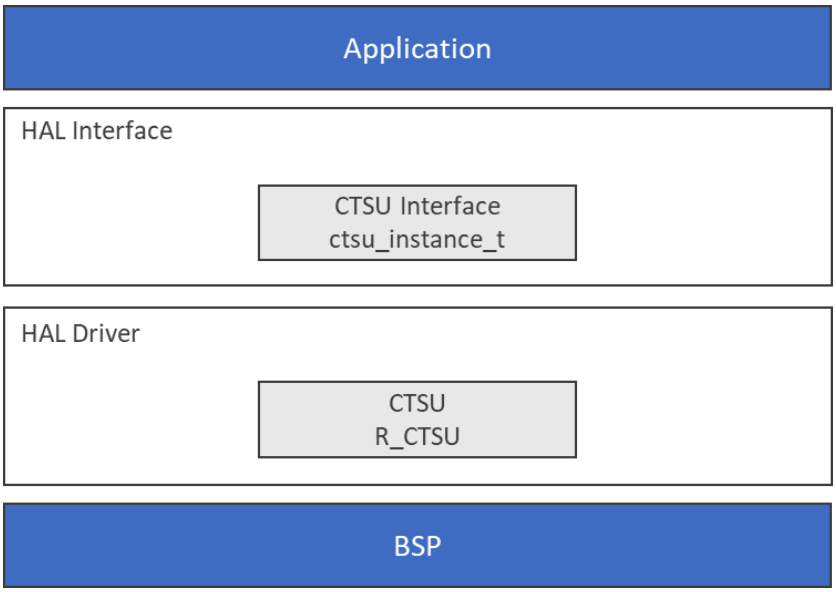


Figure 14.10 Capacitive Touch Sensing Unit

The following hardware features are, or are not, supported by SSP for the CTSU.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Self-Cap Single Scan Mode	Self-Cap Multi Scan Mode	Mutual-Cap Full Scan Mode	Sensor Stabilization Wait Time and Measurement Time	CTSU Interrupts	ELC Linking
S124	✓	✓	✓	✓	✓	N/A
S128	✓	✓	✓	✓	✓	N/A
S1JA	☒	☒	☒	☒	☒	N/A
S3A1	☒	☒	☒	☒	☒	N/A
S3A3	✓	✓	✓	✓	✓	N/A
S3A6	☒	☒	☒	☒	☒	N/A
S3A7	✓	✓	✓	✓	✓	N/A
S5D5	✓	✓	✓	✓	✓	N/A
S5D9	✓	✓	✓	✓	✓	N/A
S7G2	✓	✓	✓	✓	✓	N/A

14.12 Digital to Analog (DAC)

The DAC HAL module provides high-level APIs for digital-to-analog conversion applications implemented on r_dac. The DAC HAL module supports a dual-channel 12-bit D/A converter (DAC12) peripheral on Synergy MCUs.

This module configures the dual-channel 12-bit D/A Converter (DAC12) to output one of 4096 voltage levels between the positive and negative reference voltages. The DAC HAL Module includes configuration settings to:

- Set either a left-justified or right-justified 12-bit value format for the 16-bit input data registers
- Enable or disable output amplifiers
- Operate in synchronous anti-interference mode with the Analog-to-Digital Converter (ADC) module.

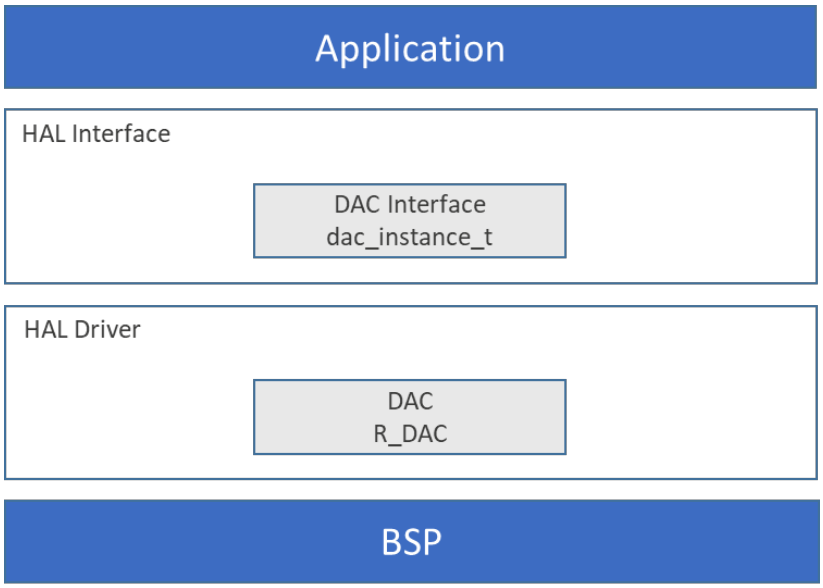


Figure 14.11 12-bit Digital to Analog Converter

The following hardware features are, or are not, supported by SSP for the DAC.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	12-Bit	2 Channels Output	Module Stop Function	Event link function through ELC HAL driver *
S124	✓	N/A	✓	☒
S128	N/A	N/A	N/A	☒
S1JA	N/A	N/A	N/A	N/A
S3A1	✓	N/A	✓	☒
S3A3	✓	N/A	✓	☒
S3A6	✓	N/A	✓	☒
S3A7	✓	✓	✓	☒
S5D5	✓	✓	✓	☒
S5D9	✓	✓	✓	☒
S7G2	✓	✓	✓	☒

* Note: The ELC event could be used instead of calling the DAC `start()` interface. This would have to be programmed by the user by setting up the link rather than using the ELC API.

14.13 Digital to Analog 8-bit (DAC8)

The DAC 8 HAL module provides high-level APIs for digital-to-analog conversion applications implemented on r_dac8. The DAC 8 HAL module supports an 8-bit D/A converter (DAC 8) peripheral on applicable Synergy MCUs.

DAC 8 HAL Module Features:

- Available on Synergy S1JA, S128, S3A3, and S3A6 MCUs
- 8-Bit D/A Converter with two channels for S1JA, S3A3, and S3A6 and three channels for S128
- Left-Justified or Right-Justified Input Data Format
- Synchronization with the Analog-to-Digital Converter (ADC) module
- Multiple Operation Modes
  - Normal
  - Real-Time (Event Link)
- Charge Pump Control

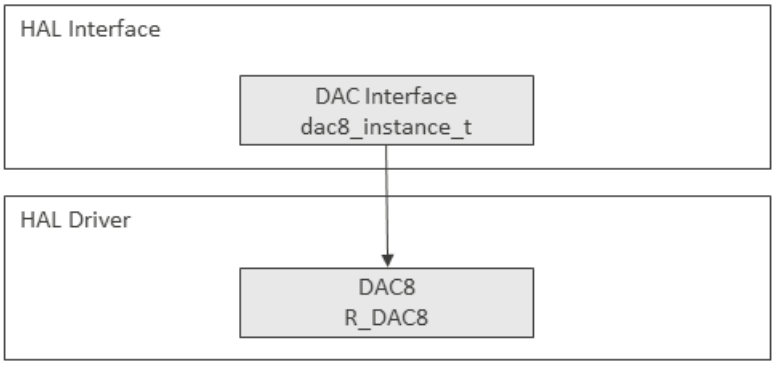


Figure 14.12 8-bit Digital to Analog Convertor

The following hardware features are, or are not, supported by SSP for the DAC8:

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

	8-Bit	2 Channels Output	3 Channel Output	Module Stop Function	Event link function through ELC HAL driver *
S124	N/A	N/A	N/A	N/A	N/A

	8-Bit	2 Channels Output	3 Channel Output	Module Stop Function	Event link function through ELC HAL driver *
S128	✓	✓	✓	✓	✓
S1JA	✓	✓	N/A	✓	✓
S3A1	N/A	N/A	N/A	☒	☒
S3A3	✓	✓	N/A	✓	✓
S3A6	☒	☒	N/A	☒	☒
S3A7	N/A	N/A	N/A	N/A	N/A
S5D5	N/A	N/A	N/A	N/A	N/A
S5D9	N/A	N/A	N/A	N/A	N/A
S7G2	N/A	N/A	N/A	N/A	N/A

* Note: The ELC event could be used instead of calling the DAC `start()` interface. This would have to be programmed by the user by setting up the link rather than using the ELC API.

14.14 Direct Memory Access Controller (DMAC)

The Direct Memory Access Controller or DMAC HAL module provides high-level APIs for data-transfer applications and is implemented on r_dmac. The DMAC HAL module moves data from a user-specified source to a user-specified destination when an interrupt or event occurs. The DMAC HAL module uses the DMAC peripheral on the Synergy MCU. A user-defined callback can be created to inform the CPU when transfer events occur.

DMAC HAL Module Features

- DMAC module on a Synergy MCU
- Interrupts, if desired
- Multiple transfer modes
  - Single Transfer
  - Repeat Transfer
  - Block Transfer
  - Address increment or fixed modes
- Multiple channels, with the number depending on the MCU used

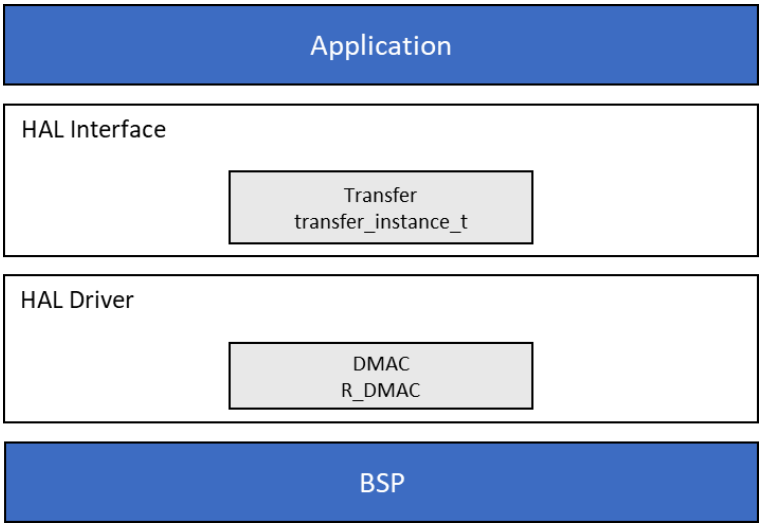


Figure 14.13 DMA Controller

The following hardware features are, or are not, supported by SSP for DMAC.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Normal Transfer Mode	Repeat Transfer Mode	Block Transfer Mode	Extended repeat area function	Event link function through ELC HAL driver
S124	N/A	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	N/A	N/A
S1JA	N/A	N/A	N/A	N/A	N/A
S3A1	✓	✓	✓	✓	☒
S3A3	✓	✓	✓	✓	☒
S3A6	✓	✓	✓	✓	☒
S3A7	✓	✓	✓	✓	☒
S5D5	✓	✓	✓	✓	☒
S5D9	✓	✓	✓	✓	☒
S7G2	✓	✓	✓	✓	☒

14.15 Data Operation Circuit (DOC)

The Data Operation Circuit (DOC) HAL module provides high-level APIs for DOC applications and is implemented on r_doc. The DOC HAL module uses the DOC peripherals on the Renesas Synergy™ MCU device. A user-defined callback can be created to inform the CPU when an event occurs.

DOC HAL Module Features

The DOC HAL module peripheral is used to compare 16-bit data and can detect the following events:

- A mismatch or match between data values
- Overflow of an addition operation
- Underflow of a subtraction operation

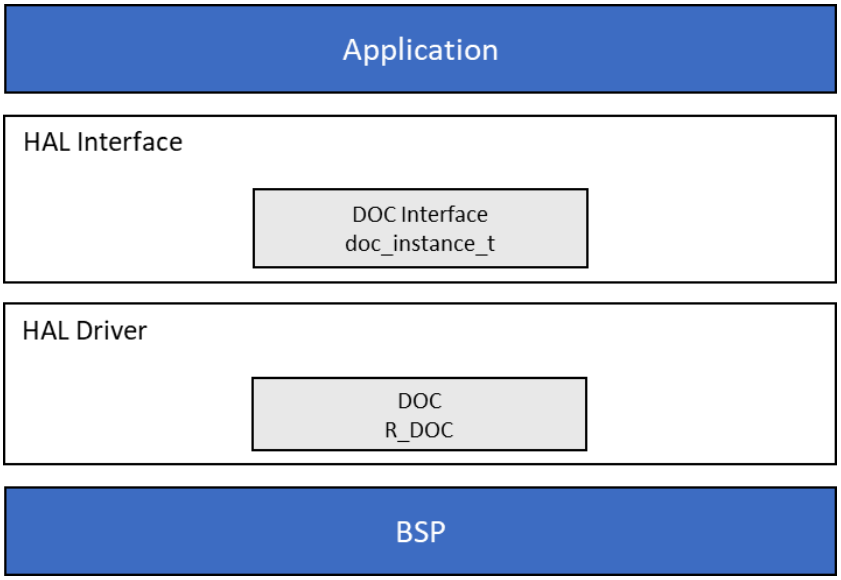


Figure 14.14 Data Operation Circuit

The following hardware features are, or are not, supported by SSP for DOC.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Module-stop function	Event link function through ELC HAL driver
S124	✓	☒
S128	✓	☒
S1JA	✓	☒
S3A1	✓	☒
S3A3	✓	☒
S3A6	✓	☒
S3A7	✓	☒
S5D5	✓	☒
S5D9	✓	☒
S7G2	✓	☒

14.16 Data Transfer Controller (DTC)

The Data Transfer Controller (DTC) HAL module provides high-level APIs for data-transfer applications and is implemented on r_dtc. The DTC HAL module moves data from a user-specified source to a user-specified destination when an interrupt or event occurs. The module uses the DTC peripheral on the Synergy MCU. A user-defined callback can be created to inform the CPU when transfer events occur.

The DTC HAL Module supports the following features:

- Moves data from a user-specified source to a user-specified destination when an interrupt or event occurs
- Supports the DTC module on a Synergy MCU
- Interrupts, if needed
- Multiple transfer modes
  - Single transfer
  - Repeat transfer
  - Block transfer
  - Address increment or fixed modes
  - Chain transfers
- Multiple channels (depending on selected implementation)
  - Number of channels is limited only by the size of the RAM-based vector table

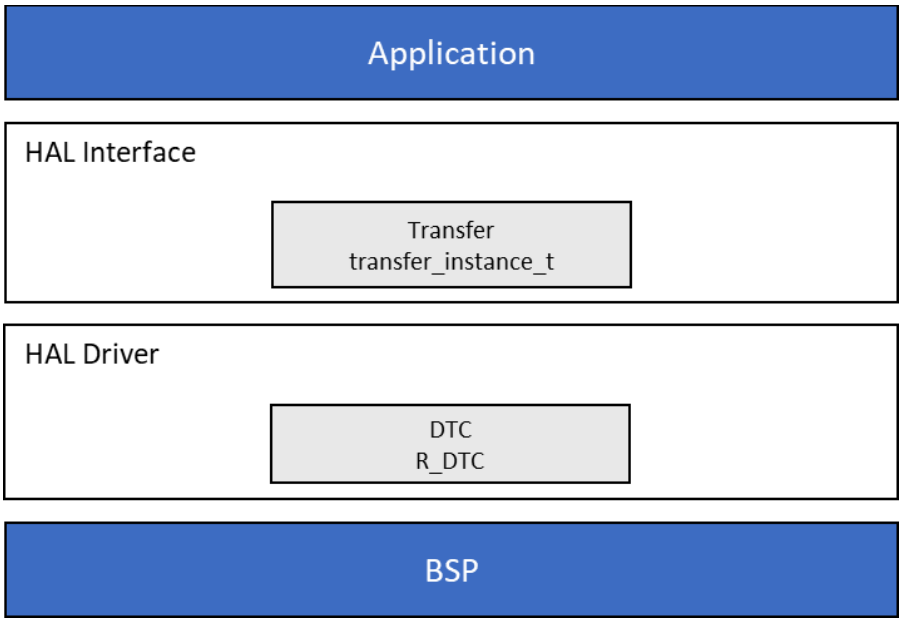


Figure 14.15 Data Transfer Controller HAL Interface

The following hardware features are, or are not, supported by SSP for DTC.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Normal Transfer Mode	Repeat Transfer Mode	Block Transfer Mode	Selectable Data Transfer Units (8 bits, 16 bits, 32 bits)	API/Config for Event link function upon completion of DTC transfers
S124	✓	✓	✓	✓	☒
S128	✓	✓	✓	✓	☒
S1JA	✓	✓	✓	✓	☒
S3A1	✓	✓	✓	✓	☒
S3A3	✓	✓	✓	✓	☒
S3A6	✓	✓	✓	✓	☒
S3A7	✓	✓	✓	✓	☒
S5D5	✓	✓	✓	✓	☒
S5D9	✓	✓	✓	✓	☒
S7G2	✓	✓	✓	✓	☒

14.17 Event Link Controller (ELC)

The Event Link Controller (ELC) HAL module provides high-level APIs for ELC HAL applications and is implemented on r_elc. The ELC HAL module uses the ELC peripheral on the Synergy MCU. There are no callbacks associated with the ELC HAL module. The project configurator in the e^2 studio ISDE includes the ELC HAL module in every project by default. To configure the ELC HAL module, select it in the HAL/Common module in the Threads tab, and then in the HAL/Common Stacks window.

The ELC Module Functions

- Creates an event link between two blocks.
- Breaks that event link between two blocks.
- Generates one of two software events which interrupt the CPU.

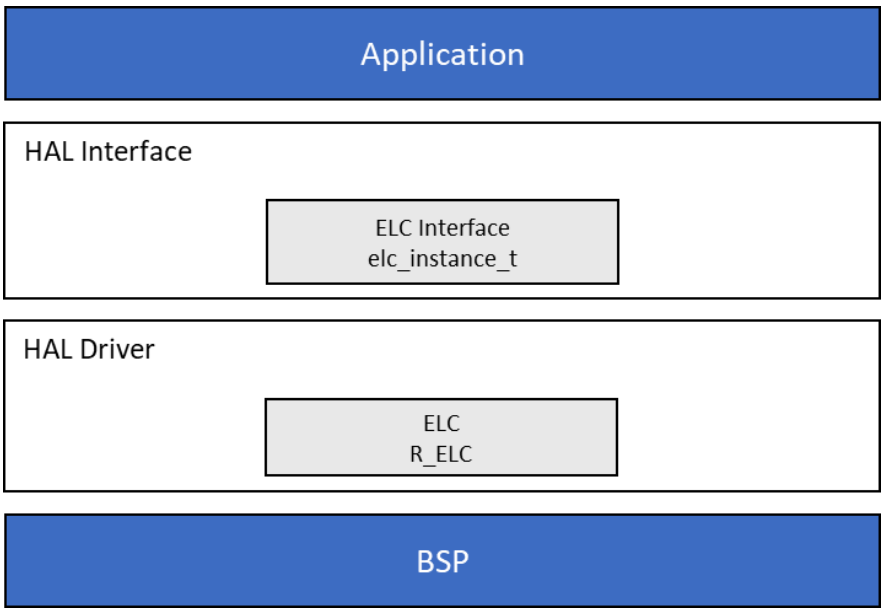


Figure 14.16 Event Link Controller (ELC)

The following hardware features are, or are not, supported by SSP for ELC.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Create	Break	Generate
S124	✓	✓	✓
S128	✓	✓	✓
S1JA	✓	✓	✓
S3A1	✓	✓	✓
S3A3	✓	✓	✓
S3A6	✓	✓	✓
S3A7	✓	✓	✓
S5D5	✓	✓	✓
S5D9	✓	✓	✓
S7G2	✓	✓	✓

14.18 Flash Memory

There are two separate Flash modules: r_flash_lp and r_flash_hp. The High-Performance Flash module (Flash_HP) is used for programming the S7 and S5 family of MCUs. The Low-Power Flash module (Flash_LP) is used for programming the S3 and S1 family of MCUs. The two are not interchangeable, although the APIs and other features of the modules are very similar.

The Flash HAL modules APIs allow an application to read, write, and erase both the data and ROM flash areas that reside within the MCU. The amount of flash memory available varies across MCU parts, but the API functions apply to all devices.

The Flash HAL Module Features

- Block erasing, reading, writing, and blank checking of code flash (ROM).
- Both blocking and non-blocking erasing, reading, writing, and blank checking of data flash.
- Blocking erasing, reading, writing, and blank checking of code flash.
- Callback functions for completion of non-blocking data-flash operations.
- Access window (write protection) for ROM Flash, allowing only specified areas of code flash to be erased or written.
- Boot block-swapping which allows safe rewriting of the startup program without first erasing it.

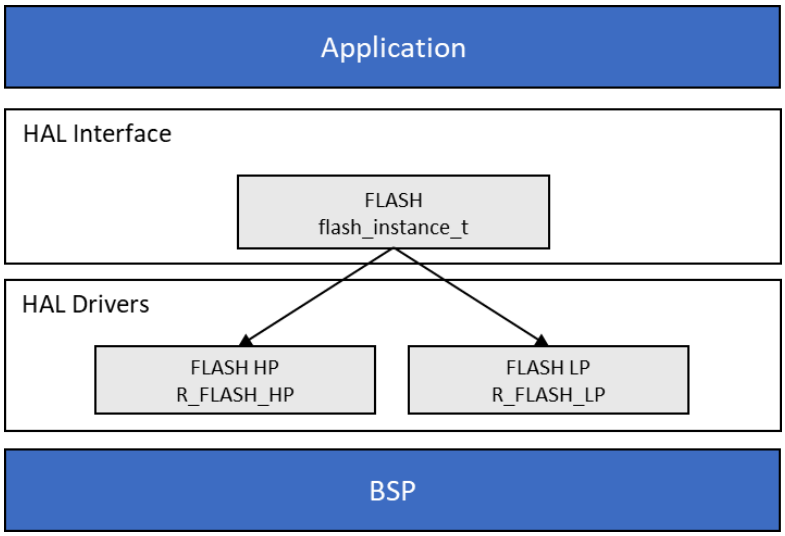


Figure 14.17 Flash Memory-HAL Interface

The following hardware features are, or are not, supported by SSP for the Flash_HP.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Programming by dedicated flash-memory programmer through a serial interface (serial programming)	Programming of flash memory by user program (self-programming).	Background operations (BGOs)
S124	N/A	N/A	N/A
S128	N/A	N/A	N/A
S1JA	N/A	N/A	N/A
S3A1	N/A	N/A	N/A
S3A3	N/A	N/A	N/A
S3A6	N/A	N/A	N/A
S3A7	N/A	N/A	N/A
S5D5	✓	✓	✓
S5D9	✓	✓	✓
S7G2	✓	✓	✓

The following hardware features are, or are not, supported by SSP for the Flash_LP.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Programming by dedicated flash-memory programmer through a serial interface (serial programming)	Programming of flash memory by user program (self-programming).	Background operations (BGOs)
S124	✓	✓	✓
S128	✓	✓	✓
S1JA	✓	✓	✓
S3A1	✓	✓	✓
S3A3	✓	✓	✓
S3A6	✓	✓	✓
S3A7	✓	✓	✓
S5D5	N/A	N/A	N/A
S5D9	N/A	N/A	N/A
S7G2	N/A	N/A	N/A

14.19 Factory Microcontroller Information (FMI)

The FMI HAL module provides high-level APIs for applications that read records from the Factory MCU Information Flash Table. The FMI HAL module is implemented on r_fmi. The FMI HAL module uses the Flash Interface on the Synergy MCU.

FMI HAL Module Features

The FMI HAL module reads the FMIFRT (Factory MCU Information Flash Root Table) on a Synergy microcontroller, looking up the address of the start of the table in flash. The module sets the caller's pointer to the Product Information record from the table. This information may be used to determine the capabilities of features specific to this MCU package. Information available from the FMI HAL module includes:

- Product Information: Product Name, Package, Pin count, and Temperature range
- Product Features: Version major, Version minor, and Variant data
- Event Information such as Interrupts and Events

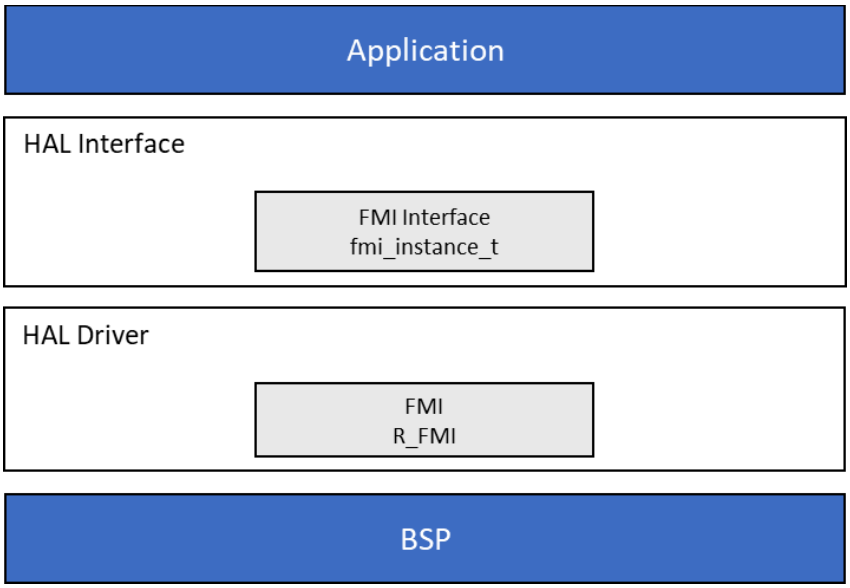


Figure 14.18 FMI HAL Module

14.20 Graphics LCD Controller (GLCD)

The Graphics LCD Controller (GLCDC) HAL module provides high-level APIs for GLCDC applications and is implemented on r_glcd. The GLCDC HAL module uses the Graphics LCD Driver peripheral on the Synergy MCU. A user-defined callback can be created to handle frame buffer switching and underflow detection.

GLCDC HAL Module Features

- LCD panels with RGB interface (up to 24 bits) and sync signals (HSYNC, VSYNC, and Data Enable (optional)
- Various color formats for input graphics planes (RGB888, ARGB8888, RGB565, ARGB1555, ARGB4444, CLUT8, CLUT4, CLUT1)
- the Color Look-Up Table (CLUT) usage for input graphics planes ARGB8888 with 512 words (32 bits/word)
- Various color formats for output (RGB888, RGB666, RGB565, Serial RGB888)
- Can input two graphics planes on top of the background plane and blend them on the screen
- Generates a dot clock to the panel. The clock source is selectable from internal or external (LCD_EXTCLK)
- Brightness adjustment, contrast adjustment and gamma correction
- GLCDC interrupts to handle frame-buffer switching or underflow detection

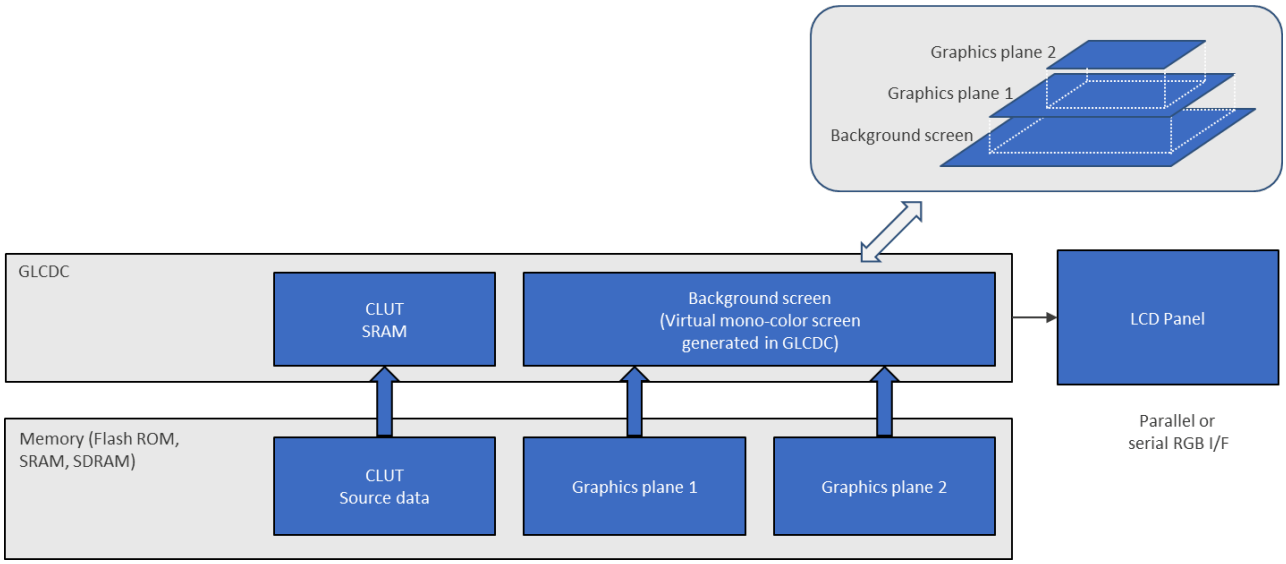


Figure 14.19 GCLD Controller Data Flow

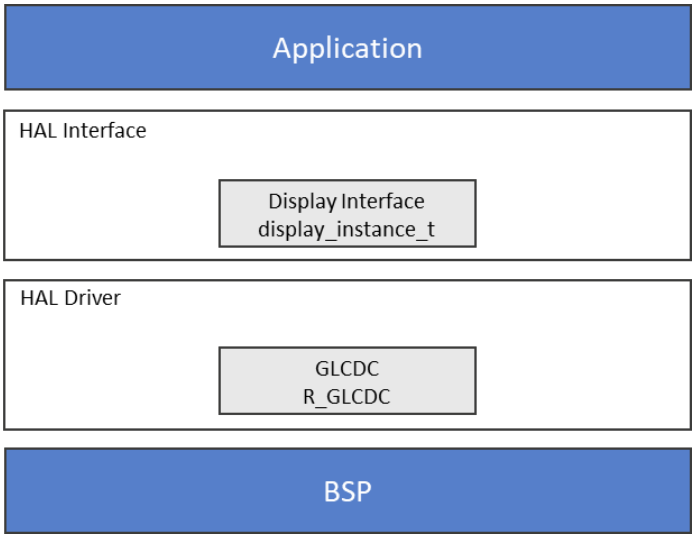


Figure 14.20 GCLD Controller

The following hardware features are, or are not, supported by SSP for GLCD:

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Single Color Background Plane	Graphics 1 Plane	Graphics 2 Plane	Support 16 bit per pixel graphics	Support 32 bit per pixel graphics	Support 1 bit LUT
S124	N/A	N/A	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	N/A	N/A	N/A
S1JA	N/A	N/A	N/A	N/A	N/A	N/A
S3A1	N/A	N/A	N/A	N/A	N/A	N/A
S3A3	N/A	N/A	N/A	N/A	N/A	N/A
S3A6	N/A	N/A	N/A	N/A	N/A	N/A
S3A7	N/A	N/A	N/A	N/A	N/A	N/A
S5D5	N/A	N/A	N/A	N/A	N/A	N/A
S5D9	✓	✓	✓	✓	✓	✓
S7G2	✓	✓	✓	✓	✓	✓

MCU Group	Support 4-bit LUT	Support 8-bit LUT	Support All Pixel formats	Supports Alpha Blending	Video Signal Timing Adjustment	Supports All Output Data formats
S124	N/A	N/A	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	N/A	N/A	N/A
S1JA	N/A	N/A	N/A	N/A	N/A	N/A
S3A1	N/A	N/A	N/A	N/A	N/A	N/A
S3A3	N/A	N/A	N/A	N/A	N/A	N/A
S3A6	N/A	N/A	N/A	N/A	N/A	N/A
S3A7	N/A	N/A	N/A	N/A	N/A	N/A
S5D5	N/A	N/A	N/A	N/A	N/A	N/A
S5D9	✓	✓	✓	✓	✓	✓
S7G2	✓	✓	✓	✓	✓	✓

MCU Group	Supports All Dithering Modes	Support output of VSYNC, HSYNC and Horizontal Data Enable	Supports Brightness and Contrast	Supports Gamma Correction
S124	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	N/A
S1JA	N/A	N/A	N/A	N/A
S3A1	N/A	N/A	N/A	N/A
S3A3	N/A	N/A	N/A	N/A
S3A6	N/A	N/A	N/A	N/A
S3A7	N/A	N/A	N/A	N/A
S5D5	N/A	N/A	N/A	N/A
S5D9	✓	✓	✓	✓
S7G2	✓	✓	✓	✓

14.21 General Purpose Timer (GPT)

The General PWM Timer (GPT) HAL module provides high-level APIs for timer applications and is implemented on r_gpt. The GPT HAL module uses the GPT peripheral on the Synergy MCU. A user-defined callback can be created to respond to a timer event.

GPT HAL Module Features

The GPT HAL module configures a timer to a user-specified period. When the period elapses, any of the following events can occur:

- CPU interrupt that calls a user callback function, if provided
- Toggle a port pin
- Data transfer using DMAC/DTC if configured with Transfer Interface
- Starting of another peripheral if configured with events and peripheral definitions

General PWM Timer (GPT) supports:

- PCLKD as core clock
  - — Two output pins per channel
  - — Up-counting saw/triangle waves
  - — Two output compare and input capture registers
  - — Can be configured as a response to 8 Event Link Controllers (ELCs), and as an event in ELC

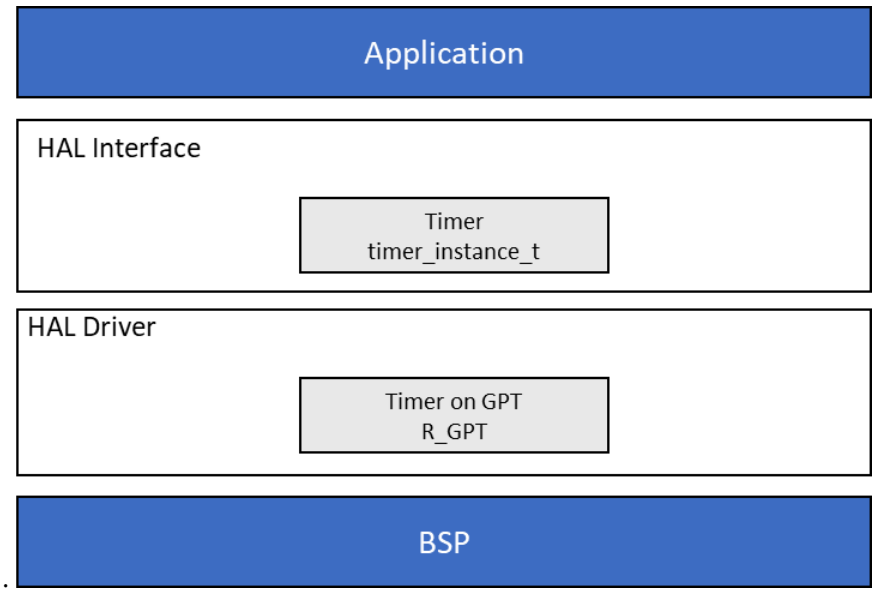


Figure 14.21 General Purpose Timer

The following hardware features are, or are not, supported by SSP for GPT.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Saw Waves	Triangle Waves	PWM waveform for controlling brushless DC motors	Compare match output for Low, High, and Toggle	Input capture function	Automatic addition of dead time
S124	☒	✓	☒	✓	☒	☒
S128	☒	✓	☒	✓	☒	☒
S1JA	☒	✓	☒	✓	☒	☒
S3A1	☒	✓	☒	✓	☒	☒
S3A3	☒	✓	☒	✓	☒	☒
S3A6	☒	✓	☒	✓	☒	☒
S3A7	☒	✓	☒	✓	☒	☒
S5D5	☒	✓	☒	✓	☒	☒
S5D9	☒	✓	☒	✓	☒	☒
S7G2	☒	✓	☒	✓	☒	☒

MCU Group	PWM Mode	Phase Count Function	One-Shot Operation	Event link function through ELC HAL driver	Noise filtering function
S124	✓	☒	✓	☒	☒
S128	✓	☒	✓	☒	☒
S1JA	✓	☒	✓	☒	☒
S3A1	✓	☒	✓	☒	☒
S3A3	✓	☒	✓	☒	☒
S3A6	✓	☒	✓	☒	☒
S3A7	✓	☒	✓	☒	☒
S5D5	✓	☒	✓	☒	☒
S5D9	✓	☒	✓	☒	☒
S7G2	✓	☒	✓	☒	☒

14.22 Independent Watchdog Timer (IWDT)

The Independent Watchdog Timer (IWDT) HAL module provides high-level APIs for watchdog timer applications and is implemented on r_iwdt. The Watchdog Timer uses the IWDT peripheral on the Synergy MCU. A user-defined callback can be created to respond to event notifications.

IWDT HAL Module Features

- When the WDT underflows or is refreshed outside of the permitted refresh window, one of the following events can occur:
  - — Device reset
  - — NMI generation
- Supports the internal Watchdog timer peripheral (IWDT), which has its own clock source which improves safety.
- Supports automatic hardware configuration after reset.

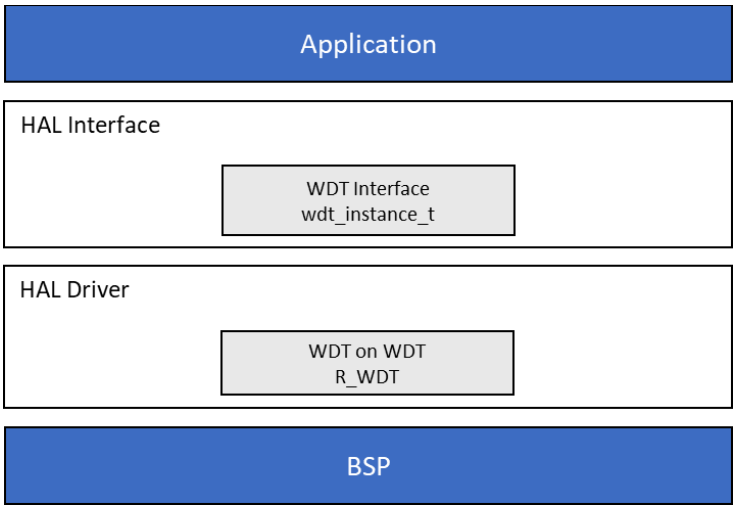


Figure 14.22 Independent Watchdog Timer (IWDT)

The following hardware features are, or are not, supported by SSP for IWDT:

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Count down	Auto-start mode	Reset output	Interrupt request output
S124	✓	✓	✓	✓
S128	✓	✓	✓	✓
S1JA	✓	✓	✓	✓
S3A1	✓	✓	✓	✓
S3A3	✓	✓	✓	✓
S3A6	✓	✓	✓	✓
S3A7	✓	✓	✓	✓
S5D5	✓	✓	✓	✓
S5D9	✓	✓	✓	✓
S7G2	✓	✓	✓	✓

MCU Group	Sleep mode count stop control output	Event link function through ELC HAL driver	Window function	Conditions for stopping the Counter – reset/underflow-refresh error	Refresh error and under flow error detect	Reading the counter value
S124	☒	☒	✓	✓	✓	✓
S128	☒	☒	✓	✓	✓	✓
S1JA	☒	☒	✓	✓	✓	✓
S3A1	☒	☒	✓	✓	✓	✓
S3A3	☒	☒	✓	✓	✓	✓
S3A6	☒	☒	✓	✓	✓	✓
S3A7	☒	☒	✓	✓	✓	✓
S5D5	☒	☒	✓	✓	✓	✓
S5D9	☒	☒	✓	✓	✓	✓
S7G2	☒	☒	✓	✓	✓	✓

MCU Group	Selecting the clock frequency division ratio after a reset	Selecting the timeout period of the independent watchdog timer
S124	☒	✓ Set by the OFS registers in the BSP
S128	☒	✓ Set by the OFS registers in the BSP
S1JA	☒	✓ Set by the OFS registers in the BSP
S3A1	☒	✓ Set by the OFS registers in the BSP
S3A3	☒	✓ Set by the OFS registers in the BSP
S3A6	☒	✓ Set by the OFS registers in the BSP
S3A7	☒	✓ Set by the OFS registers in the BSP
S5D5	☒	✓ Set by the OFS registers in the BSP
S5D9	☒	✓ Set by the OFS registers in the BSP
S7G2	☒	✓ Set by the OFS registers in the BSP

14.23 Input Capture (GPT_INPUT_CAPTURE)

The Input Capture HAL module provides high-level APIs used for measuring input pulse-widths which is implemented on r_gpt_input_capture. The Input Capture HAL module configures the input capture parameters to use with the GPT peripheral on the Synergy MCU. A user-defined callback can be created to acquire the value each time a new measurement is complete.

The Input Capture HAL module configures the GPT for an input capture function.

- The Input Capture HAL allows the user to perform the following tasks:
  - Initialize the module
  - Enable input capture measurement
  - Disable input capture measurement
  - Get the status (running or not) of the measurement counter
  - Get the last captured timer/overflows counter value
  - Close the input capture operation
- The Input Capture HAL module supports:
  - Pulse-width measurement only
  - Rising-edge or falling-edge measurement start
  - One-shot or periodic mode
  - Hardware-enable signals to enable captures (low enable/high enable)
  - Callback function with the following events:
    - Counter overflow
    - Input capture occur
- There is a callback structure that provides data on the interrupting event, including which interrupt occurs and the associated counter values.

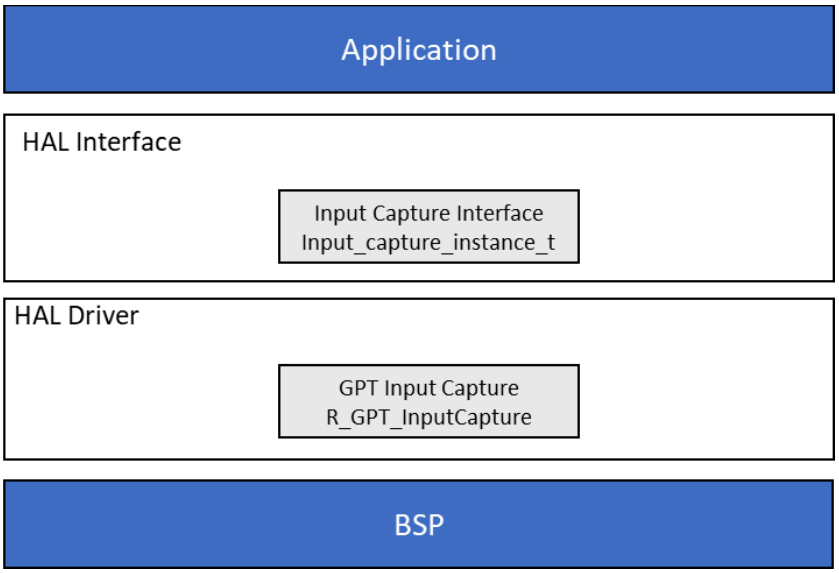


Figure 14.23 Input Capture HAL Module

The following hardware features are, or are not, supported by SSP for GPT_INPUT_CAPTURE.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Saw Waves	Triangle Waves *	PWM waveform for controlling brushless DC motors	Compare match output for Low, High, and Toggle	Input capture function	Automatic addition of dead time
S124	☒	☒	☒	☒	✓	☒
S128	☒	☒	☒	☒	✓	☒
S1JA	☒	☒	☒	☒	✓	☒
S3A1	☒	☒	☒	☒	✓	☒
S3A3	☒	☒	☒	☒	✓	☒
S3A6	☒	☒	☒	☒	✓	☒
S3A7	☒	☒	☒	☒	✓	☒
S5D5	☒	☒	☒	☒	✓	☒
S5D9	☒	☒	☒	☒	✓	☒
S7G2	☒	☒	☒	☒	✓	☒

MCU Group	PWM Mode	Phase Count Function	Event link function through ELC HAL driver	Noise filtering function
S124	☒	☒	☒	✓
S1JA	☒	☒	☒	✓
S3A1	☒	☒	☒	✓
S3A7	☒	☒	☒	✓
S5D9 or S7G2	☒	☒	☒	✓

14.24 Interrupt Controller Unit – External (ICU)

The External IRQ HAL module is an API for configuring and using external IRQ pins on Synergy MCUs. The External IRQ HAL module is implemented on r_icu and uses the Interrupt Controller Unit (ICU) of the Synergy MCU.

External IRQ HAL Module Features

- External interrupt pins available on the target Synergy MCU
- Multiple function options:
  - Enabling and disabling generation of an interrupt
  - Enabling and disabling the IRQ noise filter
  - Setting external pin IRQ trigger (Rising edge, falling edge or low level on the IRQ pin)
- Configuring a user callback function, which will be invoked by the HAL module when an external pin interrupt is generated.

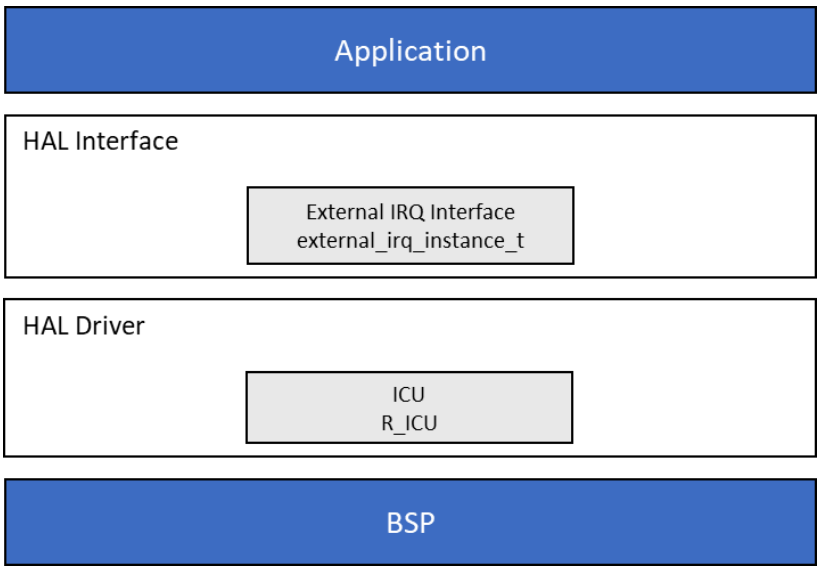


Figure 14.24 External Interrupt Controller Unit (ICU)

The following hardware features are, or are not, supported by SSP for ICU.

MCU Group	Peripheral function interrupts	External pin interrupts	DTC and DMAC control	Interrupt sources for NVIC	Non-maskable interrupts (see notes)	Return from low-power mode (see notes)
S124	✓	✓	✓	✓	✓	✓
S128	✓	✓	✓	✓	✓	✓
S1JA	✓	✓	✓	✓	✓	✓
S3A1	✓	✓	✓	✓	✓	✓
S3A3	✓	✓	✓	✓	✓	✓
S3A6	✓	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓	✓
S5D5	✓	✓	✓	✓	✓	✓
S5D9	✓	✓	✓	✓	✓	✓
S7G2	✓	✓	✓	✓	✓	✓

Notes:
- The ICU module in SSP (r_icu) handles only external pin interrupts and not other features above.
- Peripheral function interrupts are controlled by BSPs for each MCU and each peripheral driver modules in SSP.
- DTC or DMA control is handled by DTC or DMAC module in SSP (r_dtc or r_dmac).

— Non-maskable interrupts supported in SSP are IWDT Underflow, WDT Underflow and Voltage Monitor Interrupts. Those NMIs are controlled by IWDT, WDT or LVD modules (r_iwdt, r_wdt or r_lvd), respectively.
— LVD module (r_lvd) supports the Wake Up Interrupt Enable setting. For low power mode details, see the LPM section.

MCU Group	RPEST	RECCST	BUSSST	BUSMST	SPEST
S124	☒	N/A	N/A	N/A	N/A
S128	☒	N/A	N/A	N/A	N/A
S1JA	☒	☒	☒	☒	☒
S3A1	☒	☒	☒	☒	☒
S3A3	☒	☒	☒	☒	☒
S3A6	☒	☒	☒	☒	☒
S3A7	☒	☒	☒	☒	☒
S5D5	☒	☒	☒	☒	☒
S5D9	☒	☒	☒	☒	☒
S7G2	☒	☒	☒	☒	☒

14.25 I/O Port (GPIO / IOPORT)

The I/O Port HAL module provides high-level APIs for controlling I/O pins and is implemented on `r_ioport`. The I/O Port HAL module configures MCU pins and provides functions that manipulate them. The operating state of the I/O pins can be set via the Synergy configurator. When the Synergy project is built, a pin configuration file is created, and when the application runs, the BSP configures the I/O port accordingly, using the applicable APIs.

I/O Port HAL Module Features

This module configures one or more I/O pins. The direction of the pin or pins can be configured along with a number of other options provided as follows:

- Pull-up
- NMOS/PMOS
- Drive strength
- Event edge trigger (falling, rising or both)
- Whether the pin is to be used as an IRQ pin
- Whether the pin is to be used as an analog pin
- Whether the pin is to be used as a peripheral pin and which peripheral

The module also provides the following functionality:

- Changes the direction of one or more pins on a port
- Writes to one or more pins on a port
- Reads from one or more pins on a port
- Sets event output data
- Reads event input data

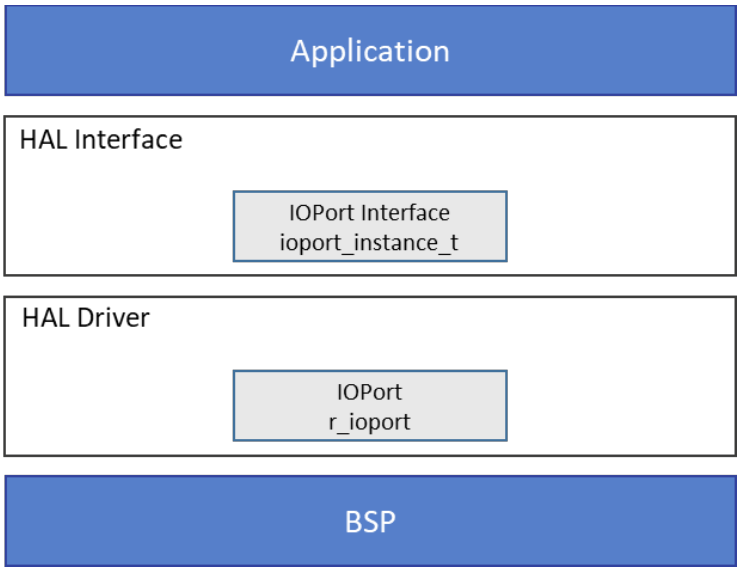


Figure 14.25 GPIO / IOPort

The following hardware features are, or are not, supported by SSP for GPIO.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Port Direction Setting	Input Data Read function	Output Port Write function	Pin Mode Control	Ethernet Mode Configuration	ELC_PORTn Event Input Read function*
S124	✓	✓	✓	✓	N/A	✓
S128	✓	✓	✓	✓	N/A	✓
S1JA	✓	✓	✓	✓	N/A	✓
S3A1	✓	✓	✓	✓	N/A	✓
S3A3	✓	✓	✓	✓	N/A	✓
S3A6	✓	✓	✓	✓	N/A	✓
S3A7	✓	✓	✓	✓	N/A	✓
S5D5	✓	✓	✓	✓	✓	✓
S5D9	✓	✓	✓	✓	✓	✓
S7G2	✓	✓	✓	✓	✓	✓

MCU Group	ELC_PORTn Event Output Setting *
S124	✓
S128	✓
S1JA	✓
S3A1	✓
S3A3	✓
S3A6	✓
S3A7	✓
S5D5	✓
S5D9	✓
S7G2	✓

* Note: Event Linking would have to be set up by the user, rather than using the ELC API.

14.26 JPEG Decode

The JPEG Decode HAL module provides high-level APIs for decoding JPEG images. The JPEG Decode HAL module uses the Synergy MCU JPEG Codec peripheral. A user callback function is available to inform the application program of key processing events.

The JPEG Decode Module supports the following features:

- Supports JPEG decompression.
- Supports polling mode that allows an application to wait for JPEG Decoder to complete.
- Supports interrupt mode with user-supplied callback functions.
- Configures parameters such as horizontal and vertical subsample values, horizontal stride, decoded pixel format, input and output data format, and color space.
- Obtains the size of the image prior to decoding it.
- Supports putting coded data in an input buffer and an output buffer to store the decoded image frame.
- Supports streaming coded data into JPEG Decoder module. This feature allows an application to read coded JPEG image from a file or from network without buffering the entire image.
- Configures the number of image lines to decode. This feature enables the application to process the decoded image on the fly without buffering the entire frame.
- Supports the input decoded format YCbCr444, YCbCr422, YCbCr420, YCbCr411.
- Supports the output format ARGB8888, RGB565.
- Returns error when the JPEG image's size, height and width don't meet the requirements

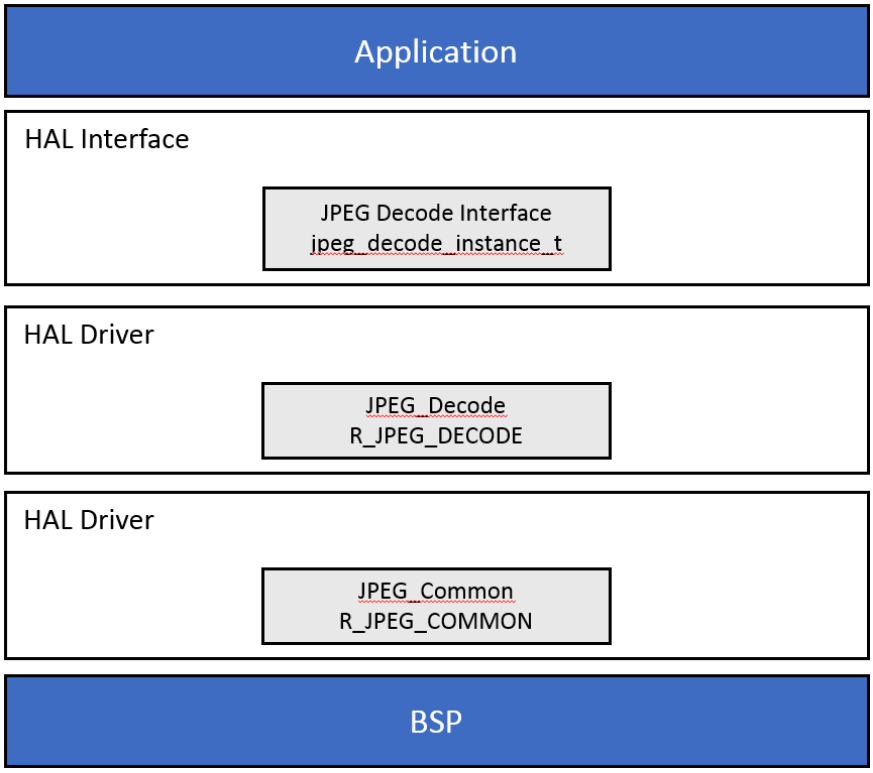


Figure 14.26 JPEG Decode

The following hardware features are, or are not, supported by SSP for JPEG.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	8 lines by 8 pixels in YCbCr444	8 lines by 16 pixels in YCbCr422	8 lines by 32 pixels in YCbCr411	16 lines by 16 pixels in YCbCr420	Output decoded format ARGB8888	Output decoded format RGB565
S124	N/A	N/A	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	N/A	N/A	N/A
S1JA	N/A	N/A	N/A	N/A	N/A	N/A
S3A1	N/A	N/A	N/A	N/A	N/A	N/A
S3A3	N/A	N/A	N/A	N/A	N/A	N/A
S3A6	N/A	N/A	N/A	N/A	N/A	N/A
S3A7	N/A	N/A	N/A	N/A	N/A	N/A
S5D5	N/A	N/A	N/A	N/A	N/A	N/A
S5D9	✓	✓	✓	✓	✓	✓
S7G2	✓	✓	✓	✓	✓	✓

14.27 JPEG Encode

The JPEG Encode HAL module provides high-level APIs for encoding images. The JPEG Encode HAL module uses the JPEG Codec peripheral on Synergy MCU. A user callback function is available to inform the application program of key processing events.

The JPEG Codec Encode Module supports the following features:

- Supports JPEG Compression.
- Supports polling mode that allows an application to wait for JPEG Encoder to complete.
- Supports interrupt mode with user-supplied callback functions.
- Configures parameters such as horizontal and vertical resolution, horizontal stride, and Quality factor.
- Supports putting raw image data in an input buffer and an output buffer to store the encoded/compressed jpeg image.
- Supports streaming raw image data into JPEG Encoder module. This feature allows an application to read coded raw image from a capture device or camera or from network without buffering the entire image.
- Only supports the YCbCr422 color space to input.
- Configurable for creating Motion JPEG (MJPEG) video compression/decompression system
- Support DRI Maker for RTP streaming application.
- Support quality factor configuration: The quality factor value determines the quality of output image.

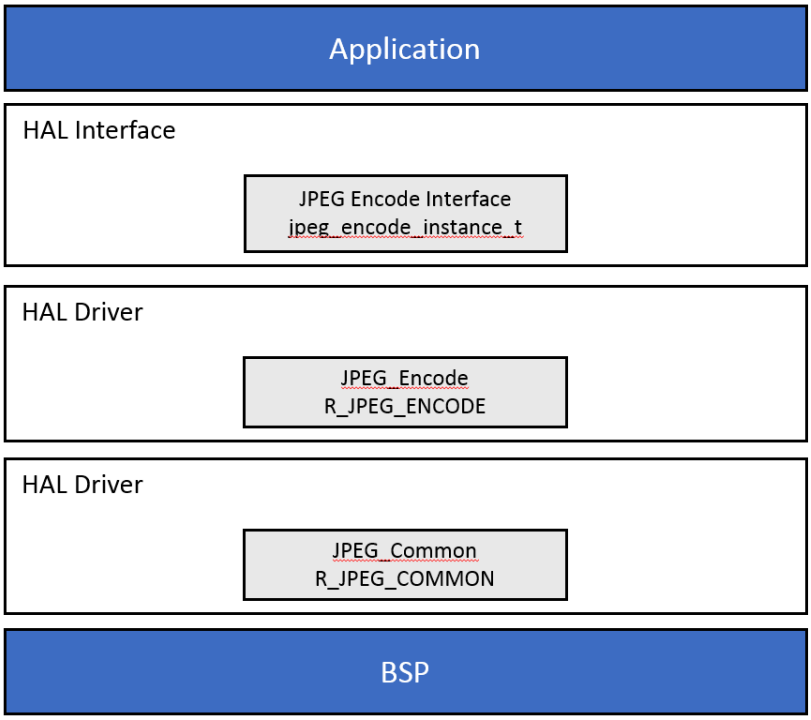


Figure 14.27 JPEG Codec Encode

The following hardware features are, or are not, supported by SSP for JPEG.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Input data format 8 lines by 8 pixels in YCbCr444	Input data format 8 lines by 16 pixels in YCbCr422	Input data format 8 lines by 32 pixels in YCbCr411	Input data format 16 lines by 16 pixels in YCbCr420	Output format jpeg
S124	N/A	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	N/A	N/A
S1JA	N/A	N/A	N/A	N/A	N/A
S3A1	N/A	N/A	N/A	N/A	N/A
S3A3	N/A	N/A	N/A	N/A	N/A
S3A6	N/A	N/A	N/A	N/A	N/A
S3A7	N/A	N/A	N/A	N/A	N/A
S5D5	N/A	N/A	N/A	N/A	N/A
S5D9	N/A	✓	N/A	N/A	✓
S7G2	N/A	✓	N/A	N/A	✓

14.28 Key Matrix Driver Interface (KINT)

The Key Matrix HAL module provides high-level APIs for Key Matrix HAL applications and is implemented on r_kint. The Key Matrix HAL module uses the key-interrupt function peripheral on the Synergy MCU. A user-defined callback can be created to inform the CPU of a key press event.

Key Matrix HAL Module Features:

This Key Matrix HAL module configures and controls the Key Interrupt (KINT) peripheral. It implements the following key functions:

- Supports both rising and falling edges on KINT channels
- Supports interrupt-based event notification
- Supports a bit-masking function to capture multiple events efficiently
- Supports a matrix keypad with edges on any two channels

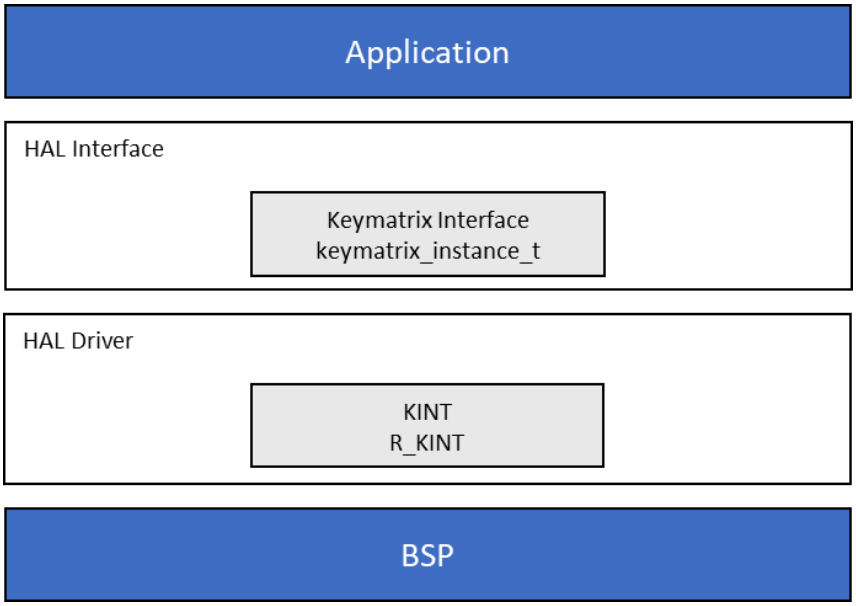


Figure 14.28 Keyboard Interrupt

The following hardware features are, or are not, supported by SSP for KINT.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Vary Input from KR00 to KR07
S124	✓
S128	✓
S1JA	✓
S3A1	✓
S3A3	✓
S3A6	✓
S3A7	✓
S5D5	✓
S5D9	✓
S7G2	✓

14.29 Low Power Mode Version 1 (LPMV1)

The Low Power Modes HAL module provides high-level APIs for low-power mode applications and is implemented on r_lpm. The Low Power Modes HAL module uses the low-power mode hardware peripheral on the Synergy MCU.

The LPM HAL module supports configuration of MCU operating power control modes and MCU low power modes using the Low Power Mode hardware peripheral.

The LPM driver supports the following operating power control modes:

- Low-voltage mode
- Low-speed mode
- Middle-speed mode
- High-speed mode
- Subosc-speed mode

The LPM driver supports the following low power modes:

- Deep Software Standby mode
- Software Standby mode
- Sleep mode
- Snooze mode

Note: LPMV1 is a deprecated module. Do not use this LPMV1 for new projects; use the LPMV2 HAL module. Not all low power modes are available on all MCUs. Not all operating modes are available on all MCUs.

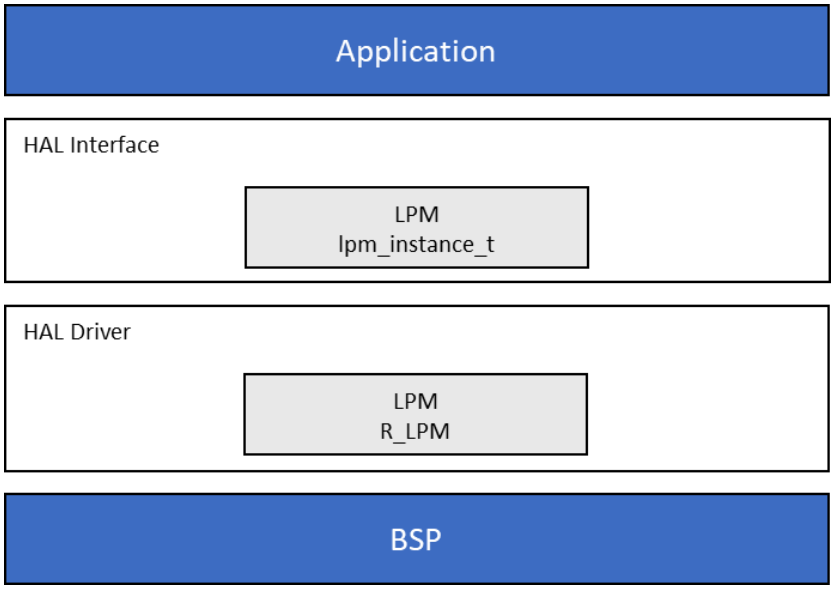


Figure 14.29 Low Power Mode Version 1 HAL Module

14.30 Low Power Modes Version 2 (LPM V2)

The Low Power Modes V2 HAL module provides high-level APIs for low-power mode applications and is implemented on r_lpm2. The Low Power Modes V2 HAL module uses the low-power mode hardware peripheral on the Synergy MCU.

The LPM V2 HAL module supports configuration of MCU operating power-control modes and MCU low-power modes using the low-power mode hardware peripheral.

The LPM Version 2 HAL Module Features

- Deep Software Standby mode
- Software Standby mode
- Sleep mode
- Snooze mode

The LPM V2 HAL module supports reducing power consumption when in deep stand-by mode through internal power-supply control and by resetting the states of I/O ports. The LPM V2 HAL module supports disabling and enabling of the MCU's other hardware peripherals.

Notes:

— Not all low-power V2 modes are available on all MCUs Groups.

— No longer will the Low Power Modes V2 HAL module handle operating power-control modes of the MCU; these are now handled by the CGC HAL module.

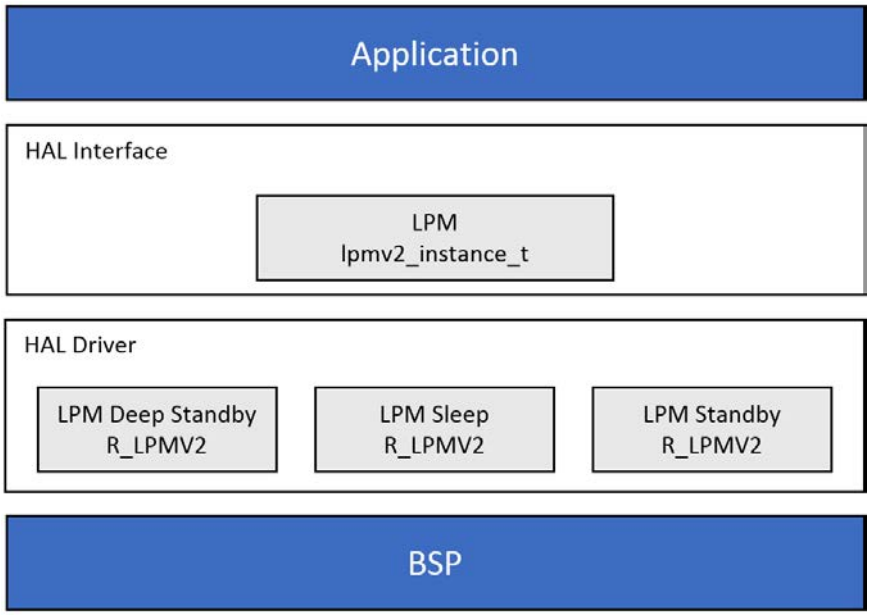


Figure 14.30 Low Power Modes Version 2 (LPM V2)

The following hardware features are, or are not, supported by SSP for LPM V2:

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Module-stop/start bits access	Low Voltage Operating Power Control Mode	Subosc Speed Operating Power Control Mode
S124	✓	☒	✓
S128	✓	☒	✓
S1JA	✓	☒	✓

MCU Group	Module-stop/start bits access	Low Voltage Operating Power Control Mode	Subosc Speed Operating Power Control Mode
S3A1	✓	☒	✓
S3A3	✓	☒	✓
S3A6	✓	☒	✓
S3A7	✓	☒	✓
S5D5	✓	N/A	✓
S5D9	✓	N/A	✓
S7G2	✓	N/A	✓

MCU Group	Sleep Low Power Mode	Software Standby Low Power Mode	Deep Standby Low Power Mode	Snooze enabled in Software Standby Low Power Mode	Snooze Linking using ELC	DTC state in Snooze Mode
S124	✓	✓	N/A	✓	☒	✓
S128	✓	✓	N/A	✓	☒	✓
S1JA	✓	✓	N/A	✓	☒	✓
S3A1	✓	✓	N/A	✓	☒	✓
S3A6	✓	✓	N/A	✓	☒	✓
S3A3	✓	✓	N/A	✓	☒	✓
S3A7	✓	✓	N/A	✓	☒	✓
S5D5	✓	✓	✓	✓	☒	✓
S5D9	✓	✓	✓	✓	☒	✓
S7G2	✓	✓	✓	✓	☒	✓

MCU Group	State of address bus and bus signals in Standby or Deep Standby Mode	Enter Snooze mode via RXD0 (SCI0)	IO Port state control after wake from Deep Standby Mode	Internal Power Supply control in Deep Standby Mode (power supply to LOCO, Standby SRAM, AGTn, and USBHS/FS)
S124	☒	✓	✓	☒
S128	☒	✓	✓	☒
S1JA	☒	✓	✓	☒
S3A1	✓	✓	✓	☒
S3A3	✓	✓	✓	☒
S3A6	✓	✓	✓	☒
S3A7	✓	✓	✓	☒
S5D5	✓	✓	✓	✓
S5D9	✓	✓	✓	✓
S7G2	✓	✓	✓	✓

14.31 Low Voltage Detection (LVD)

The Low Voltage Detection (LVD) HAL module provides high-level APIs for voltage-detection applications and is implemented on r_lvd. The LVD HAL module uses the LVD peripheral on the Synergy MCU. A user-defined callback can be created to notify the CPU when a voltage-detection event is triggered. The VCC is the source for all voltage-detection functions.

LVD HAL Module Features

- V_{CC} as the voltage-detection input
- One build-time configurable low-voltage detector (via OFS1 register)
- Two run-time configurable low-voltage detectors
- Two result flags; one for a threshold check and one for the current state
- Support for both interrupt or polling-event checking

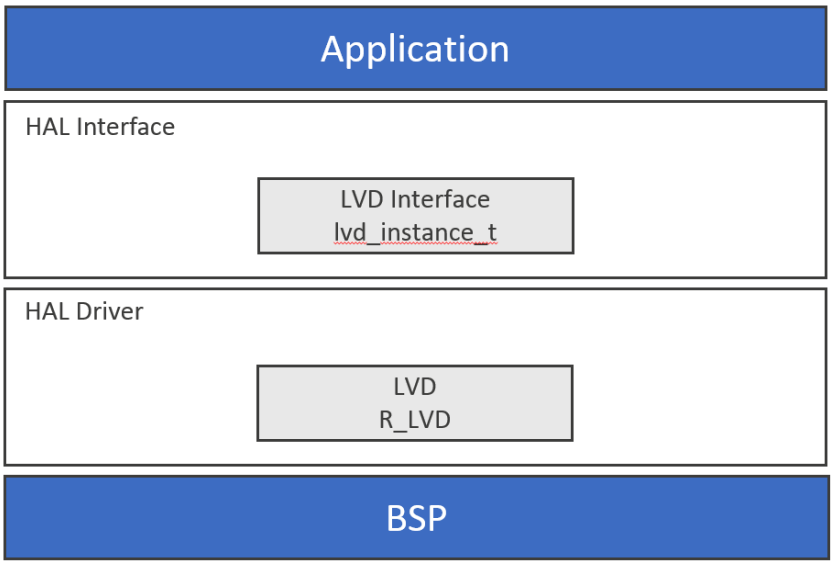


Figure 14.31 Low Voltage Detection (LVD)

The following hardware features are, or are not, supported by SSP for LVD:

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	VCC Rising voltage generates an interrupt or non-maskable interrupt	VCC Falling voltage generates an interrupt or non-maskable interrupt	VCC Rising and falling voltage generates an interrupt or non-maskable interrupt	Callback notification for maskable and non-maskable interrupt	Reset on falling voltage	Monitoring LVD 1 and 2 status flags by polling
S124	✓	✓	✓	✓	✓	✓
S128	✓	✓	✓	✓	✓	✓
S1JA	✓	✓	✓	✓	✓	✓
S3A1	✓	✓	✓	✓	✓	✓
S3A3	✓	✓	✓	✓	✓	✓
S3A6	✓	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓	✓

MCU Group	VCC Rising voltage generates an interrupt or non-maskable interrupt	VCC Falling voltage generates an interrupt or non-maskable interrupt	VCC Rising and falling voltage generates an interrupt or non-maskable interrupt	Callback notification for maskable and non-maskable interrupt	Reset on falling voltage	Monitoring LVD 1 and 2 status flags by polling
S5D5	✓	✓	✓	✓	✓	✓
S5D9	✓	✓	✓	✓	✓	✓
S7G2	✓	✓	✓	✓	✓	✓

MCU Group	Digital Filtering with adjustable filter time	Event link function through ELC HAL driver
S124	N/A	☒
S128	N/A	☒
S1JA	N/A	☒
S3A1	N/A	☒
S3A3	N/A	☒
S3A6	N/A	☒
S3A7	N/A	☒
S5D5	✓	☒
S5D9	✓	☒
S7G2	✓	☒

14.32 Operational Amplifier (OPAMP)

The OPAMP HAL module implements the OPAMP API for signal amplification applications on r_opamp. It supports the OPAMP peripheral available on Synergy MCUs.

OPAMP HAL Module Features

- Low power or high-speed mode
- Start by software or AGT compare match
- Stop by software or ADC conversion end (stop by ADC conversion end only supported on op-amp channels configured to start by AGT compare match)
- Trimming available on some MCUs (see hardware manual)

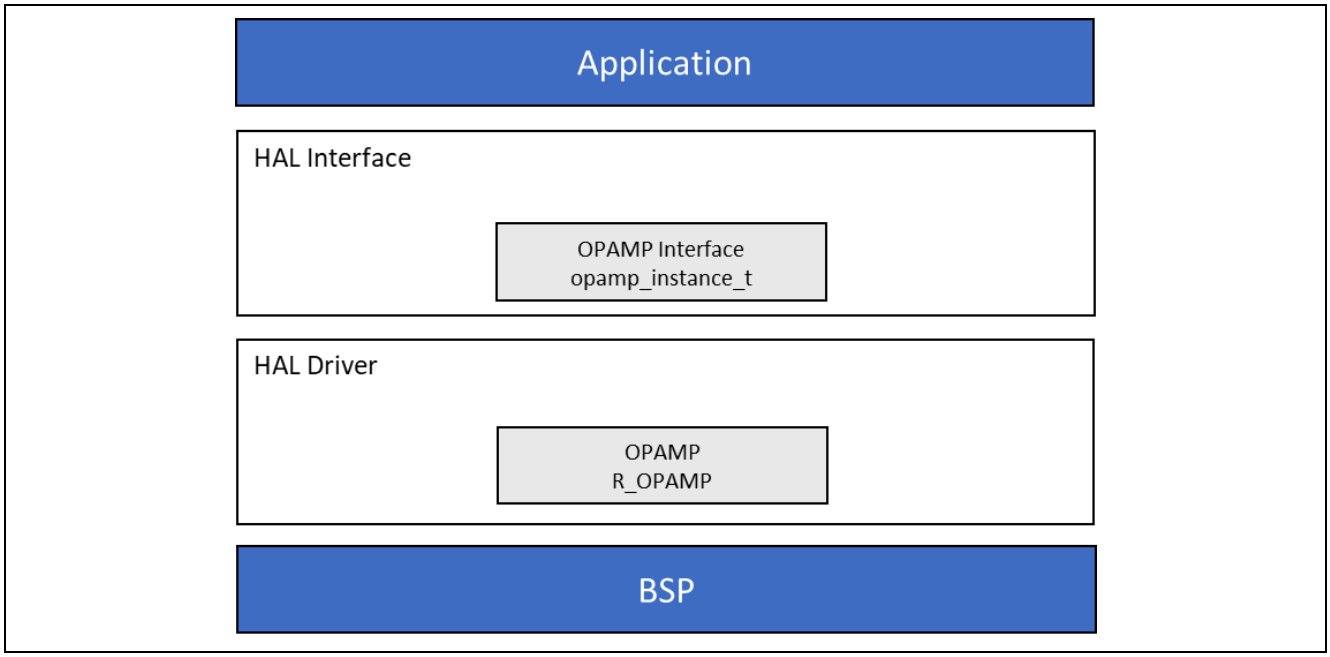


Figure 14.32 OPAMP HAL Module Block Diagram

The following hardware features are, or are not, supported by SSP for OPAMP:

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Low power mode	High speed mode	Start by SW	AGT compare match	Stop by SW	ADC conversion end
S124	N/A	N/A	N/A	N/A	N/A	N/A
S128	☒	☒	☒	☒	☒	☒
S1JA	✓	✓	✓	✓	✓	✓
S3A1	☒	☒	☒	☒	☒	☒
S3A3	☒	☒	☒	☒	☒	☒
S3A6	☒	☒	☒	☒	☒	☒
S3A7	☒	☒	☒	☒	☒	☒
S5D5	N/A	N/A	N/A	N/A	N/A	N/A
S5D9	N/A	N/A	N/A	N/A	N/A	N/A
S7G2	N/A	N/A	N/A	N/A	N/A	N/A

14.33 Parallel Data Capture Unit (PDC)

The Parallel Data Capture Unit (PDC) HAL module provides high-level APIs to capture images from a camera application and is implemented on r_pdc. The PDC HAL module uses the PDC peripherals on the Synergy MCU. A user-defined callback can be created to inform the CPU when a capture has been completed.

PDC HAL Module Features

- Supports capture from a connected and configured camera.
- Supports a callback that informs the CPU when a capture is complete
- Provides a pointer to the capture buffer
- Provides an indication of the event triggering the callback

The PDC driver implements the PDC interface in the SSP:

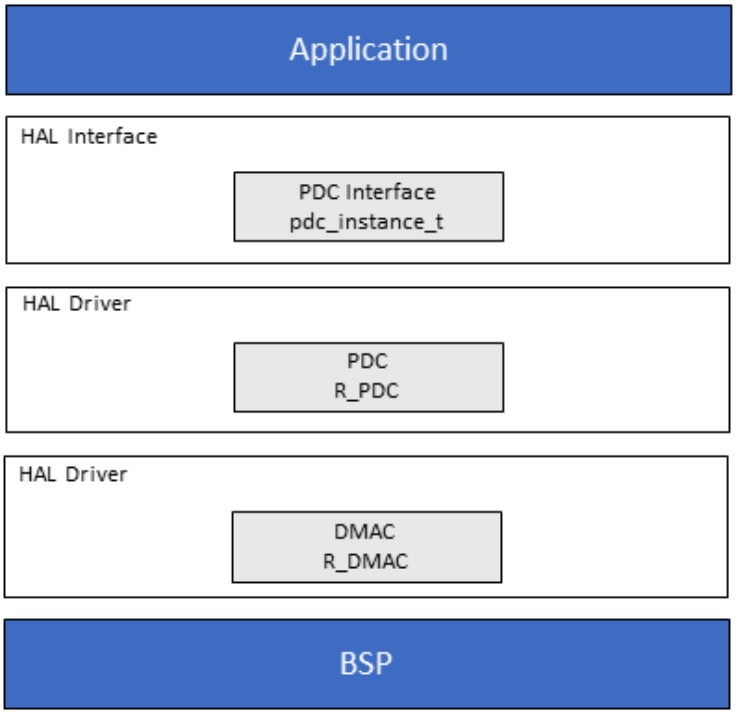


Figure 14.33 Parallel Data Capture Unit

The following hardware features are, or are not, supported by SSP for PDC:

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Supports up to 4095 lines vertical	Supports 4 to 4095 bytes horizontal	Accepts interrupts from Receive Data Ready	Accepts interrupts from Frame End	Accepts interrupts from Overrun	Accepts interrupts from Under run
S124	N/A	N/A	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	N/A	N/A	N/A
S1JA	N/A	N/A	N/A	N/A	N/A	N/A
S3A1	N/A	N/A	N/A	N/A	N/A	N/A
S3A3	N/A	N/A	N/A	N/A	N/A	N/A
S3A6	N/A	N/A	N/A	N/A	N/A	N/A
S3A7	N/A	N/A	N/A	N/A	N/A	N/A
S5D5	✓	✓	✓	✓	✓	✓
S5D9	✓	✓	✓	✓	✓	✓
S7G2	✓	✓	✓	✓	✓	✓

MCU Group	Accepts interrupts from Error in wrong number of lines	Accepts interrupts from Error in wrong number of bytes per line	Frame end and receive data ready interrupts can start DTC	Frame end and receive data ready interrupts can start DMAC	Frequency division ratio: Selectable from 2, 4, 6, 8, 10, 12, 14, and 16	Supports PDC Reset Function
S124	N/A	N/A	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	N/A	N/A	N/A
S1JA	N/A	N/A	N/A	N/A	N/A	N/A

MCU Group	Accepts interrupts from Error in wrong number of lines	Accepts interrupts from Error in wrong number of bytes per line	Frame end and receive data ready interrupts can start DTC	Frame end and receive data ready interrupts can start DMAC	Frequency division ratio: Selectable from 2, 4, 6, 8, 10, 12, 14, and 16	Supports PDC Reset Function
S3A1	N/A	N/A	N/A	N/A	N/A	N/A
S3A3	N/A	N/A	N/A	N/A	N/A	N/A
S3A6	N/A	N/A	N/A	N/A	N/A	N/A
S3A7	N/A	N/A	N/A	N/A	N/A	N/A
S5D5	✓	✓	☒	☒	☒	☒
S5D9	✓	✓	☒	☒	☒	☒
S7G2	✓	✓	☒	☒	☒	☒

MCU Group	Supports Selectable active polarity for VSYNC and HSYNC signals	Supports Monitoring of VSYNC and HSYNC signals	Endian order selectable
S124	N/A	N/A	N/A
S128	N/A	N/A	N/A
S1JA	N/A	N/A	N/A
S3A1	N/A	N/A	N/A
S3A3	N/A	N/A	N/A
S3A6	N/A	N/A	N/A
S3A7	N/A	N/A	N/A
S5D5	✓	✓	✓
S5D9	✓	✓	✓
S7G2	✓	✓	✓

14.34 Quad SPI (QSPI)

The Quad SPI (QSPI) HAL module provides high-level APIs for erasing and programming the contents of a QSPI flash device connected to the microcontroller and is implemented on the r_qspi peripheral on the Synergy MCU. Unlike many other modules, there is no callback function for the QSPI.

QSPI HAL Module Features:

- Initialize the QSPI peripheral
- Access Quad SPI flash devices using Direct Communication Mode
- Read data from a QSPI flash device
- Program the page of a QSPI flash device
- Erase sectors of a QSPI flash device
- Select a bank to control access to a QSPI flash device

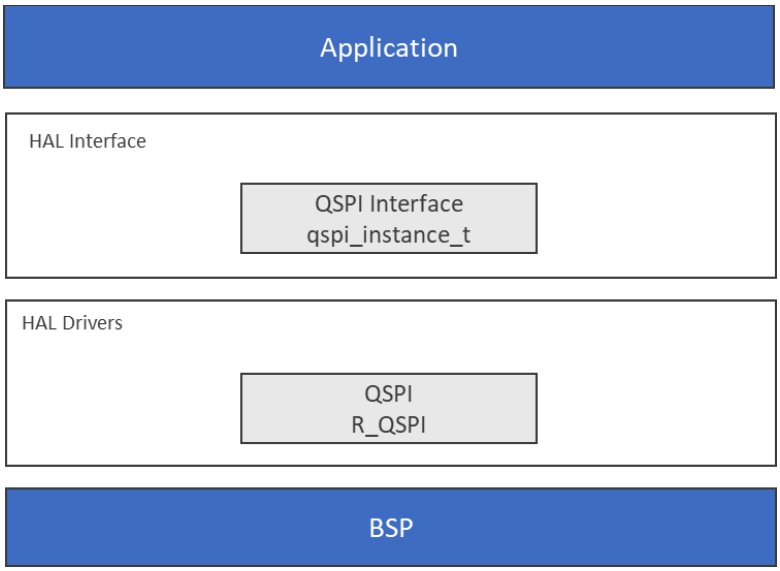


Figure 14.34 Quad SPI

The following hardware features are, or are not, supported by SSP for the QSPI.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Extended SPI	Dual SPI	Quad SPI	SPI Mode 1 & 3	Selectable Address (8, 16, 24, 32 bits) via SFMSAC register	Timing adjustment function
S124	N/A	N/A	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	N/A	N/A	N/A
S1JA	N/A	N/A	N/A	N/A	N/A	N/A
S3A1	✓	✓	N/A	☒	3 byte and 4 byte addresses	☒
S3A3	✓	✓	N/A	☒	3 byte and 4 byte addresses	☒
S3A6	✓	✓	N/A	☒	3 byte and 4 byte addresses	☒
S3A7	✓	✓	✓	☒	3 byte and 4 byte addresses	☒
S5D5	✓	✓	✓	☒	3 byte and 4 byte addresses	☒
S5D9	✓	✓	✓	☒	3 byte and 4 byte addresses	☒
S7G2	✓	✓	✓	☒	3 byte and 4 byte addresses	☒

MCU Group	Flash read function: Read	Flash read function: Fast Read	Flash read function: Fast Read/Dual Output	Flash read function: Fast Read/Dual I/O	Flash read function: Fast Read/Quad Output	Flash read function: Fast Read/Quad I/O
S124	N/A	N/A	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	N/A	N/A	N/A
S1JA	N/A	N/A	N/A	N/A	N/A	N/A
S3A1	✓	☒	☒	☒	☒	☒
S3A3	✓	☒	☒	☒	☒	☒
S3A6	✓	☒	☒	☒	☒	☒
S3A7	✓	☒	☒	☒	☒	☒
S5D5	✓	☒	☒	☒	☒	☒
S5D9	✓	☒	☒	☒	☒	☒
S7G2	✓	☒	☒	☒	☒	☒

MCU Group	Substitutable Instruction Code	Adjustment # of Dummy cycles	Prefetch Function	Polling Processing *	SPI bus cycle extension Function	Direct communication function
S124	N/A	N/A	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	N/A	N/A	N/A
S1JA	N/A	N/A	N/A	N/A	N/A	N/A
S3A1	N/A	☒	☒	✓	N/A	N/A
S3A3	N/A	☒	☒	✓	N/A	N/A
S3A6	N/A	☒	☒	✓	N/A	N/A
S3A7	N/A	☒	☒	✓	N/A	N/A
S5D5	N/A	☒	☒	✓	N/A	N/A
S5D9	N/A	☒	☒	✓	N/A	N/A
S7G2	N/A	☒	☒	✓	N/A	N/A

MCU Group	Interrupt source	Module-stop function	XIP
S124	N/A	N/A	N/A
S128	N/A	N/A	N/A
S1JA	N/A	N/A	N/A
S3A1	N/A	N/A	✓
S3A3	N/A	N/A	✓
S3A6	N/A	N/A	✓
S3A7	N/A	N/A	✓
S5D5	N/A	N/A	✓
S5D9	N/A	N/A	✓
S7G2	N/A	N/A	✓

14.35 Realtime Clock (RTC)

The Real-Time Clock (RTC) HAL module provides high-level APIs for RTC applications and is implemented on r_rtc. The RTC HAL module configures the RTC module and controls clock, calendar, and alarm functions. The RTC uses the real-time clock module on the Synergy MCU. A user-defined callback can be created to respond to any of the three supported interrupt types: alarm, periodic, or carry.

The RTC HAL module supports the following functions of the real-time clock:

- RTC peripheral configuration
- RTC time and date get and set
- RTC time and date alarm get and set
- RTC time counter start and stop
- RTC alarm, periodic, and carry event notification
- RTC event type enable and disable
- RTC event rate configuration
- RTC clock source get and set

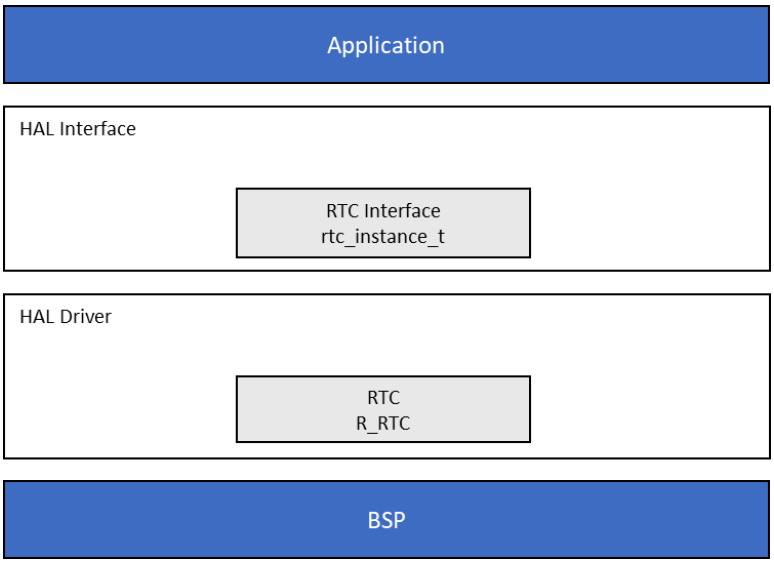


Figure 14.35 Realtime Clock

The following hardware features are, or are not, supported by SSP for the RTC.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Calendar Mode	Binary Mode	Sub-clock (XCIN) Count Source	LOCO Count Source	12 hours/24 hours	Alarm interrupt (RTC_ALM)
S124	✓	☒	✓	✓	24 Hours	✓
S128	✓	☒	✓	✓	24 Hours	✓
S1JA	✓	☒	✓	✓	24 Hours	✓
S3A1	✓	☒	✓	✓	24 Hours	✓
S3A3	✓	☒	✓	✓	24 Hours	✓
S3A6	✓	☒	✓	✓	24 Hours	✓
S3A7	✓	☒	✓	✓	24 Hours	✓
S5D5	✓	☒	✓	✓	24 Hours	✓
S5D9	✓	☒	✓	✓	24 Hours	✓
S7G2	✓	☒	✓	✓	24 Hours	✓

MCU Group	Periodic interrupt (RTC_PRD)	Carry interrupt (RTC_CUP)	Time capture function	Event link function through ELC HAL driver	Start/stop function	Clock error correction function
S124	✓	✓	☒	☒	✓	✓
S128	✓	✓	☒	☒	✓	✓
S1JA	✓	✓	☒	☒	✓	✓
S3A1	✓	✓	☒	☒	✓	✓
S3A3	✓	✓	☒	☒	✓	✓
S3A6	✓	✓	☒	☒	✓	✓
S3A7	✓	✓	☒	☒	✓	✓
S5D5	✓	✓	☒	☒	✓	✓
S5D9	✓	✓	☒	☒	✓	✓
S7G2	✓	✓	☒	☒	✓	✓

14.36 RIIC (I^2C Full Featured)

SSP provides two RIIC (Renesas I^2C) drivers which implements two separate HAL interfaces for I^2C peripheral:

1. RIIC Master HAL module implements the I^2C master interface which configures and operates I^2C peripheral in master mode.
2. RIIC Slave HAL module implements I^2C slave interface which configures and operates I^2C peripheral in slave mode.

These two HAL implementations are mutually exclusive in that an I^2C peripheral can only operate either in master or slave mode (that is, in a multi-master scenario, the I^2C peripheral cannot dynamically switches between master and slave configurations).

Both RIIC master and slave drivers have the following capabilities:

- Interrupt driven transmit/receive processing.
- Callback function support which can return an event code.
- Supports I^2C Normal-mode with bit rates up to 100 kbps
- Supports I^2C Fast-mode with bit rates of up to 400 kbps
- Supports Fast-mode plus with 1 Mbps bit rates (On select channels of S7G2 and S5D9).

The callback functions will be called with the following events:

- Transfer aborted
- Transmit complete (number of bytes transmitted provided)
- Receive complete (number of bytes received provided)

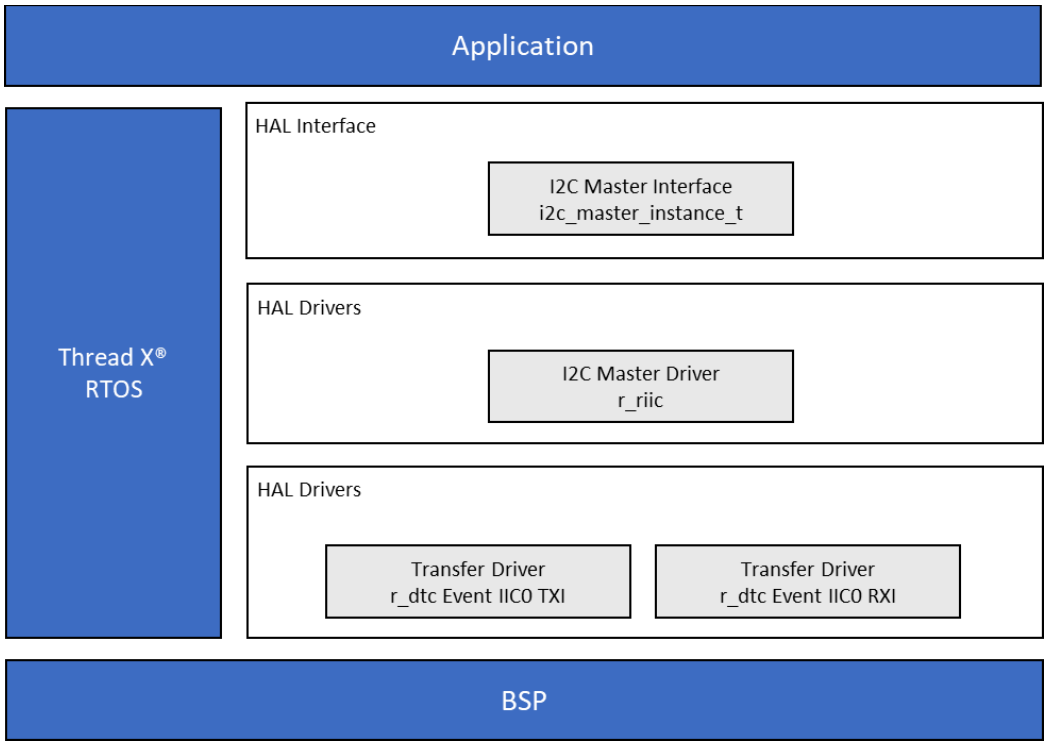


Figure 14.36 I²C (RIIC) Master

RIIC master driver has the following additional capabilities:

- Read from an I²C slave device with 7 or 10 bit address.
- Write to an I²C slave device with 7 or 10 bit address.
- Reset the I²C peripheral from the user code.
- Set the communication slave address dynamically prior to an I²C transaction.

RIIC slave driver has the following additional capabilities:

- Support for I²C Slave operations
- Support transactions with a I²C master device
  - Read
  - Write
- Callback support
  - Transmit complete (number of bytes transmitted provided)
  - Receive complete (number of bytes received provided)

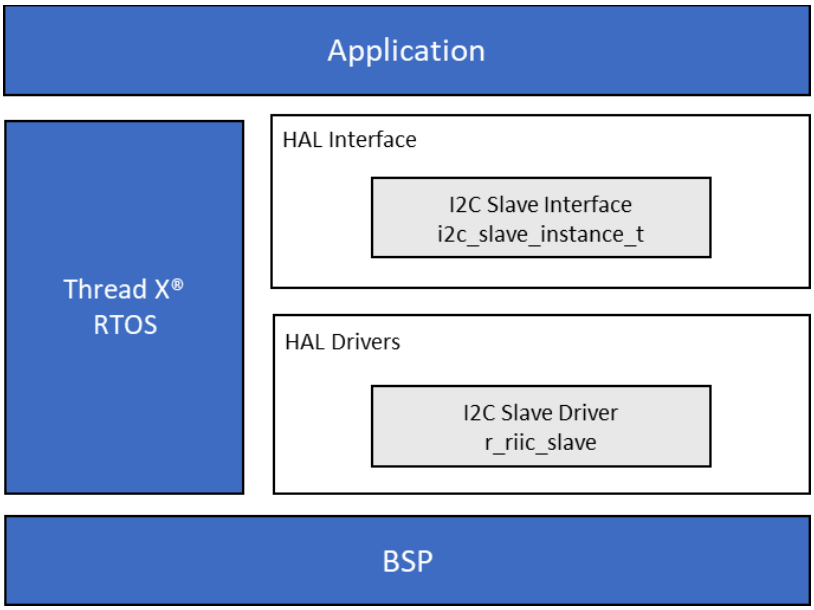


Figure 14.37 I²C (RIIC) Slave

The following hardware features are, or are not, supported by SSP for the RIIC Master Driver:

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	I²C Format	SMBus Format	Master Mode	Slave Mode	Fast Mode Plus	Selectable duty cycle
S124	✓	☒	✓	✓	N/A	☒
S128	✓	☒	✓	✓	N/A	☒
S1JA	✓	☒	✓	✓	N/A	☒
S3A1	✓	☒	✓	✓	N/A	☒
S3A3	✓	☒	✓	✓	N/A	☒
S3A6	✓	☒	✓	✓	N/A	☒
S3A7	✓	☒	✓	✓	N/A	☒
S5D5	✓	☒	✓	✓	✓	☒
S5D9	✓	☒	✓	✓	✓	☒
S7G2	✓	☒	✓	✓	✓	☒

MCU Group	Configurable to up to three different slave addresses	7- and 10-bit address formats	General call address, Device ID address and SMBus host address detectable	Automatic loading of the ack-nowledge bit	SDA output delay function	Selectable Wait functions (8/9 or 9/1)
S124	☒	7 and 10 bit	☒	✓	☒	☒
S128	☒	7 and 10 bit	☒	✓	☒	☒
S1JA	☒	7 and 10 bit	☒	✓	☒	☒
S3A1	☒	7 and 10 bit	☒	✓	☒	☒

MCU Group	Configurable to up to three different slave addresses	7- and 10-bit address formats	General call address, Device ID address and SMBus host address detectable	Automatic loading of the ack-nowledge bit	SDA output delay function	Selectable Wait functions (8/9 or 9/1)
S3A3	☒	7 and 10 bit	☒	✓	☒	☒
S3A6	☒	7 and 10 bit	☒	✓	☒	☒
S3A7	☒	7 and 10 bit	☒	✓	☒	☒
S5D5	☒	7 and 10 bit	☒	✓	☒	☒
S5D9	☒	7 and 10 bit	☒	✓	☒	☒
S7G2	☒	7 and 10 bit	☒	✓	☒	☒

MCU Group	Full Arbitration Support	Internal Detect Time-Out	Programmable Digital Noise Filters	Support all Interrupt Sources
S124	Master, NACK arbitrations	✓	☒	✓
S128	Master, NACK arbitrations	✓	☒	✓
S1JA	Master, NACK arbitrations	✓	☒	✓
S3A1	Master, NACK arbitrations	✓	☒	✓
S3A3	Master, NACK arbitrations	✓	☒	✓
S3A6	Master, NACK arbitrations	✓	☒	✓
S3A7	Master, NACK arbitrations	✓	☒	✓
S5D5	Master, NACK arbitrations	✓	☒	✓
S5D9	Master, NACK arbitrations	✓	☒	✓
S7G2	Master, NACK arbitrations	✓	☒	✓

RIIC Slave driver supported features:

MCU Group	I^2C Format	SMBus Format	Master Mode	Slave Mode	Fast Mode Plus	Selectable duty cycle
S124	✓	☒	☒	✓	N/A	☒
S128	✓	☒	☒	✓	N/A	☒
S1JA	✓	☒	☒	✓	N/A	☒
S3A1	✓	☒	☒	✓	N/A	☒
S3A3	✓	☒	☒	✓	N/A	☒
S3A6	✓	☒	☒	✓	N/A	☒
S3A7	✓	☒	☒	✓	N/A	☒
S5D5	✓	☒	☒	✓	✓	☒
S5D9	✓	☒	☒	✓	✓	☒
S7G2	✓	☒	☒	✓	✓	☒

MCU Group	Configurable to up to three different slave addresses	7- and 10-bit address formats	General call address, Device ID address and SMBus host address detectable	Automatic loading of the acknowledge bit	SDA output delay function	Selectable Wait functions (8/9 or 9/1)
S124	One slave address	7 and 10 bit	☒	✓	☒	☒
S128	One slave address	7 and 10 bit	☒	✓	☒	☒
S1JA	One slave address	7 and 10 bit	☒	✓	☒	☒
S3A1	One slave address	7 and 10 bit	☒	✓	☒	☒
S3A3	One slave address	7 and 10 bit	☒	✓	☒	☒
S3A6	One slave address	7 and 10 bit	☒	✓	☒	☒
S3A7	One slave address	7 and 10 bit	☒	✓	☒	☒
S5D5	One slave address	7 and 10 bit	☒	✓	☒	☒
S5D9	One slave address	7 and 10 bit	☒	✓	☒	☒
S7G2	One slave address	7 and 10 bit	☒	✓	☒	☒

MCU Group	Full Arbitration Support	Internal Detect Time-Out	Programmable Digital Noise Filters	Support all Interrupt Sources	Event link function through ELC HAL driver
S124	Slave, NACK arbitration	✓	☒	TEI not supported	☒
S128	Slave, NACK arbitration	✓	☒	TEI not supported	☒
S1JA	Slave, NACK arbitration	✓	☒	TEI not supported	☒
S3A1	Slave, NACK arbitration	✓	☒	TEI not supported	☒
S3A3	Slave and NACK arbitration	✓	☒	TEI not supported	☒
S3A6	Slave and NACK arbitration	✓	☒	TEI not supported	☒
S3A7	Slave, NACK arbitration	✓	☒	TEI not supported	☒
S5D5	Slave, NACK arbitration	✓	☒	TEI not supported	☒
S5D9	Slave, NACK arbitration	✓	☒	TEI not supported	☒
S7G2	Slave and NACK arbitration	✓	☒	TEI not supported	☒

14.37 SD Multi Media Card (SDMMC)

The SDMMC (SD/MMC and SDIO) module is implemented on r_sdmmc and is used to read/write and control SD/MMC media devices as well as SDIO cards. The SDMMC module can be used as a standalone SD card, or eMMC, media driver or it can be used with FileX™, or any other compatible file system. The r_sdmmc module uses the SD/MMC peripheral on the Synergy MCU.

SDMMC HAL Module Features:

- Supports the following memory devices: SDSC (SD Standard Capacity), SDHC (SD High Capacity), SDXC (SD Extended Capacity), and eMMC (embedded Multi Media Card)
  - Supports reading, writing, and erasing SD and eMMC memory devices
  - Supports 1, 4, or 8 bit data bus (8-bit bus is supported for eMMC only)
  - Supports detection of hardware write protection (SD cards only)
  - Automatically selects between backwards compatible mode and High-Speed Single Data Rate (SDR) mode (eMMC)
- Supports SDIO
  - Supports SDIO single register access (CMD52)
  - Supports SDIO multiple register access (CMD53)
  - Supports SDIO interrupts
- Automatically configures the clock to the maximum clock rate supported by both host (MCU) and device

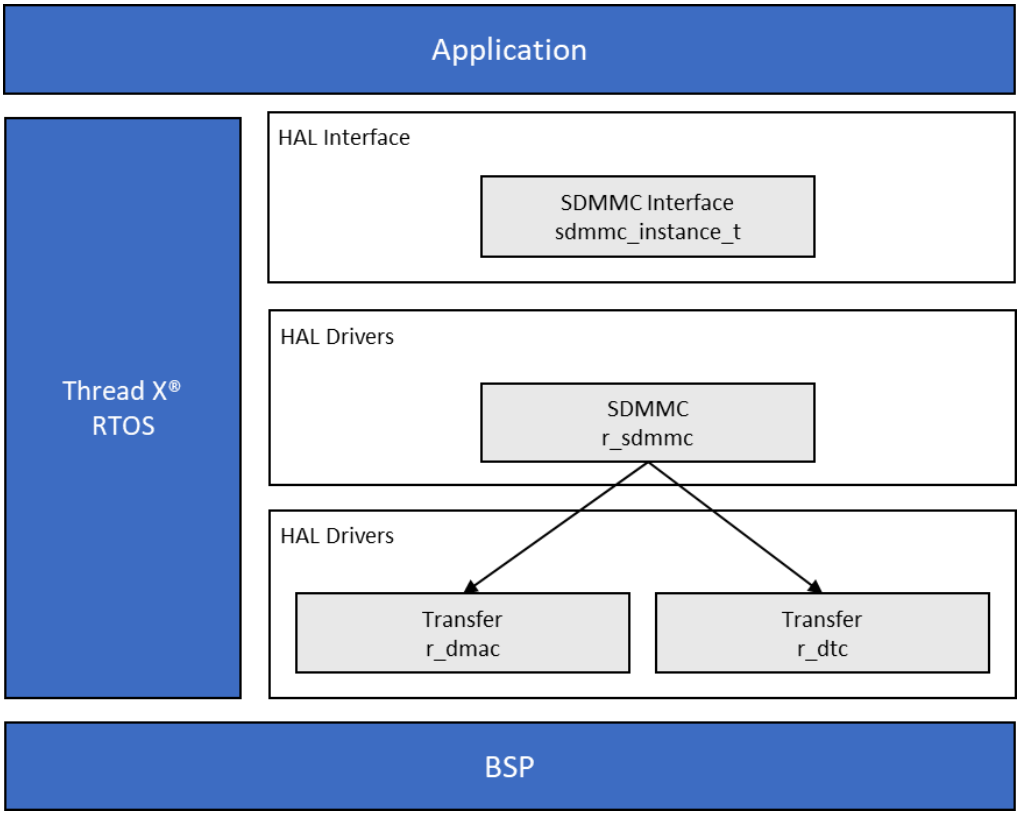


Figure 14.38 SDMMC HAL Driver interface

The following hardware features are, or are not, supported by SSP for SD/MMC (SDIO):

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

	SD 1 bit	SD 4 bit	SDHC	SDXC	MMC 1 bit	MMC 4 bit
S124	N/A	N/A	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	N/A	N/A	N/A
S1JA	N/A	N/A	N/A	N/A	N/A	N/A
S3A1	✓	✓	✓	✓	☒	☒
S3A3	N/A	N/A	N/A	N/A	N/A	N/A
S3A6	N/A	N/A	N/A	N/A	N/A	N/A
S3A7	✓	✓	✓	✓	☒	☒
S5D5	✓	✓	✓	✓	☒	☒
S5D9	✓	✓	✓	✓	☒	☒
S7G2	✓	✓	✓	✓	✓	✓

MCU Group	MMC 8 bit	MMC Backward Compatibility Mode	Card Detect: Insertion	Card Detect: Removal	Write Protect	MMC High Speed SDR Mode
S124	N/A	N/A	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	N/A	N/A	N/A
S1JA	N/A	N/A	N/A	N/A	N/A	N/A
S3A1	☒	☒	N/A	N/A	✓	☒
S3A3	N/A	N/A	N/A	N/A	N/A	N/A
S3A6	N/A	N/A	N/A	N/A	N/A	N/A
S3A7	☒	☒	N/A	N/A	✓	☒
S5D5	☒	☒	See note	See note	✓	☒
S5D9	☒	☒	See note	See note	✓	☒
S7G2	☒	☒	See note	See note	✓	✓

Note: Available on some of the MCUs (for the group indicated in the table) based on the pin map.

MCU Group	DMA Read	DMA Write	CMD w/o Response and Data	CMD w/o Data	Single Block Read	Single Block Write
S124	N/A	N/A	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	N/A	N/A	N/A
S1JA	N/A	N/A	N/A	N/A	N/A	N/A
S3A1	✓	✓	☒	☒	✓	✓
S3A3	N/A	N/A	N/A	N/A	N/A	N/A
S3A6	N/A	N/A	N/A	N/A	N/A	N/A
S3A7	✓	✓	☒	☒	✓	✓
S5D5	✓	✓	☒	☒	✓	✓
S5D9	✓	✓	☒	☒	✓	✓
S7G2	✓	✓	☒	☒	✓	✓

MCU Group	Multiple Block Read Internal Timer	Multiple Block Write External Timer	Multiple Block Read Internal Timer	Multiple Block Write External Timer	IO RW Direct	IO RW Extended
S124	N/A	N/A	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	N/A	N/A	N/A
S1JA	N/A	N/A	N/A	N/A	N/A	N/A
S3A1	✓	☒	✓	☒	✓	✓
S3A3	N/A	N/A	N/A	N/A	N/A	N/A
S3A6	N/A	N/A	N/A	N/A	N/A	N/A
S3A7	✓	☒	✓	☒	✓	✓
S5D5	✓	☒	✓	☒	✓	✓
S5D9	✓	☒	✓	☒	✓	✓
S7G2	✓	☒	✓	☒	✓	✓

MCU Group	Suspend Resume	SPI Bus	Stream Transfer MMC	HPI for MMC	Boot / Alt Boot MMC	Open Ended Multiple Block Transfer MMC
S124	N/A	N/A	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	N/A	N/A	N/A
S1JA	N/A	N/A	N/A	N/A	N/A	N/A
S3A1	N/A	N/A	N/A	N/A	N/A	N/A
S3A3	N/A	N/A	N/A	N/A	N/A	N/A
S3A6	N/A	N/A	N/A	N/A	N/A	N/A
S3A7	N/A	N/A	N/A	N/A	N/A	N/A
S5D5	N/A	N/A	N/A	N/A	N/A	N/A
S5D9	N/A	N/A	N/A	N/A	N/A	N/A
S7G2	N/A	N/A	N/A	N/A	N/A	N/A

14.38 Secure Cryptographic Engine (SCE)

The Secure Cryptographic Engine (SCE) HAL module provides high-level APIs for random number generation, digest computation (hash), data encryption and decryption, digital signing and verification, key generation (using RSA, AES and ECC algorithms) and key installation (for RSA, AES and ECC keys). The SCE is a dedicated hardware block and the functionality provided by the SCE varies across the supported MCUs

The SCE HAL module configures the cryptographic module, to enable users to build cryptographic protocols for security with the following cryptographic primitives:

- Random-number generation
- Data encryption and decryption using AES or Triple DES (3DES), ARC4 algorithms
- Signature generation and verification using the ECC, RSA, or DSA algorithms
- Message-digest computation using HASH algorithms MD5, SHA1, SHA224, or SHA256
- Key generation – AES wrapped keys, RSA plain text and wrapped keys, and ECC plain text and wrapped keys
- Installing the encrypted user key on to the Synergy Platform

The SCE interface provides a common API for SCE HAL module. The SCE interface supports multiple operations depending on the chosen module (AES, ARC4, RSA, DSA, HASH, TDES or TRNG).

The SSP Cryptographic library provides high-level, C-callable APIs for the following security functions in SCE7:

- True RNG (TRNG)
- Generates cryptographically secure 128-bit random numbers
- Generates seeds to other, deterministic random number generators (such as the NIST SP800-90A DRBG)
- Cryptographic HASH functions
- Generates hash values that provide a digital fingerprint of data
- Supports SHA1 and SHA224/SHA256

Encryption mechanism used for symmetric-key cryptography:

- Encryption/decryption key is secretly shared between transmitter and receiver.
- Advanced Encryption Standard (AES)
- Supports 128-bit, 192-bit, and 256-bit key lengths
- Supports various chaining modes: ECB, CBC, CTR, GCM, and XTS
- Data Encryption Standard 3DES
- Supports 192-bit key length, operates on a fixed 8-byte block of data
- Supports ECB and CBC chaining modes
- Used in legacy secure socket layer (SSL) and transport layer security (TLS) protocols
- 3DES applies DES three times to each block
- Alleged RC4 (ARC4)
- Supports 2048-bit key length
- Used in TLS and wired equivalent privacy (WEP)
- Throughput for 128-bit data

Encryption mechanism used for asymmetric-key cryptography:

- Public keys are exchanged between the transmitter and receiver, then the public and private keys are used to compute the shared secret between transmitter and receiver.
- Rivest, Shamir, and Adleman (RSA) supports up to 2048-bit key length
- Eliptical Curve Cryptography (ECC) supports P-192 and P-256
- Used for public-key cryptography
- Generates two keys: public and private
- Transmitter encrypts using the public key
- Receiver decrypts using the private key
- Used in digital verification for authentication, signature generation and verification, encryption/decryption for key exchange and wrapping, and other security functions

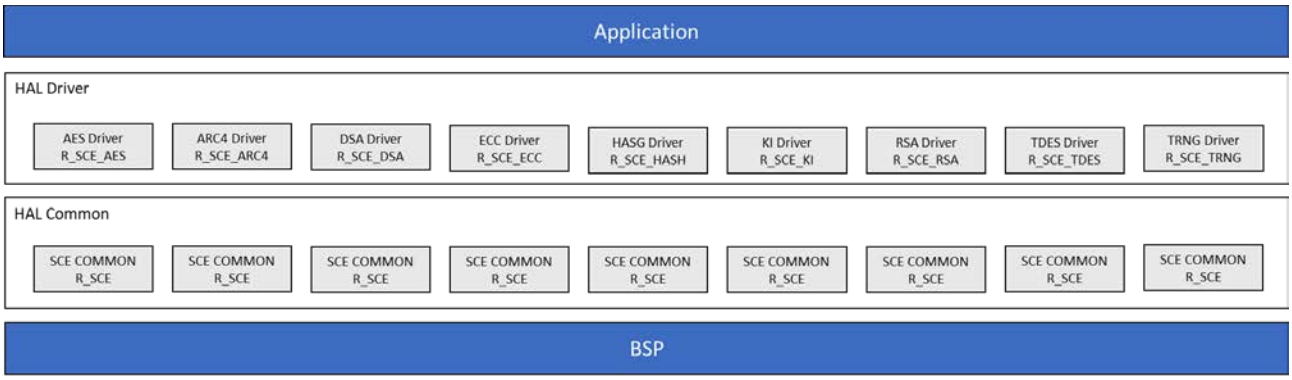


Figure14.39 SCE HAL Module Block Diagram

Note: The prior figure shows all nine available crypto modules. The SCE COMMON module is repeated for each one since it is included when the module is added to a thread stack. Common modules can be referenced by multiple other module instances across multiple Synergy stacks.

Table 14.1 Support of MCU Groups: SCE Driver

Function	S7G2, S5D9, S5D5	S3A1, S3A3, S3A7, S3A6	S1JA, S124, S128
TRNG	Generate and read random number	Generate and read random number	Generate and read random number
AES	Encryption, decryption, key generation – wrapped keys	Encryption, decryption, , key generation – wrapped keys	Encryption, decryption
AES Key Size	128-bit, 192-bit, 256-bit	128-bit, 256-bit	128-bit, 256-bit
AES Key Type	Plain text / raw key, Wrapped key	Plain text / raw key, Wrapped key	Plain text / raw key
AES Chaining Modes	ECB, CBC, CTR, GCM, XTS†	ECB, CBC, CTR, GCM, XTS	ECB, CBC, CTR
ARC4	Encryption, decryption	N/A	N/A
DSA	Signature Generation, Signature Verification	N/A	N/A
DSA Key Size	(1024, 128)-bit, (2048, 224)-bit, (2048, 256)-bit	N/A	N/A
ECC	• Key Generation – plain text and wrapped keys; • Scalar Multiplication; • ECDSA- Signature Generation • ECDSA -Signature Verification	N/A	N/A
ECC Key Size	P-192, P-256	N/A	N/A
ECC Key Type	Plain text / raw keys and wrapped keys	N/A	N/A
HASH	SHA1, SHA224, SHA256, MD5	N/A	N/A
Key installation	AES, ECC, RSA keys	AES keys	N/A
TDES	Encryption, decryption	N/A	N/A
TDES Key Size	192-bit	N/A	N/A
TDES Chaining Modes	ECB, CBC, CTR	N/A	N/A

Function	S7G2, S5D9, S5D5	S3A1, S3A3, S3A7, S3A6	S1JA, S124, S128
RSA	• Signature Generation • Signature Verification • Public-key Encryption • Private-key Decryption • Key Generation – plain text and wrapped keys	N/A	N/A
RSA Key Size	1024-bit, 2048-bit	N/A	N/A
RSA Key Type	Plain text / raw key, wrapped key	N/A	N/A

Framework Interfaces for Cryptographic Functions (sf_crypto) available for this release includes: HASH, TRNG, Key Generation (RSA plain text and wrapped keys; AES wrapped keys; and, ECC wrapped keys), Cipher (RSA and AES encryption and decryption), Signature (RSA Signature generation and verification), and key Installation (AES, RSA and ECC keys).

† XTS is supported for 128-bit and 256-bit keys only.

14.39 Serial Communication Interface I²C over SCI (SCI_I2C)

The SCI_I2C driver supports the SCI peripheral in Synergy MCUs. The driver supports the I²C protocol for communicating in **master mode only** and provides following capabilities:

- Read from an I²C slave device with 7 or 10 bit address.
- Write to an I²C slave device with 7 or 10 bit address.
- Reset the I²C peripheral.
- Set the communication slave address dynamically prior to an I²C transaction.
- Interrupt driven transmit/receive processing.
- Transfer rate up to 400 kbps (Fast mode).
- Callback function support which can return an event code.
- Includes bit-rate modulation function on all SCI modules

The callback functions will be called with the following events:

- Transfer aborted
- Transmit complete
- Receive complete

The callback structure provides the number of bytes that were sent or received. The I²C on SCI driver implements the I²C interface in SSP.

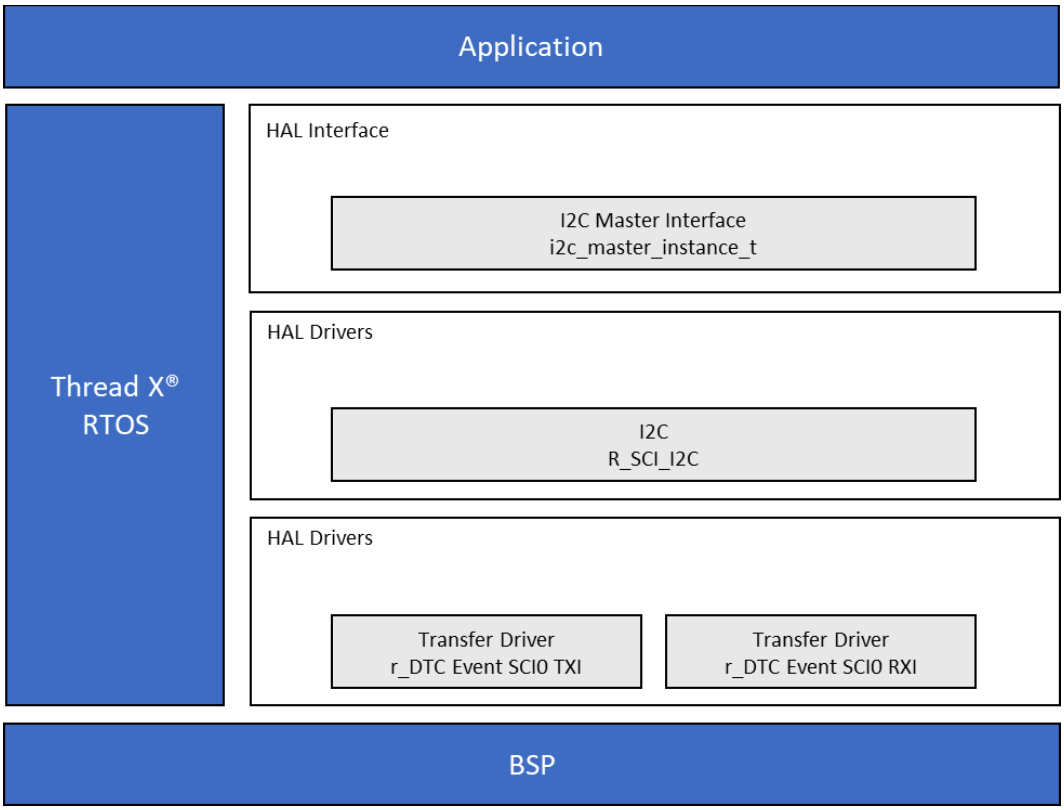


Figure 14.40 Serial Communication Interface I²C over SCI (SCI_I2C)

The following hardware features are, or are not, supported by SSP for the I²C over SPI.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Master Mode	Slave mode	Support all Interrupt Sources	Digital noise filter	Bit rate modulation	SDA delay
S124	✓	N/A	ERI not supported	☒	✓	✓
S128	✓	N/A	ERI not supported	☒	✓	✓
S1JA	✓	N/A	ERI not supported	☒	✓	✓
S3A1	✓	N/A	ERI not supported	☒	✓	✓
S3A3	✓	N/A	ERI not supported	☒	✓	✓
S3A6	✓	N/A	ERI not supported	☒	✓	✓
S3A7	✓	N/A	ERI not supported	☒	✓	✓
S5D5	✓	N/A	ERI not supported	☒	✓	✓
S5D9	✓	N/A	ERI not supported	☒	✓	✓
S7G2	✓	N/A	ERI not supported	☒	✓	✓

MCU Group	Timeout on bus lockout
S124	N/A
S128	N/A
S1JA	N/A

MCU Group	Timeout on bus lockout
S3A1	N/A
S3A3	N/A
S3A6	N/A
S3A7	N/A
S5D5	N/A
S5D9	N/A
S7G2	N/A

14.40 Serial Communication Interface SPI (SCI_SPI)

The SCI SPI HAL module provides high-level APIs for master-based SPI serial communications and is implemented on the r_sci_spi (simple SPI) module. The SCI SPI HAL module configures and controls the SPI functionality of a Synergy MCU using the SCI (Serial Communications Interface) peripheral. The module is configured in the ISDE. A user-defined callback can be created to signal when the SPI has transmitted data, aborted a data transfer, or detected an error condition.

SCI SPI HAL modules are enabled, with data transfer support, by incorporating the DTC HAL module of the MCU to perform SCI SPI transfer through the DTC without intervention of the CPU.

SCI SPI HAL Module Features

- Driver initialization
- Serial communication through SPI operation using 8-bit data transfers
- Configurable among four clock phase and clock polarity settings
- Support for callbacks. The callback functions are called with the following events:
  - Transfer aborted
  - Transfer complete
  - Over run error
- SPI communication in master mode.

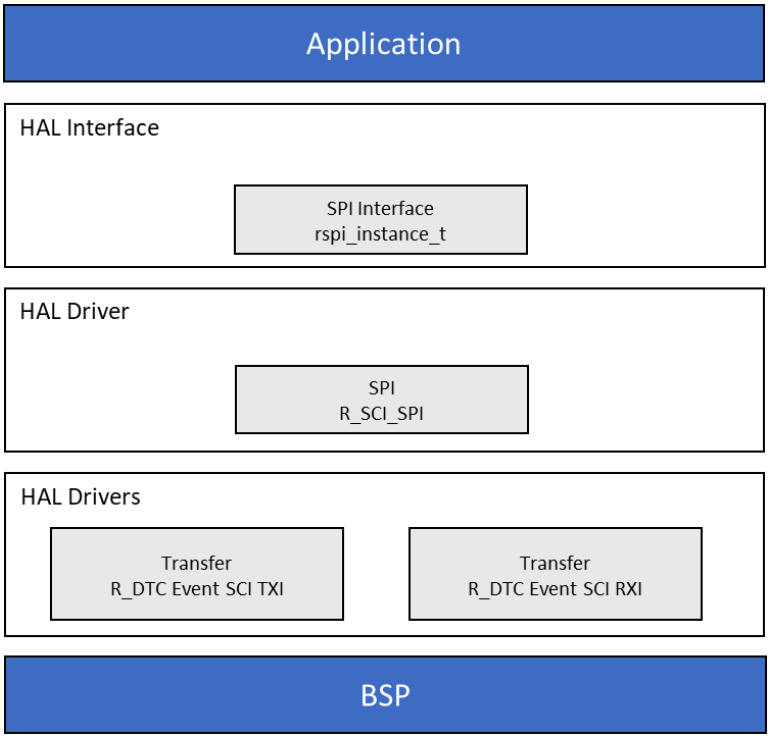


Figure 14.41 Serial Communication Interface SPI

The following hardware features are, or are not, supported by SSP for the SCI_SPI:

MCU Group	Master Out/Slave In (MOSI)	Master In/Slave Out (MISO)	SSL Slave Select	SPI Operation (4-wire method)	Clock Synchronous Operation (3-wire method)	Full Duplex or Transmit Only selectable
S124	✓	✓	GPIO	N/A	✓	N/A
S128	✓	✓	GPIO	N/A	✓	N/A
S1JA	✓	✓	GPIO	N/A	✓	N/A
S3A1	✓	✓	GPIO	N/A	✓	N/A
S3A3	✓	✓	GPIO	N/A	✓	N/A
S3A6	✓	✓	GPIO	N/A	✓	N/A
S3A7	✓	✓	GPIO	N/A	✓	N/A
S5D5	✓	✓	GPIO	N/A	✓	N/A
S5D9	✓	✓	GPIO	N/A	✓	N/A
S7G2	✓	✓	GPIO	N/A	✓	N/A

MCU Group	Overrun error detection	Data format transfer bit length	Master or Slave	Clock source internal/external	Configurable phase and polarity	Interrupt sources support
S124	✓	8-bit	Master	internal	✓	✓
S128	✓	8-bit	Master	internal	✓	✓
S1JA	✓	8-bit	Master	internal	✓	✓
S3A1	✓	8-bit	Master	internal	✓	✓
S3A3	✓	8-bit	Master	internal	✓	✓
S3A6	✓	8-bit	Master	internal	✓	✓
S3A7	✓	8-bit	Master	internal	✓	✓
S5D5	✓	8-bit	Master	internal	✓	✓
S5D9	✓	8-bit	Master	internal	✓	✓
S7G2	✓	8-bit	Master	internal	✓	✓

14.41 Serial Communication Interface UART (SCI_UART)

The UART HAL Module provides high-level APIs for UART applications and is implemented on r_sci_uart. The UART HAL module uses the SCI peripherals on the Synergy MCU. A user-defined callback can be created to manage hardware-handshake and data operation, if needed.

The UART HAL module supports the standard UART protocol. The UART HAL module used in concert with the SCI peripheral in UART mode (UART on SCI) supports the following features (in addition to the standard UART protocol):

- Full-duplex UART communication
  - Simultaneous communication with multiple channels
  - Interrupt-driven data transmission and reception
  - Invoking the user-callback function with an event code in the argument
  - Baud-rate change at run-time
  - Hardware resource locking during UART transaction
  - CTS/RTS hardware flow control (with an associated IOPORT pin and supported by user-defined callback function)
  - Integration with the DTC transfer module
  - Abort in-progress read/write operations,
  - Abort only reception
  - Abort only transmission
  - Abort both transmission and reception

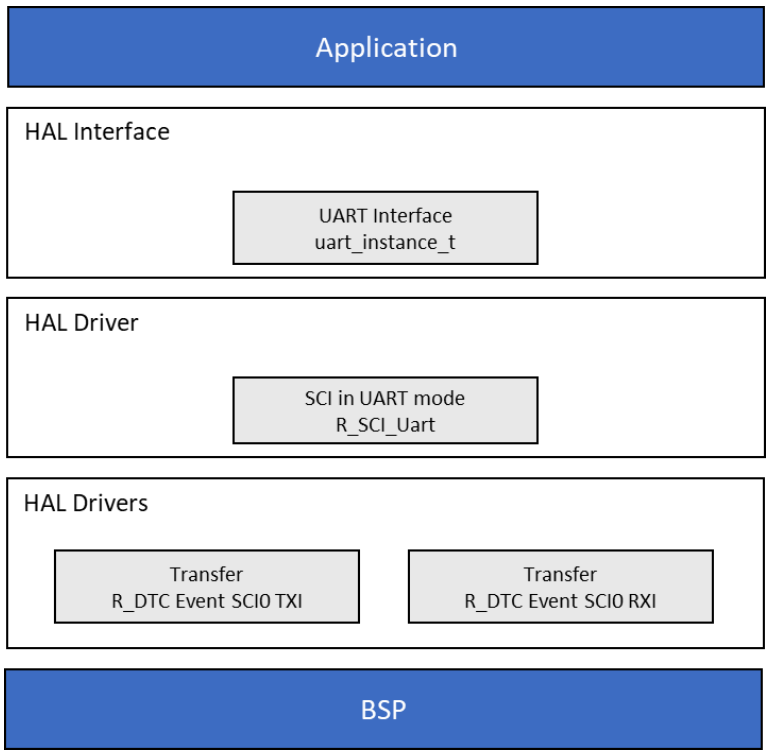


Figure 14.42 Serial Communication Interface UART

The following hardware features are, or are not, supported by SSP for the UART (SCI).

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Serial communication mode: Asynchronous	Serial communication mode: Clock synchronous	Serial communication mode: Smart card	Serial communication mode: Simple IIC	Serial communication mode: Simple SPI
S124	✓	✓	☒	✓	✓
S128	✓	✓	☒	✓	✓
S1JA	✓	✓	☒	✓	✓
S3A1	✓	✓	☒	✓	✓
S3A3	✓	✓	☒	✓	✓
S3A6	✓	✓	☒	✓	✓
S3A7	✓	✓	☒	✓	✓
S5D5	✓	✓	☒	✓	✓
S5D9	✓	✓	☒	✓	✓
S7G2	✓	✓	☒	✓	✓

MCU Group	Bit selectable transfer speed	Data Length 7, 8 or 9 bits	Support all Interrupt Sources	Transmission stop bit 1 or 2 bits	Parity Even parity, odd parity, or no parity	Receive error detection
S124	✓	✓	✓	✓	✓	✓
S128	✓	✓	✓	✓	✓	✓
S1JA	✓	✓	✓	✓	✓	✓

MCU Group	Bit selectable transfer speed	Data Length 7, 8 or 9 bits	Support all Interrupt Sources	Transmission stop bit 1 or 2 bits	Parity Even parity, odd parity, or no parity	Receive error detection
S3A1	✓	✓	✓	✓	✓	✓
S3A3	✓	✓	✓	✓	✓	✓
S3A6	✓	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓	✓
S5D5	✓	✓	✓	✓	✓	✓
S5D9	✓	✓	✓	✓	✓	✓
S7G2	✓	✓	✓	✓	✓	✓

MCU Group	Hardware flow control	Transmission and reception	Address match	Address un-match	Start-bit detection	Break detection
S124	✓	Register	☒	☒	✓	☒
S128	✓	Register	☒	☒	✓	☒
S1JA	✓	Register	☒	☒	✓	☒
S3A1	✓	FIFO	☒	☒	✓	☒
S3A3	✓	FIFO	☒	☒	✓	☒
S3A6	✓	FIFO	☒	☒	✓	☒
S3A7	✓	FIFO	☒	☒	✓	☒
S5D5	✓	FIFO	☒	☒	✓	☒
S5D9	✓	FIFO	☒	☒	✓	☒
S7G2	✓	FIFO	☒	☒	✓	☒

MCU Group	Clock source Internal/ external	Double-speed mode	Multi-processor communications	Noise cancellation	Bit rate modulation function	iRDA support
S124	✓	☒	☒	✓	✓	☒
S128	✓	☒	☒	✓	✓	☒
S1JA	✓	☒	☒	✓	✓	☒
S3A1	✓	☒	☒	✓	✓	☒
S3A3	✓	☒	☒	✓	✓	☒
S3A6	✓	☒	☒	✓	✓	☒
S3A7	✓	☒	☒	✓	✓	☒
S5D5	✓	☒	☒	✓	✓	☒
S5D9	✓	☒	☒	✓	✓	☒
S7G2	✓	☒	☒	✓	✓	☒

14.42 Serial Peripheral Interface (RSPI)

The SPI HAL module is a generic API for communication using the SPI protocol. The module supports both the SPI and SCI peripherals available on the Synergy microcontroller hardware and is implemented on r_rspi and r_sci_spi. This section refers to the r_rspi HAL module, which is referred as the SPI module (formerly known as RSPI). The SPI HAL module supports standard SPI master and slave mode communications functions. Callbacks are provided for transfer events.

The SPI HAL modules are enabled with data-transfer support by incorporating the Data Transfer Controller (DTC) module of the MCU. This performs SPI transfers through the DTC without requiring interrupt processing, by the CPU, for each frame.

The SPI HAL module support following key features:

- Initialization of the driver
- SPI transfer functions:
  - Allows serial communication through the SPI operation using the four-wire method
  - Capable of serial communication in master and slave modes
  - Switching the polarity of the serial transfer clock
  - Switching the phase of the serial transfer clock
- Data Format
  - MSB-first/LSB-first selectable
  - Transfer bit length is selectable as 8, 16 and 32 bits
- Error Detection
  - Mode fault detection
  - Overrun error detection
  - Parity error detection
- SSL control functions
  - Internally select up to four SSL signals (SSLn0 to SSLn3) for each channel in master mode
  - External hardware slave select can be used in master mode
- Interrupts
  - RSPI receive interrupt (receive buffer full)
  - RSPI transmit interrupt (transmit buffer empty)
  - RSPI error interrupt (mode fault, overrun and parity error)
- Delays
  - Add SPI clock delay
  - Add slave select negation delay
  - Add next-access delay

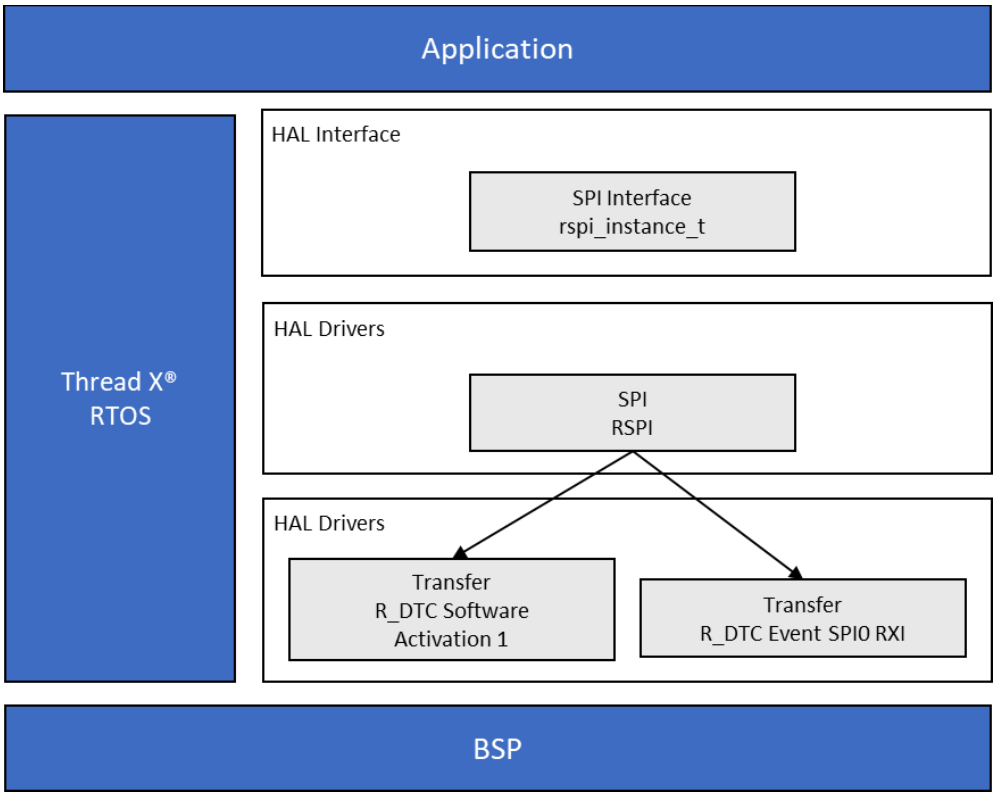


Figure 14.43 Serial Peripheral Interface

The following hardware features are, or are not, supported by SSP for the RSPI.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	SPI Operation mode	Clock Syn mode	Full-duplex or transmit-only can be selected	Switching of the polarity and Phase	Master and Slave mode	MSB first/LSB first selectable
S124	✓	✓	✓	✓	✓	✓
S128	✓	✓	✓	✓	✓	✓
S1JA	✓	✓	✓	✓	✓	✓
S3A1	✓	✓	✓	✓	✓	✓
S3A3	✓	✓	✓	✓	✓	✓
S3A6	✓	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓	✓
S5D5	✓	✓	✓	✓	✓	✓
S5D9	✓	✓	✓	✓	✓	✓
S7G2	✓	✓	✓	✓	✓	✓

MCU Group	Transfer length - 8/16/32	Up to four frames can be transferred in one round	Bit rate configuration	Double buffer configuration	Error detection	SSL control function
S124	✓	☒	✓	☒	✓	✓
S128	✓	☒	✓	☒	✓	✓

MCU Group	Transfer length - 8/16/32	Up to four frames can be transferred in one round	Bit rate configuration	Double buffer configuration	Error detection	SSL control function
S1JA	✓	☒	✓	☒	✓	✓
S3A1	✓	☒	✓	☒	✓	✓
S3A3	✓	☒	✓	☒	✓	✓
S3A6	✓	☒	✓	☒	✓	✓
S3A7	✓	☒	✓	☒	✓	✓
S5D5	✓	☒	✓	☒	✓	✓
S5D9	✓	☒	✓	☒	✓	✓
S7G2	✓	☒	✓	☒	✓	✓

MCU Group	Transfer of up to eight commands	All Interrupt sources	DTC Support	Event link function through ELC HAL driver	Function for switching between CMOS output and open-drain output	Loopback mode
S124	☒	✓	✓16-bit transfer	☒	✓	☒
S128	☒	✓	✓16-bit transfer	☒	✓	☒
S1JA	☒	✓	✓16-bit transfer	☒	✓	☒
S3A1	☒	✓	✓(32-bit transfer)	☒	✓	☒
S3A3	☒	✓	✓(32-bit transfer)	☒	✓	☒
S3A6	☒	✓	✓(32-bit transfer)	☒	✓	☒
S3A7	☒	✓	✓(32-bit transfer)	☒	✓	☒
S5D5	☒	✓	✓(32-bit transfer)	☒	✓	☒
S5D9	☒	✓	✓(32-bit transfer)	☒	✓	☒
S7G2	☒	✓	✓(32-bit transfer)	☒	✓	☒

MCU Group	Module Stop Function	Multi-master mode
S124	☒	☒
S128	☒	☒
S1JA	☒	☒
S3A1	☒	☒
S3A3	☒	☒
S3A6	☒	☒
S3A7	☒	☒
S5D5	☒	☒
S5D9	☒	☒
S7G2	☒	☒

14.43 Segment LCD (SLCDC)

The Segment LCD Controller HAL module provides high-level APIs for Segment LCD applications and is implemented on r_slcdc. The Segment LCD Controller HAL module displays data on a Segment LCD and modifies the displayed data. The Segment LCD Controller HAL module uses the Segment LCD Controller module on a Synergy MCU.

SLCDC HAL Module Features:

The SLCDC HAL module uses the Segment LCD Controller (SLCDC) to display data on a Segment LCD. The driver initializes the LCD for displaying data and configures the drive-voltage generator, the display waveform, the number of time slices, and the bias methods to drive the LCD. This module provides functions to display data to a specified set of segments, to update existing segment data, to enable and disable display, to set the display area, and to adjust the contrast.

The SLCDC HAL Module other features include:

- Internal voltage-boosting for the LCD driver voltage generator: select the capacitor split method or the external resistance division.
- Display bias: select the 1/2 bias method, 1/3 bias method, or 1/4 bias method.
- Time slice of the display: select static, 2-time slice, 3-time slice, 4-time slice, or 8-time slice.
- Display waveform: select waveform A or waveform B.
- Display data area: select A-pattern, B-pattern, or blinking. You can switch the display data area.
- Use the RTC periodic interrupt (PRD) to generate a blinking display with A-pattern and B-pattern.
- Adjust the reference voltage (which is generated when operating the voltage boost circuit) in 16 steps (contrast adjustment).

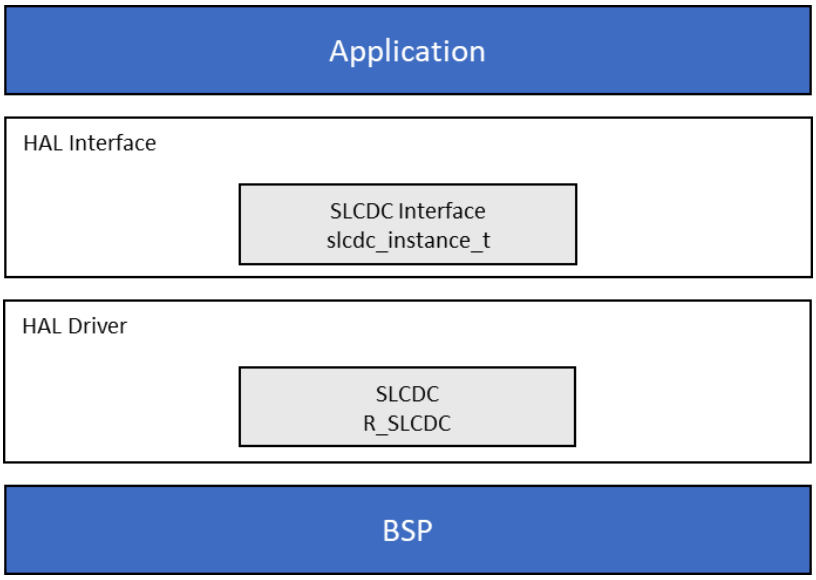


Figure 14.44 Segment LCD

The following hardware features are, or are not, supported by SSP for the SLCD.

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Liquid crystal waveform (waveform A or B) selectable	Voltage Generator-Internal voltage boosting method, and external resistance division method	Voltage Generator-Capacitor split method	LCD blinking and display functions	Source Clock support: Main Clock Oscillator	Source Clock support: Sub Clock Oscillator
S124	N/A	N/A	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	N/A	N/A	N/A
S1JA	N/A	N/A	N/A	N/A	N/A	N/A
S3A1	✓	✓	☒	✓	☒	☒
S3A3	✓	✓	☒	✓	☒	☒
S3A6	✓	✓	☒	✓	☒	☒
S3A7	✓	✓	☒	✓	☒	☒
S5D5	N/A	N/A	N/A	N/A	N/A	N/A
S5D9	N/A	N/A	N/A	N/A	N/A	N/A
S7G2	N/A	N/A	N/A	N/A	N/A	N/A

MCU Group	Source Clock support: Low speed Clock Oscillator	Source Clock support: High speed Clock Oscillator	Time slice modes – Static, 4	Time slice modes – 1, 2, 3 and 8	Bias method 2, 3, 4	Contrast adjustment
S124	N/A	N/A	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	N/A	N/A	N/A
S1JA	N/A	N/A	N/A	N/A	N/A	N/A
S3A1	✓	✓	✓	☒	✓	✓
S3A3	✓	✓	✓	☒	✓	✓
S3A6	✓	✓	✓	☒	✓	✓
S3A7	✓	✓	✓	☒	✓	✓
S5D5	N/A	N/A	N/A	N/A	N/A	N/A
S5D9	N/A	N/A	N/A	N/A	N/A	N/A
S7G2	N/A	N/A	N/A	N/A	N/A	N/A

14.44 Serial Sound Interface (SSI) — also known as I²S

The I²S HAL Driver module provides high-level APIs for the generic I²S audio serial communication protocol; it is used to send or receive uncompressed audio data in master mode.

I²S HAL Module Features

The I²S Driver used with the SSI peripheral in I²S master mode that supports the following features, as well as the standard I²S protocol.

- Full-duplex I²S communication (SSI channel 0 only)
- Interrupt driven data transmission and reception
- Integration with the DTC transfer module.
- A user-defined callback can be created to respond to the need for additional data.

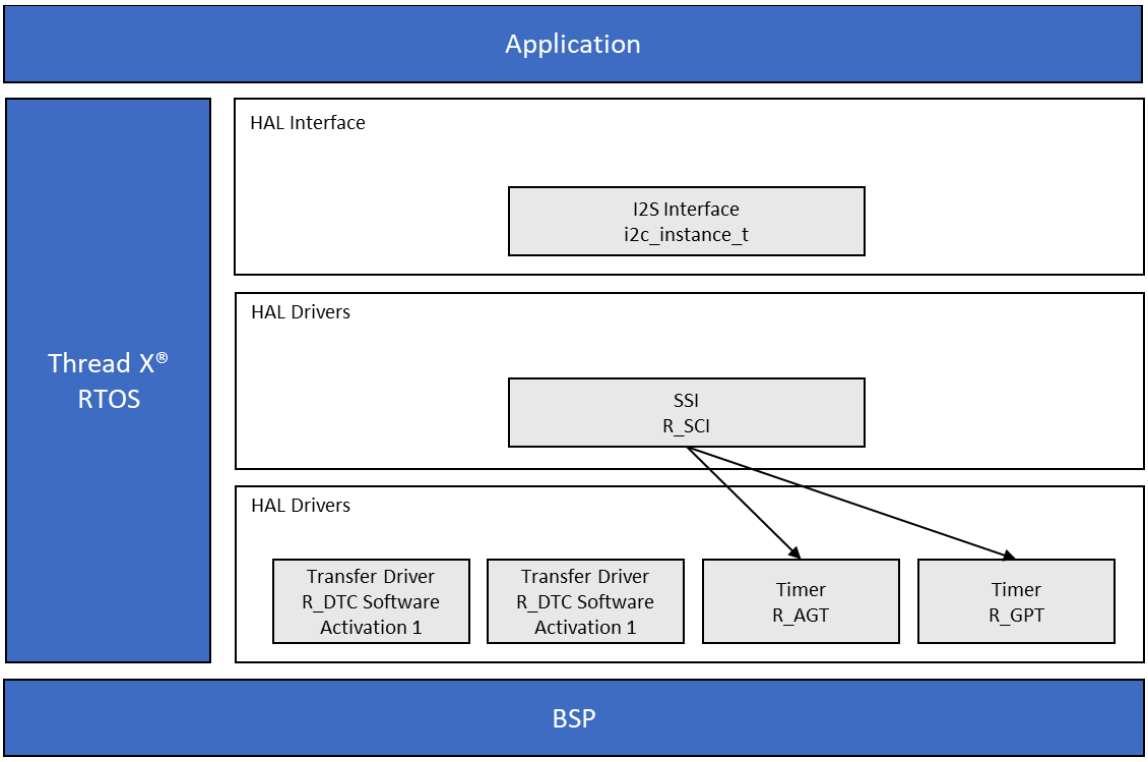


Figure 14.45 Serial Sound Interface (SSI)

The following hardware features are, or are not, supported by SSP for SSI:

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Supports 2 Channels	SSI Format Support	MSB-first Format Support	Serial bit clock configurable {16, 32, 48, and 64 sampling rate}	Master clock input from the master clock pin for audio (AUDIO_CLK)	Master clock input from the master clock pin for GPT output (GTIOC1A)
S124	N/A	N/A	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	N/A	N/A	N/A
S1JA	N/A	N/A	N/A	N/A	N/A	N/A
S3A1	☒	☒	☒	✓	✓	☒
S3A6	☒	☒	☒	✓	✓	☒
S3A3	☒	☒	☒	✓	✓	☒
S3A7	☒	☒	☒	✓	✓	☒
S5D5	✓	☒	☒	✓	✓	☒
S5D9	✓	☒	☒	✓	✓	☒
S7G2	✓	☒	☒	✓	✓	☒

MCU Group	Ability to select Stop Word SSIWS	Accepts Interrupts from Communication Errors	Accepts Interrupts from Receive Data Full	Accepts Interrupts from Transmit Data Empty	Internal Connection to GPT output GTIOC1A
S124	☒	☒	☒	☒	N/A
S128	☒	☒	☒	☒	N/A
S1JA	N/A	N/A	N/A	N/A	N/A
S3A1	☒	✓	✓	✓	✓
S3A3	☒	✓	✓	✓	✓
S3A6	☒	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓
S5D5	☒	✓	✓	✓	✓
S5D9	☒	✓	✓	✓	✓
S7G2	✓	✓	✓	✓	✓

14.45 Sigma-Delta ADC (SDADC)

The SDADC HAL module implements the ADC API for analog-to-digital conversions on r_sdadc. It supports the SDADC24 24-bit analog-to-digital converter peripheral available on the Synergy microcontroller hardware. A user-defined callback can be created to process the data each time a new sample is available.

- 24-bit sigma delta A/D Converter
- Single scan or continuous scan operation mode
- Single-ended or differential input
- Gain of up to 32 on differential inputs
- Oversampling ratio configurable on differential inputs

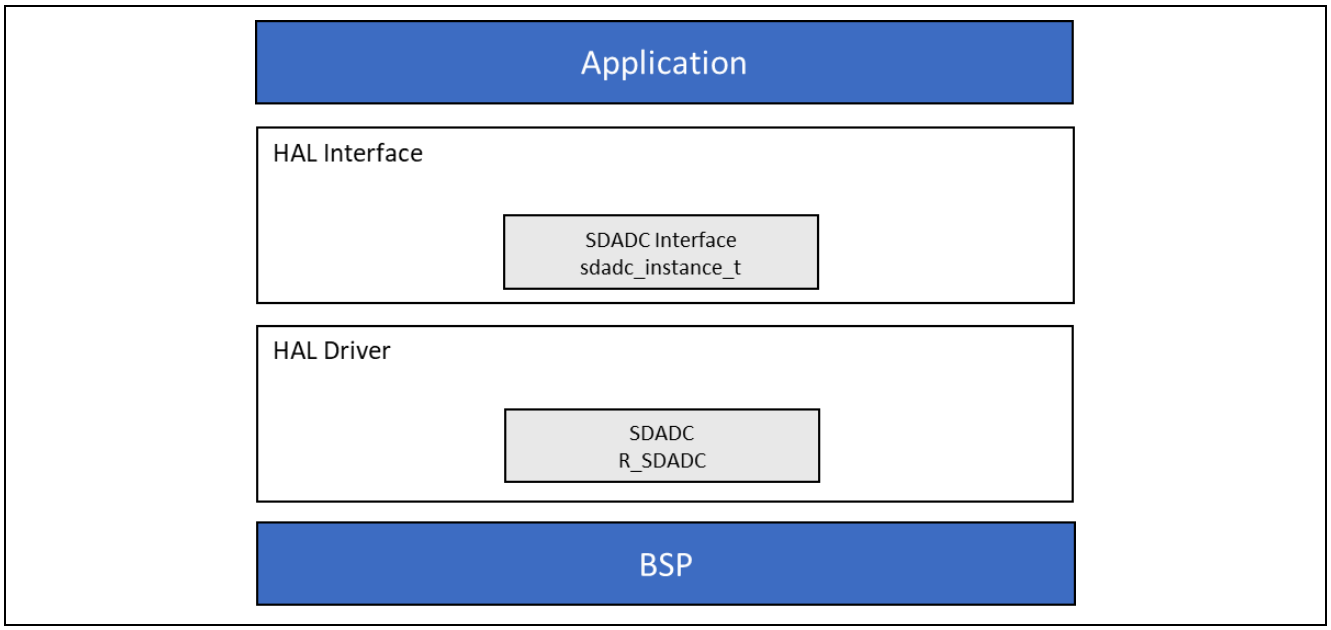


Figure 14.46 SDADC HAL Module Block Diagram

The following hardware features are, or are not, supported by SSP for SDADC:

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Support for all Analog Channels	24-Bit	Single-scan Mode	Continuous-scan mode
S124	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	N/A
S1JA	✓	✓	✓	✓
S3A1	N/A	N/A	N/A	N/A
S3A3	N/A	N/A	N/A	N/A
S3A6	N/A	N/A	N/A	N/A
S3A7	N/A	N/A	N/A	N/A
S5D5	N/A	N/A	N/A	N/A
S5D9	N/A	N/A	N/A	N/A
S7G2	N/A	N/A	N/A	N/A

MCU Group	Single-ended input	Differential input	Programmable Gain Amplifier	Configurable oversampling ratio
S124	N/A	N/A	N/A	N/A
S128	N/A	N/A	N/A	N/A
S1JA	✓	✓	✓	✓
S3A1	N/A	N/A	N/A	N/A
S3A3	N/A	N/A	N/A	N/A
S3A6	N/A	N/A	N/A	N/A
S3A7	N/A	N/A	N/A	N/A
S5D5	N/A	N/A	N/A	N/A
S5D9	N/A	N/A	N/A	N/A
S7G2	N/A	N/A	N/A	N/A

14.46 Watchdog Timer (WDT)

The WDT (Watchdog Timer) HAL module provides high-level APIs for WDT applications and is implemented on r_wdt. The WDT HAL module uses the WDT peripheral on the Synergy MCU. A user-defined callback can be created to respond to event notifications.

WDT HAL Module Features

- When the WDT underflows or is refreshed outside of the permitted refresh window, one of the following events can occur:
  - Reset of the device
  - Generation of an NMI
- Supports the Watchdog Timer (WDT) peripheral, which uses an external clock.
- Can be configured in register start mode through the WDT registers.
- Supports automatic hardware configuration after reset.
- Can be started from the application.

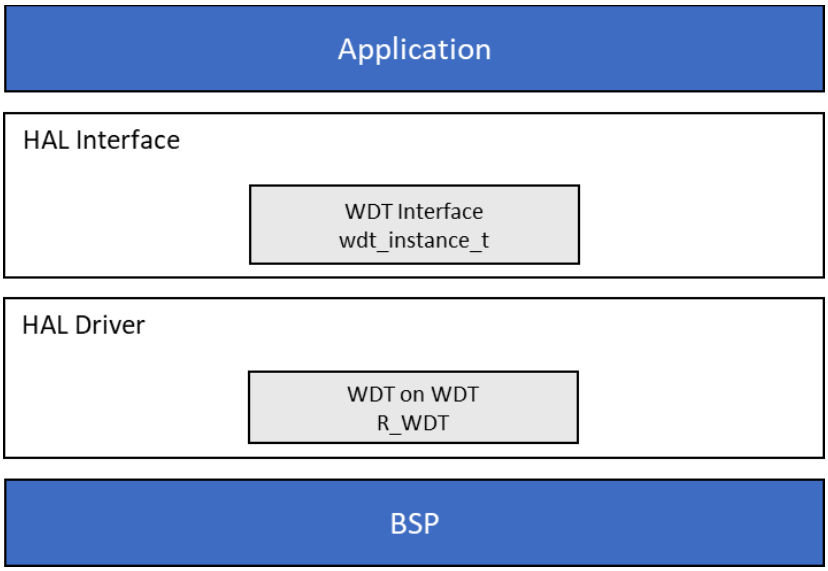


Figure 14.47 Watchdog Timer

The following hardware features are, or are not, supported by SSP for WDT:

Legend:

✓	Available (Tested)
☒	Not Available (Not tested/not functional or both)
N/A	Not supported by MCU

MCU Group	Clock Divide by 4, 64, 128, 512, 2,048, or 8,192	Count down	Register-start mode	Auto-start mode	Reset output	Interrupt request output
S124	✓	✓	✓	✓	✓	✓
S128	✓	✓	✓	✓	✓	✓
S1JA	✓	✓	✓	✓	✓	✓
S3A1	✓	✓	✓	✓	✓	✓
S3A3	✓	✓	✓	✓	✓	✓
S3A6	✓	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓	✓
S5D5	✓	✓	✓	✓	✓	✓
S5D9	✓	✓	✓	✓	✓	✓
S7G2	✓	✓	✓	✓	✓	✓

MCU Group	Sleep mode count stop control output	Event link function through ELC HAL driver	Window function	Conditions for stopping the Counter – reset/underflow-refresh error	Refresh error and under flow error detect	Reading the counter value
S124	☒	☒	✓	✓	✓	✓
S128	☒	☒	✓	✓	✓	✓
S1JA	☒	☒	✓	✓	✓	✓
S3A1	☒	☒	✓	✓	✓	✓
S3A3	☒	☒	✓	✓	✓	✓
S3A6	☒	☒	✓	✓	✓	✓
S3A7	☒	☒	✓	✓	✓	✓

MCU Group	Sleep mode count stop control output	Event link function through ELC HAL driver	Window function	Conditions for stopping the Counter – reset/underflow-refresh error	Refresh error and under flow error detect	Reading the counter value
S5D5	☒	☒	✓	✓	✓	✓
S5D9	☒	☒	✓	✓	✓	✓
S7G2	☒	☒	✓	✓	✓	✓

15. Board Support Package (BSP)

The SSP includes board support packages (BSPs) for DK-S7G2, SK-S7G2, PK-S5D9, DK-S3A7, DK-S128, DK-S124, PE-HMI1 TB-S5D5, TB-S3A6, and TB-S3A3 target boards.

The BSP is responsible for getting the MCU from reset to the user's application (the `main()` function). Before reaching the user's application, the BSP sets up the stacks, heap, clocks, interrupts, and a C runtime environment. The BSP also configures and sets up the port I/O pins and performs any board specific initializations.

The key features of the BSPs provided with SSP are:

- Support for designated kits
- Creation of custom BSPs using e^2 studio or IAR Embedded Workbench® for Renesas Synergy™
- System initialization and configuration during startup
- Software and hardware locking/unlocking
- Register protection
- CMSIS compliant
- Standardized definitions for processor peripherals
- Standardized access functions to access processor features
- Standardized function names for system exception handlers
- Standardized functions for system initialization.
- Standardized software variables for clock speed information

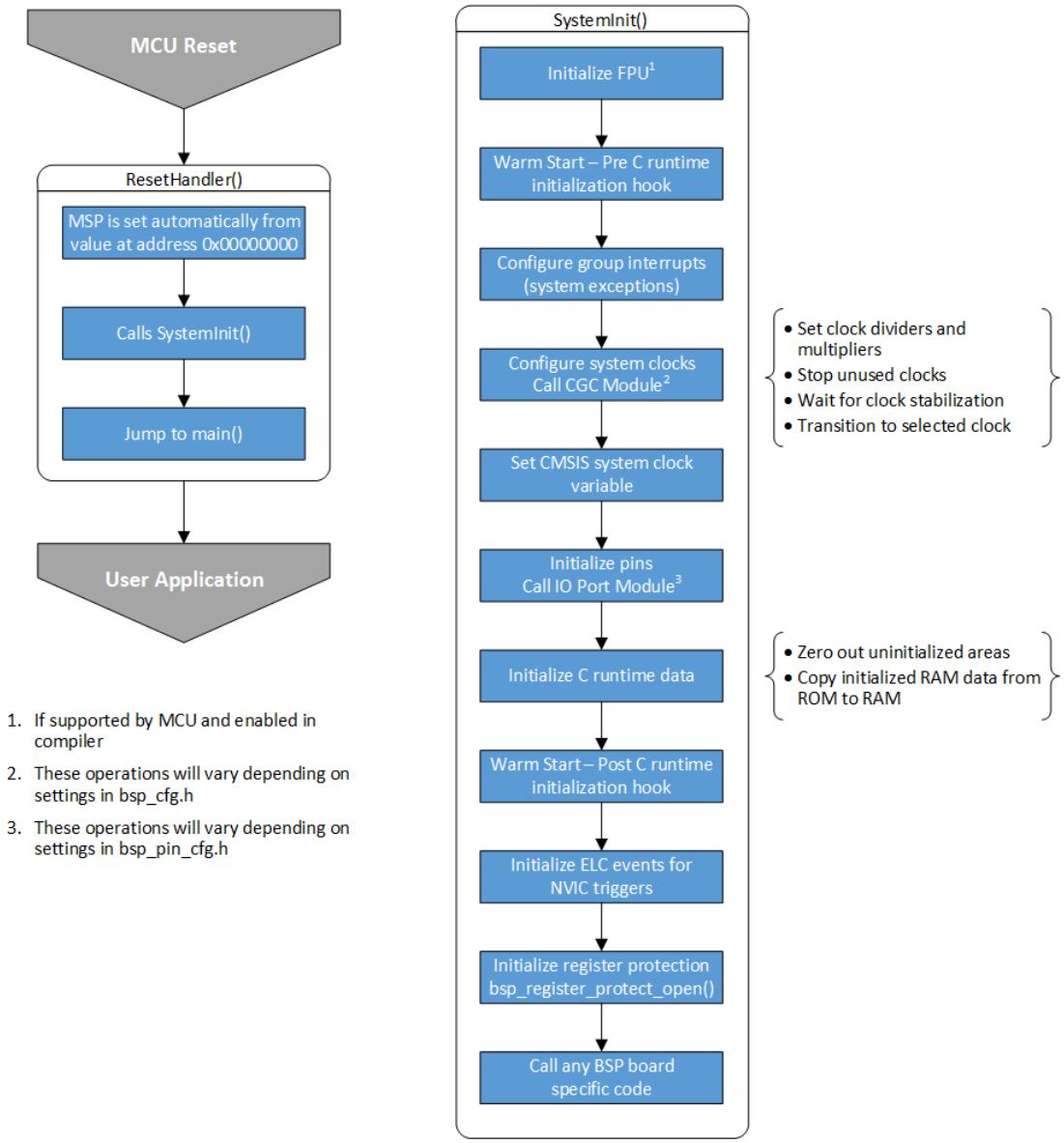


Figure 15.1 Board Support Package Flow of Events

16. Memory Size Estimation

Estimating memory size requirements for a design can be difficult. Design, coding, compiler options, and even error-checking can affect the resultant binary's sizes, as well as their runtime memory requirements. In the following sections, various configurations and options are shown to give the Developer useful information in which to judge how much ROM is required given the inclusion of certain features.

The estimated memory requirements provided specify the estimated memory consumption for each module. The memory usage for the modules include:

ROM Usage = .text + .data

Note: These are estimated memory requirements. Different build options can give different results. These estimates are accurate to +/- 5%.

For ThreadX® and the X-Ware™ components, the ability to turn compiler-time error checking on, or off, affects memory sizes, so Renesas shows both conditions. Using ThreadX as an example, if error checking is turned off, the resultant ROM sizes are (for "event"):

Name	ROM
Event	1156

If error checking is turned on, the following amounts need to be **added** to the ROM size (in bytes):

Name	ROM
Event	360

Thus, the total bytes needed for ROM for this ThreadX function is 1516, if every function on the "event" directory is being used.

Notes:

- Memory sizes for all possible options are not covered by this datasheet.
- Various compiler tool chains produce different memory sizes, and compiling for an Arm® Cortex M4 architecture is different from compiling for an Arm® Cortex M0+.
- Arm® Cortex M4 means devices from S3A3, S3A6, S3A7, S5D5, S5D9, and S7G2 Synergy MCU Groups.
- Arm® Cortex M0+ means devices from S124 and S128 Synergy MCU Group.
- The process Renesas uses to calculate Flash memory size is as follows:
- Compile source code to create the object file.
- Generate the memory footprint using the following command:

```
arm-none-eabi-size --format=Berkeley  <filename>.o
```

These tables list the ROM memory sizes in bytes for the SSP v1.4.0 build using both GCC –O2 and IAR –OH (speed). Memory sizes are dependent on build options and sizes provided should be considered an estimate, not a guarantee. By having this data in a single location, the Developer can quickly determine the ROM memory required for a given function and can easily aggregate different functions to get sizing for their underlying Renesas Synergy Platform needs.

16.1 GCC ROM Memory Size Estimation

Table 16.1 GCC ROM results compiled for various MCU Groups

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
_bsp	bsp_cache	-	8	8	112	112	112	112	112	112	112
_bsp	bsp_clocks	-	100	100	320	332	328	320	413	429	468
_bsp	bsp_common	-	92	92	84	84	84	84	84	84	84
_bsp	bsp_common_leds	-	24	24	32	32	32	32	32	32	32
_bsp	bsp_delay	-	100	100	100	100	100	100	100	100	100
_bsp	bsp_feature	-	248	276	308	308	308	316	272	272	272
_bsp	bsp_fmi_R7FS124*	-	1024	-	-	-	-	-	-	-	-
_bsp	bsp_fmi_R7FS128*	-	-	1024	-	-	-	-	-	-	-
_bsp	bsp_fmi_R7FS3A1*	-	-	-	1024	-	-	-	-	-	-
_bsp	bsp_fmi_R7FS3A3*	-	-	-	-	1024	-	-	-	-	-
_bsp	bsp_fmi_R7FS3A6*	-	-	-	-	-	1024	-	-	-	-
_bsp	bsp_fmi_R7FS3A7*	-	-	-	-	-	-	1024	-	-	-
_bsp	bsp_fmi_R7FS5D5*	-	-	-	-	-	-	-	1024	-	-
_bsp	bsp_fmi_R7FS5D9*	-	-	-	-	-	-	-	-	1024	-
_bsp	bsp_fmi_R7FS7G2*	-	-	-	-	-	-	-	-	-	1024
_bsp	bsp_group_irq	-	468	516	612	612	612	612	612	612	612
_bsp	bsp_hw_locks	-	220	232	308	344	336	336	432	432	432
_bsp	bsp_init	-	4	4	4	4	4	4	16	16	28
_bsp	bsp_irq	-	96	96	96	96	96	96	96	96	96
_bsp	bsp_leds	-	14	14	16	16	16	16	12	14	16
_bsp	bsp_locking	-	300	300	308	308	308	308	308	308	308
_bsp	bsp_module_stop	-	250	270	274	290	290	274	282	282	282
_bsp	bsp_qspi	-	-	-	1013	1013	-	1013	1009	1009	1021
_bsp	bsp_register_protection	-	188	188	204	204	204	204	204	204	204
_bsp	bsp_rom_registers	-	24	76	76	76	76	76	76	76	76
_bsp	bsp_sbrk	-	68	68	68	68	68	68	68	68	68
_bsp	bsp_sdram	-	-	-	-	-	-	-	336	336	336
adc	r_adc	-	3920	3920	3556	3556	3556	3552	3552	3552	3552
adc_periodic	sf_adc_periodic	-	1060	1060	1068	1068	1068	1068	1068	1068	1068
agt	r_agt	-	2286	2286	1982	1982	1982	1978	1978	1978	1978

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
analog	r_acmphs	-	-	-	-	-	-	-	-	-	-
analog	r_acmplp	-	-	-	-	-	-	-	-	-	-
analog	r_opamp	-	-	872	-	-	-	-	-	-	-
analog	r_sdadc	-	-	-	-	-	-	-	-	-	-
audio_playback	sf_audio_playback	-	3001	3001	2849	2849	2849	2849	2849	2849	2849
audio_playback_dac	sf_audio_playback_hw_dac	-	752	840	788	788	788	788	788	788	788
audio_playback_i2s	sf_audio_playback_hw_i2s	-	-	-	268	268	268	268	268	268	268
audio_record_adc	sf_audio_record_adc	-	436	-	-	388	388	388	388	388	388
ble	ble_rl78g1d_if	-	-	-	-	-	-	-	-	-	8693
ble	ble_rl78g1d_prf_anp_if	-	-	-	-	-	-	-	-	-	1095
ble	ble_rl78g1d_prf_blp_if	-	-	-	-	-	-	-	-	-	1363
ble	ble_rl78g1d_prf_fmp_if	-	-	-	-	-	-	-	-	-	525
ble	ble_rl78g1d_prf_hid_if	-	-	-	-	-	-	-	-	-	2044
ble	ble_rl78g1d_prf_hrp_if	-	-	-	-	-	-	-	-	-	1283
ble	ble_rl78g1d_prf_htp_if	-	-	-	-	-	-	-	-	-	1350
ble	ble_rl78g1d_prf_if	-	-	-	-	-	-	-	-	-	1912
ble	ble_rl78g1d_prf_paps_if	-	-	-	-	-	-	-	-	-	1187
ble	ble_rl78g1d_prf_pxp_if	-	-	-	-	-	-	-	-	-	899
ble	ble_rl78g1d_prf_scps_if	-	-	-	-	-	-	-	-	-	668
ble	ble_rl78g1d_prf_tip_if	-	-	-	-	-	-	-	-	-	1228
ble	ble_serial	-	-	-	-	-	-	-	-	-	1407
ble	bmi	-	-	-	-	-	-	-	-	-	1505
ble	rble_api_anpc	-	-	-	-	-	-	-	-	-	552
ble	rble_api_anps	-	-	-	-	-	-	-	-	-	336
ble	rble_api_blpc	-	-	-	-	-	-	-	-	-	788
ble	rble_api_blps	-	-	-	-	-	-	-	-	-	328
ble	rble_api_cppc	-	-	-	-	-	-	-	-	-	1044
ble	rble_api_cpps	-	-	-	-	-	-	-	-	-	968
ble	rble_api_cscpc	-	-	-	-	-	-	-	-	-	824

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
ble	rble_api_cscps	-	-	-	-	-	-	-	-	-	312
ble	rble_api_fmpl	-	-	-	-	-	-	-	-	-	256
ble	rble_api_fmpt	-	-	-	-	-	-	-	-	-	132
ble	rble_api_gap	-	-	-	-	-	-	-	-	-	1968
ble	rble_api_gatt	-	-	-	-	-	-	-	-	-	1104
ble	rble_api_glpc	-	-	-	-	-	-	-	-	-	992
ble	rble_api_glps	-	-	-	-	-	-	-	-	-	556
ble	rble_api_hgbh	-	-	-	-	-	-	-	-	-	1408
ble	rble_api_hghd	-	-	-	-	-	-	-	-	-	504
ble	rble_api_hgrh	-	-	-	-	-	-	-	-	-	1560
ble	rble_api_hrpc	-	-	-	-	-	-	-	-	-	708
ble	rble_api_hrps	-	-	-	-	-	-	-	-	-	228
ble	rble_api_htpc	-	-	-	-	-	-	-	-	-	924
ble	rble_api_htpt	-	-	-	-	-	-	-	-	-	364
ble	rble_api_lnpc	-	-	-	-	-	-	-	-	-	1112
ble	rble_api_lnps	-	-	-	-	-	-	-	-	-	864
ble	rble_api_paspc	-	-	-	-	-	-	-	-	-	464
ble	rble_api_pasps	-	-	-	-	-	-	-	-	-	292
ble	rble_api_pxpm	-	-	-	-	-	-	-	-	-	468
ble	rble_api_pxpr	-	-	-	-	-	-	-	-	-	144
ble	rble_api_rscpc	-	-	-	-	-	-	-	-	-	928
ble	rble_api_rscps	-	-	-	-	-	-	-	-	-	384
ble	rble_api_sm	-	-	-	-	-	-	-	-	-	652
ble	rble_api_sppc	-	-	-	-	-	-	-	-	-	396
ble	rble_api_spps	-	-	-	-	-	-	-	-	-	220
ble	rble_api_tipc	-	-	-	-	-	-	-	-	-	644
ble	rble_api_tips	-	-	-	-	-	-	-	-	-	480
ble	rble_api_vs	-	-	-	-	-	-	-	-	-	992
ble	rble_host	-	-	-	-	-	-	-	-	-	1464
ble	rble_if_api_cb	-	-	-	-	-	-	-	-	-	31334
ble	rscip	-	-	-	-	-	-	-	-	-	320
ble	rscip_cntl	-	-	-	-	-	-	-	-	-	3436

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
ble	rscip_uart	-	-	-	-	-	-	-	-	-	1447
ble	sf_ble_rl78g1d	-	-	-	-	-	-	-	-	-	2610
ble	sf_ble_rl78g1d_onboard_profile	-	-	-	-	-	-	-	-	-	960
ble	timer	-	-	-	-	-	-	-	-	-	348
block_media_sdmmc	sf_block_media_qspi	-	-	-	-	844	-	844	844	844	844
block_media_sdmmc	sf_block_media_ram	-	-	-	-	364	-	364	-	364	364
block_media_sdmmc	sf_block_media_sdmmc	-	-	-	515	515	-	515	515	515	515
cac	r_cac	-	2272	2272	1912	1912	1912	1908	1908	1908	1908
can	hw_can	-	1880	1880	1620	1620	1620	1620	1620	1620	1620
can	r_can	-	2892	2892	2464	2464	2464	2464	2464	2464	2464
cellular	cellular_serial	-	-	-	-	-	813	813	813	813	813
cellular	sf_cellular_cat1	-	-	-	-	-	4772	4772	4772	4772	4772
cellular	sf_cellular_cat1_socket	-	-	-	-	-	636	636	636	636	636
cellular	sf_cellular_cat1_socket_private	-	-	-	-	-	4951	4951	4951	4951	4951
cellular	sf_cellular_cat3	-	-	-	-	-	1892	1892	1892	1892	1892
cellular	sf_cellular_cat3_socket	-	-	-	-	-	612	612	612	612	612
cellular	sf_cellular_cat3_socket_private	-	-	-	-	-	4888	4888	4888	4888	4888
cellular	sf_cellular_common	-	-	-	-	-	1648	1648	1648	1648	1648
cellular	sf_cellular_common_private	-	-	-	-	-	9395	9395	9395	9395	9395
cellular	sf_cellular_qctlcatm1	-	-	-	-	-	-	-	-	2800	2800
cellular	sf_cellular_qctlcatm1_socket	-	-	-	-	-	-	-	-	696	696
cellular	sf_cellular_qctlcatm1_socket_private	-	-	-	-	-	-	-	-	5165	5165
cgc	hw_cgc	-	2942	2942	3002	3002	3002	3002	3002	3002	3002
cgc	r_cgc	-	3410	3410	3170	3170	3170	3170	3170	3170	3170
console	sf_console	-	-	-	-	-	-	-	-	-	2283
ctsu	hw_ctsu_common	-	588	-	-	596	596	596	-	596	-
ctsu	sf_touch_ctsu	-	4312	-	-	4212	4212	4212	-	4212	-
ctsu	sf_touch_ctsu_button	-	1888	-	-	1964	1964	1964	-	1964	-
ctsu	sf_touch_ctsu_slider	-	-	-	-	2840	-	2840	-	-	-
ctsu	sf_touch_panel_i2c	-	-	-	-	-	-	-	-	1063	1063

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
ctsu	touch_panel_i2c_ft5x06	-	-	-	-	-	-	-	-	295	295
ctsu	touch_panel_i2c_sx8654	-	-	-	-	-	-	-	-	604	604
ctsu	r_ctsu	-	13889	-	-	13961	13961	13957	-	13957	-
dac	r_dac	-	644	644	592	592	592	592	592	592	592
dac8	r_dac8	-	-	816	-	724	-	-	-	-	-
dave/2d	dave_64bitoperation	-	-	-	-	-	-	-	-	0	0
dave/2d	dave_base	-	-	-	-	-	-	-	-	21	21
dave/2d	dave_blit	-	-	-	-	-	-	-	-	1272	1272
dave/2d	dave_box	-	-	-	-	-	-	-	-	3516	3516
dave/2d	dave_circle	-	-	-	-	-	-	-	-	1312	1312
dave/2d	dave_context	-	-	-	-	-	-	-	-	2992	2992
dave/2d	dave_curve	-	-	-	-	-	-	-	-	1104	1104
dave/2d	dave_dlist	-	-	-	-	-	-	-	-	3840	3840
dave/2d	dave_driver	-	-	-	-	-	-	-	-	2718	2718
dave/2d	dave_edge	-	-	-	-	-	-	-	-	1648	1648
dave/2d	dave_errorcodes	-	-	-	-	-	-	-	-	1184	1184
dave/2d	dave_gradient	-	-	-	-	-	-	-	-	2132	2132
dave/2d	dave_hardware	-	-	-	-	-	-	-	-	300	300
dave/2d	dave_line	-	-	-	-	-	-	-	-	8744	8744
dave/2d	dave_math	-	-	-	-	-	-	-	-	264	264
dave/2d	dave_memory	-	-	-	-	-	-	-	-	224	224
dave/2d	dave_pattern	-	-	-	-	-	-	-	-	464	464
dave/2d	dave_perfcount	-	-	-	-	-	-	-	-	200	200
dave/2d	dave_polyline	-	-	-	-	-	-	-	-	2116	2116
dave/2d	dave_quad	-	-	-	-	-	-	-	-	3260	3260
dave/2d	dave_rbuffer	-	-	-	-	-	-	-	-	1796	1796
dave/2d	dave_render	-	-	-	-	-	-	-	-	4304	4304
dave/2d	dave_texture	-	-	-	-	-	-	-	-	3700	3700
dave/2d	dave_triangle	-	-	-	-	-	-	-	-	2496	2496
dave/2d	dave_utility	-	-	-	-	-	-	-	-	1072	1072
dave/2d	dave_viewport	-	-	-	-	-	-	-	-	600	600
dave/2d	dave_wedge	-	-	-	-	-	-	-	-	1196	1196

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
dave/2d	sf_tes_2d_drw_base	-	-	-	-	-	-	-	-	275	275
dave/2d	sf_tes_2d_drw_irq	-	-	-	-	-	-	-	-	320	320
dave/2d	sf_tes_2d_drw_memory	-	-	-	-	-	-	-	-	229	229
dmac	r_dmac	-	-	-	1924	1924	1924	1936	1936	1936	1936
doc	r_doc	-	984	984	808	808	808	804	804	804	804
dtc	r_dtc	-	1821	1821	1669	1669	1669	1685	1685	1685	1685
el_fx	sf_el_fx	-	-	-	536	536	-	536	536	536	536
el_gx	sf_el_gx	-	-	-	-	-	-	-	-	1458	1458
el_nx	nx_hw_init	-	-	-	-	-	-	-	920	920	920
el_nx	nx_renesas_synergy	-	-	-	-	-	-	-	2376	2376	2376
el_nx	sf_el_nx_comms	-	-	-	-	-	-	-	1534	1534	1534
el_ux	sf_el_ux_comms	-	918	918	878	878	878	878	878	878	878
el_ux	usb_irq_veneer	-	28	28	32	32	32	32	32	96	96
el_ux	usbfs_irq_veneer	-	60	60	68	68	68	68	68	68	136
el_ux	ux_dcd_synergy_buffer_empty_interrupt	-	76	76	84	84	84	84	84	84	84
el_ux	ux_dcd_synergy_buffer_notready_interrupt	-	76	76	84	84	84	84	84	84	84
el_ux	ux_dcd_synergy_buffer_read	-	80	80	76	76	76	76	76	76	76
el_ux	ux_dcd_synergy_buffer_ready_interrupt	-	76	76	84	84	84	84	84	84	84
el_ux	ux_dcd_synergy_buffer_write	-	116	116	112	112	112	112	112	112	112
el_ux	ux_dcd_synergy_current_endpoint_change	-	140	140	140	140	140	140	140	140	140
el_ux	ux_dcd_synergy_data_buffersize	-	64	64	60	60	60	60	60	60	60
el_ux	ux_dcd_synergy_endpoint_create	-	424	424	424	424	424	424	424	424	424
el_ux	ux_dcd_synergy_endpoint_destroy	-	152	152	152	152	152	152	152	152	152
el_ux	ux_dcd_synergy_endpoint_nak_set	-	32	32	24	24	24	24	24	24	24
el_ux	ux_dcd_synergy_endpoint_reset	-	104	104	100	100	100	100	100	100	100
el_ux	ux_dcd_synergy_endpoint_stall	-	52	52	48	48	48	48	48	48	48
el_ux	ux_dcd_synergy_endpoint_status	-	28	28	20	20	20	20	20	20	20
el_ux	ux_dcd_synergy_fifo_port_change	-	60	60	60	60	60	60	60	60	60
el_ux	ux_dcd_synergy_fifo_read	-	800	800	672	672	672	672	672	1096	1096

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
el_ux	ux_dcd_synergy_fifoc_write	-	432	432	344	344	344	344	344	552	552
el_ux	ux_dcd_synergy_fifod_write	-	800	800	700	700	700	700	700	828	828
el_ux	ux_dcd_synergy_frame_number_get	-	20	20	20	20	20	20	20	20	20
el_ux	ux_dcd_synergy_function	-	136	136	84	84	84	84	84	84	84
el_ux	ux_dcd_synergy_initialize	-	846	846	750	750	750	750	750	1094	1094
el_ux	ux_dcd_synergy_initialize_complete	-	152	152	136	136	136	136	136	136	136
el_ux	ux_dcd_synergy_interrupt_handler	-	744	744	656	656	656	656	656	704	704
el_ux	ux_dcd_synergy_register_clear	-	16	16	16	16	16	16	16	16	16
el_ux	ux_dcd_synergy_register_read	-	12	12	8	8	8	8	8	8	8
el_ux	ux_dcd_synergy_register_set	-	16	16	12	12	12	12	12	12	12
el_ux	ux_dcd_synergy_register_write	-	12	12	8	8	8	8	8	8	8
el_ux	ux_dcd_synergy_transfer_callback	-	716	716	752	752	752	752	752	752	752
el_ux	ux_dcd_synergy_transfer_request	-	1068	1068	1148	1148	1148	1148	1148	1164	1164
el_ux	ux_hcd_synergy_asynch_queue_process	-	240	240	208	208	208	208	208	208	208
el_ux	ux_hcd_synergy_asynch_queue_process_bemp	-	324	324	316	316	316	316	316	316	316
el_ux	ux_hcd_synergy_asynch_queue_process_brdy	-	508	508	488	488	488	488	488	488	488
el_ux	ux_hcd_synergy_asynch_queue_process_nrdy	-	220	220	228	228	228	228	228	228	228
el_ux	ux_hcd_synergy_asynch_queue_process_sign	-	56	56	60	60	60	60	60	60	60
el_ux	ux_hcd_synergy_asynch_schedule	-	92	92	80	80	80	80	80	80	80
el_ux	ux_hcd_synergy_asynchronous_endpoint_create	-	68	68	60	60	60	60	60	60	60
el_ux	ux_hcd_synergy_asynchronous_endpoint_destroy	-	128	128	120	120	120	120	120	120	120
el_ux	ux_hcd_synergy_buffer_empty_interrupt	-	76	76	84	84	84	84	84	84	84
el_ux	ux_hcd_synergy_buffer_notready_interrupt	-	76	76	84	84	84	84	84	84	84
el_ux	ux_hcd_synergy_buffer_read	-	88	88	88	88	88	88	88	88	88

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
el_ux	ux_hcd_synergy_buffer_ready_interrupt	-	76	76	84	84	84	84	84	84	84
el_ux	ux_hcd_synergy_buffer_write	-	128	128	140	140	140	140	140	140	140
el_ux	ux_hcd_synergy_bulk_endpoint_create	-	368	368	380	380	380	380	380	500	500
el_ux	ux_hcd_synergy_bulk_int_td_add	-	228	228	232	232	232	232	232	232	232
el_ux	ux_hcd_synergy_control_endpoint_create	-	136	136	140	140	140	140	140	140	140
el_ux	ux_hcd_synergy_control_td_add	-	364	364	364	364	364	364	364	364	364
el_ux	ux_hcd_synergy_controller_disable	-	12	12	8	8	8	8	8	8	8
el_ux	ux_hcd_synergy_current_endpoint_change	-	116	116	124	124	124	124	124	124	124
el_ux	ux_hcd_synergy_data_buffer_size	-	44	44	40	40	40	40	40	40	40
el_ux	ux_hcd_synergy_ed_obtain	-	64	64	60	60	60	60	60	60	60
el_ux	ux_hcd_synergy_ed_td_clean	-	24	24	24	24	24	24	24	24	24
el_ux	ux_hcd_synergy_endpoint_nak_set	-	32	32	24	24	24	24	24	24	24
el_ux	ux_hcd_synergy_endpoint_reset	-	12	12	8	8	8	8	8	8	8
el_ux	ux_hcd_synergy_entry	-	400	400	380	380	380	380	380	380	380
el_ux	ux_hcd_synergy_fifo_port_change	-	72	72	64	64	64	64	64	64	64
el_ux	ux_hcd_synergy_fifo_read	-	732	732	624	624	624	624	624	712	712
el_ux	ux_hcd_synergy_fifoc_write	-	328	328	292	292	292	292	292	488	488
el_ux	ux_hcd_synergy_fifod_write	-	484	484	436	436	436	436	436	568	568
el_ux	ux_hcd_synergy_frame_number_get	-	88	88	92	92	92	92	92	92	92
el_ux	ux_hcd_synergy_frame_number_set	-	4	4	4	4	4	4	4	4	4
el_ux	ux_hcd_synergy_initialize	-	1280	1276	1168	1168	1168	1164	1164	1460	1460
el_ux	ux_hcd_synergy_interrupt_endpoint_create	-	392	392	392	392	392	392	392	392	392
el_ux	ux_hcd_synergy_interrupt_handler	-	1056	1056	968	968	968	968	968	1020	1020
el_ux	ux_hcd_synergy_iso_queue_process	-	180	180	152	152	152	152	152	152	152
el_ux	ux_hcd_synergy_iso_queue_process_bemp	-	212	212	220	220	220	220	220	220	220

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
el_ux	ux_hcd_synergy_iso_queue_process_brdy	-	276	276	264	264	264	264	264	264	264
el_ux	ux_hcd_synergy_iso_queue_process_nrdy	-	152	152	152	152	152	152	152	152	152
el_ux	ux_hcd_synergy_iso_schedule	-	92	92	80	80	80	80	80	80	80
el_ux	ux_hcd_synergy_iso_td_add	-	136	136	132	132	132	132	132	132	132
el_ux	ux_hcd_synergy_isochronous_endpoint_create	-	408	408	396	396	396	396	396	620	620
el_ux	ux_hcd_synergy_isochronous_td_obtain	-	64	64	60	60	60	60	60	60	60
el_ux	ux_hcd_synergy_least_traffic_list_get	-	68	68	68	68	68	68	68	68	68
el_ux	ux_hcd_synergy_periodic_endpoint_destroy	-	132	132	128	128	128	128	128	128	128
el_ux	ux_hcd_synergy_periodic_schedule	-	76	76	72	72	72	72	72	72	72
el_ux	ux_hcd_synergy_periodic_tree_create	-	164	164	144	144	144	144	144	144	144
el_ux	ux_hcd_synergy_port_disable	-	36	36	36	36	36	36	36	36	36
el_ux	ux_hcd_synergy_port_enable	-	68	68	68	68	68	68	68	68	68
el_ux	ux_hcd_synergy_port_reset	-	136	136	120	120	120	120	120	120	120
el_ux	ux_hcd_synergy_port_resume	-	4	4	4	4	4	4	4	4	4
el_ux	ux_hcd_synergy_port_status_get	-	88	88	88	88	88	88	88	88	88
el_ux	ux_hcd_synergy_port_suspend	-	4	4	4	4	4	4	4	4	4
el_ux	ux_hcd_synergy_power_down_port	-	4	4	4	4	4	4	4	4	4
el_ux	ux_hcd_synergy_power_on_port	-	4	4	4	4	4	4	4	4	4
el_ux	ux_hcd_synergy_power_root_hubs	-	4	4	4	4	4	4	4	4	4
el_ux	ux_hcd_synergy_register_clear	-	12	12	12	12	12	12	12	12	12
el_ux	ux_hcd_synergy_register_read	-	8	8	8	8	8	8	8	8	8
el_ux	ux_hcd_synergy_register_set	-	12	12	12	12	12	12	12	12	12
el_ux	ux_hcd_synergy_register_write	-	8	8	8	8	8	8	8	8	8
el_ux	ux_hcd_synergy_regular_td_obtain	-	96	96	96	96	96	96	96	96	96
el_ux	ux_hcd_synergy_request_bulk_transfer	-	224	224	216	216	216	216	216	340	340

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
el_ux	ux_hcd_synergy_request_control_transfer	-	528	528	508	508	508	508	508	508	508
el_ux	ux_hcd_synergy_request_interupt_transfer	-	76	76	72	72	72	72	72	72	72
el_ux	ux_hcd_synergy_request_isochronous_transfer	-	312	312	288	288	288	288	288	288	288
el_ux	ux_hcd_synergy_request_transfer	-	68	68	72	72	72	72	72	72	72
el_ux	ux_hcd_synergy_td_add	-	24	24	24	24	24	24	24	24	24
el_ux	ux_hcd_synergy_transfer_abort	-	244	244	244	244	244	244	244	244	244
elc	r_elc	-	336	336	312	312	312	312	312	312	312
external_irq	sf_external_irq	-	484	484	476	476	476	476	476	476	476
flash	hw_codeflash	-	1652	1652	1612	1612	1612	1612	-	-	-
flash	hw_codeflash_extra	-	444	444	408	408	408	408	-	-	-
flash	hw_dataflash	-	1132	1132	1048	1048	1048	1048	-	-	-
flash	hw_flash_common	-	24	24	28	28	28	28	-	-	-
flash	hw_flash_hp	-	-	-	-	-	-	-	3012	3012	2996
flash	hw_flash_lp	-	1116	1116	1180	1180	1180	1180	-	-	-
flash_hp	r_flash_hp	-	-	-	-	-	-	-	3106	3106	3106
flash_lp	r_flash_lp	-	3334	3334	3118	3118	3118	3118	-	-	-
fmi	r_fmi	-	1216	1216	1152	1152	1152	1156	1156	1156	1156
glcd	r_glcd	-	-	-	-	-	-	-	-	6431	6431
gpt	r_gpt	-	2744	2744	2368	2368	2368	2372	2416	2416	2416
gpt_input_capture	r_gpt_input_capture	-	1716	1716	1536	1536	1536	1536	1536	1536	1536
i2c	sf_i2c	-	-	-	-	1200	1200	1200	1200	-	1200
icu	r_icu	-	1004	1004	912	912	912	912	912	912	912
ioport	r_ioport	-	1654	1654	1780	1780	1780	1780	1780	1780	1780
iwdt	r_iwdt	-	611	611	591	591	591	591	591	591	591
jpeg	r_jpeg_common	-	-	-	-	-	-	-	-	96	96
jpeg	r_jpeg_decode	-	-	-	-	-	-	-	-	2364	2364
jpeg	r_jpeg_encode	-	-	-	-	-	-	-	-	1968	1968
jpeg	sf_jpeg_decode	-	-	-	-	-	-	-	-	1176	1176
kint	r_kint	-	989	989	853	853	853	853	853	853	853

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
lpm	hw_lpm_common	-	104	-	-	-	-	92	-	-	92
lpm	hw_lpm_s124	-	24	-	-	-	-	-	-	-	-
lpm	hw_lpm_s3a7	-	-	-	-	-	-	24	-	-	-
lpm	hw_lpm_s7g2	-	-	-	-	-	-	-	-	-	100
lpm	hw_lpmv2_s124	-	584	-	-	-	-	0	-	-	-
lpm	hw_lpmv2_s128	-	-	584	-	-	-	0	-	-	-
lpm	hw_lpmv2_s1ja	-	-	-	-	-	-	0	-	-	-
lpm	hw_lpmv2_s3a1	-	-	-	600	-	-	0	-	-	-
lpm	hw_lpmv2_s3a3	-	-	-	-	600	-	0	-	-	-
lpm	hw_lpmv2_s3a6	-	-	-	-	-	592	0	-	-	-
lpm	hw_lpmv2_s3a7	-	-	-	-	-	-	600	-	-	-
lpm	hw_lpmv2_s5d5	-	-	-	-	-	-	0	1140	-	-
lpm	hw_lpmv2_s5d9	-	-	-	-	-	-	0	-	1140	-
lpm	hw_lpmv2_s7g2	-	-	-	-	-	-	0	-	-	1140
lpm	r_lpm	-	860	-	-	-	-	848	-	-	1304
lpmv2	r_lpmv2	-	220	220	216	216	216	216	216	216	216
lvd	r_lvd	-	1968	1968	1676	1676	1676	1672	1672	1672	1672
message	sf_message	-	1404	1404	1320	1320	1320	1320	1320	1320	1320
pdc	r_pdc	-	-	-	-	-	-	-	1784	-	1784
power_profiles	sf_power_profiles	-	748	-	-	-	-	764	-	-	764
power_profiles	sf_power_profiles_v2	-	-	-	-	-	-	580	-	-	-
qspi	r_qspi	-	-	-	-	920	-	920	920	920	920
riic	r_riic	-	5112	5112	5164	5164	5164	5160	5160	5160	5160
riic_slave	r_riic_slave	-	2528	2528	2560	2560	2560	2560	2560	2560	2560
rspi	r_rspi	-	3960	3960	3732	3732	3732	3736	3736	3736	3736
rtc	r_rtc	-	4892	-	-	-	-	4412	-	-	4376
sci_i2c	r_sci_i2c	-	5081	5081	5009	5009	5009	5009	5009	5069	5069
sci_spi	r_sci_spi	-	3576	3576	3276	3276	3276	3268	3268	3268	3268
sci_uart	r_sci_uart	-	4566	4566	4302	4302	4302	4302	4302	4302	4302
sdmmc	r_sdmmc	-	-	-	5300	5300	-	5308	5308	5308	5308

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
sf_audio_record_i2s	sf_audio_record_i2s	-	-	-	576	576	576	576	576	576	576
slcdc	r_slcdc	-	-	-	1292	1292	1292	1292	-	-	-
spi	sf_spi	-	1456	1456	1448	1448	1448	1448	1448	1448	1448
spi	spi_hcd	-	-	-	-	-	-	-	-	-	2903
ssi	r_ssi	-	-	-	4745	4745	4745	4745	4745	4745	4745
system	startup_<mcu>	-	80	-	-	-	-	-	-	-	-
system	startup_S128	-	-	92	-	-	-	-	-	-	-
system	startup_S1JA	-	-	-	-	-	-	-	-	-	-
system	startup_S3A1	-	-	-	80	-	-	-	-	-	-
system	startup_S3A3	-	-	-	-	80	-	-	-	-	-
system	startup_S3A6	-	-	-	-	-	80	-	-	-	-
system	startup_S3A7	-	-	-	-	-	-	80	-	-	-
system	startup_S5D5	-	-	-	-	-	-	-	80	-	-
system	startup_S5D9	-	-	-	-	-	-	-	-	80	-
system	startup_S7G2	-	-	-	-	-	-	-	-	-	80
system	storerecall	-	-	-	-	-	-	-	-	-	0
system	system_S124	-	392	-	-	-	-	-	-	-	-
system	system_S128	-	-	352	-	-	-	-	-	-	-
system	system_S1JA	-	-	-	-	-	-	-	-	-	-
system	system_S3A1	-	-	-	396	-	-	-	-	-	-
system	system_S3A3	-	-	-	-	476	-	-	-	-	-
system	system_S3A6	-	-	-	-	-	976	-	-	-	-
system	system_S3A7	-	-	-	-	-	-	476	-	-	-
system	system_S5D5	-	-	-	-	-	-	-	1388	-	-
system	system_S5D9	-	-	-	-	-	-	-	-	1492	-
system	system_S7G2	-	-	-	-	-	-	-	-	-	1644
telnet	sf_comms_telnet	-	-	-	-	-	-	-	1486	1486	1486
thread_monitor	sf_thread_monitor	-	1139	1139	1063	1063	1063	1063	1063	1063	1063
uart	sf_uart_comms	-	1190	1190	1162	1162	1162	1162	1162	1166	1166
ul_gx	gx_display_driver_synergy_dave2d	-	-	-	-	-	-	-	-	14095	14095

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
ul_gx	gx_display_driver_synergy_dave2d_8bit_palette	-	-	-	-	-	-	-	-	5760	5760
wdt	r_wdt	-	767	767	747	747	747	747	747	747	747
wifi	api_init	-	-	-	-	-	-	-	-	-	493
wifi	api_ioctl	-	-	-	-	-	-	-	-	-	2952
wifi	api_stack_offload	-	-	-	-	-	-	-	-	-	12952
wifi	api_txrx	-	-	-	-	-	-	-	-	-	723
wifi	api_wmi_rx	-	-	-	-	-	-	-	-	-	2163
wifi	cust_api_init	-	-	-	-	-	-	-	-	-	470
wifi	cust_api_ioctl	-	-	-	-	-	-	-	-	-	5572
wifi	cust_api_stack_offload	-	-	-	-	-	-	-	-	-	4534
wifi	cust_api_stack_txrx	-	-	-	-	-	-	-	-	-	268
wifi	cust_api_txrx	-	-	-	-	-	-	-	-	-	244
wifi	cust_api_wmi_rx	-	-	-	-	-	-	-	-	-	1650
wifi	cust_driver_main	-	-	-	-	-	-	-	-	-	320
wifi	cust_driver_netbuf	-	-	-	-	-	-	-	-	-	1165
wifi	cust_spi_hcd	-	-	-	-	-	-	-	-	-	758
wifi	custom_qcom_api	-	-	-	-	-	-	-	-	-	480
wifi	driver_diag	-	-	-	-	-	-	-	-	-	594
wifi	driver_init	-	-	-	-	-	-	-	-	-	1321
wifi	driver_main	-	-	-	-	-	-	-	-	-	1318
wifi	driver_netbuf	-	-	-	-	-	-	-	-	-	84
wifi	driver_txrx	-	-	-	-	-	-	-	-	-	538
wifi	Dset	-	-	-	-	-	-	-	-	-	380
wifi	dset_api	-	-	-	-	-	-	-	-	-	592
wifi	enet_irq_veneer	-	-	-	-	-	-	-	56	56	56
wifi	ether_phy	-	-	-	-	-	-	-	876	876	904
wifi	gt202_debug	-	-	-	-	-	-	-	-	-	807
wifi	gt202_util	-	-	-	-	-	-	-	-	-	386
wifi	gt202_wifi_ctrl	-	-	-	-	-	-	-	-	-	1751
wifi	gt202_wifi_task_thread	-	-	-	-	-	-	-	-	-	1328
wifi	Htc	-	-	-	-	-	-	-	-	-	1617

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
wifi	hw_api	-	-	-	-	-	-	-	-	-	334
wifi	Osal	-	-	-	-	-	-	-	-	-	840
wifi	qcom_api	-	-	-	-	-	-	-	-	-	7440
wifi	qcom_legacy	-	-	-	-	-	-	-	-	-	56
wifi	rcv_aggr	-	-	-	-	-	-	-	-	-	359
wifi	ring_buffer	-	-	-	-	-	-	-	-	-	320
wifi	sf_wifi_gt202	-	-	-	-	-	-	-	-	-	3153
wifi	sf_wifi_gt202_onchip_stack	-	-	-	-	-	-	-	-	-	376
wifi	sf_wifi_gt202_socket	-	-	-	-	-	-	-	-	-	1127
wifi	sf_wifi_nsal_nx	-	-	-	-	-	-	-	-	-	1468
wifi	Util	-	-	-	-	-	-	-	-	-	196
wifi	Wmi	-	-	-	-	-	-	-	-	-	3164

16.2 IAR ROM Memory Size Estimation

Table 16-2 IAR ROM results compiled for various MCU Groups:

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
_bsp	bsp_cache	92	8	8	96	96	96	96	88	96	96
_bsp	bsp_clocks	212	80	80	280	288	284	280	357	373	424
_bsp	bsp_common	82	86	86	86	86	86	86	86	86	86
_bsp	bsp_common_leds	20	24	24	24	24	24	24	24	24	24
_bsp	bsp_delay	94	102	102	94	94	94	94	94	94	94
_bsp	bsp_feature	244	228	252	280	280	280	284	244	244	244
_bsp	bsp_fmi_R7FS124*	-	1024	-	-	-	-	-	-	-	-
_bsp	bsp_fmi_R7FS128*	-	-	1024	-	-	-	-	-	-	-
_bsp	bsp_fmi_R7FS1JA*	1024	-	-	-	-	-	-	-	-	-
_bsp	bsp_fmi_R7FS3A1*	-	-	-	1024	-	-	-	-	-	-
_bsp	bsp_fmi_R7FS3A3*	-	-	-	-	1024	-	-	-	-	-
_bsp	bsp_fmi_R7FS3A6*	-	-	-	-	-	1024	-	-	-	-
_bsp	bsp_fmi_R7FS3A7*	-	-	-	-	-	-	1024	-	-	-
_bsp	bsp_fmi_R7FS5D5*	-	-	-	-	-	-	-	1024	-	-
_bsp	bsp_fmi_R7FS5D9*	-	-	-	-	-	-	-	-	1024	-
_bsp	bsp_fmi_R7FS7G2*	-	-	-	-	-	-	-	-	-	1024
_bsp	bsp_group_irq	364	300	320	384	384	384	384	384	384	384
_bsp	bsp_hw_locks	300	275	290	385	430	420	420	540	540	540
_bsp	bsp_init	2	2	2	4	4	2	4	14	14	20
_bsp	bsp_irq	92	92	92	88	88	88	88	88	88	88
_bsp	bsp_leds	16	16	16	16	16	16	16	12	16	16
_bsp	bsp_locking	236	242	242	232	232	232	232	232	232	232
_bsp	bsp_module_stop	256	252	244	240	276	276	240	248	248	248
_bsp	bsp_qspi	-	-	-	940	940	-	940	940	940	952
_bsp	bsp_register_protection	152	156	156	160	160	160	160	160	160	160
_bsp	bsp_rom_registers	76	24	76	76	76	76	76	76	76	76
_bsp	bsp_sbrk	0	0	0	0	0	0	0	0	0	0
_bsp	bsp_sdram	-	-	-	-	-	-	-	236	236	236
adc	r_adc	3896	3936	3936	3856	3856	3856	3852	3852	3852	3852
adc_periodic	sf_adc_periodic	956	1004	1004	968	968	968	968	968	968	968

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
agt	r_agt	2030	2094	2094	1922	1922	1922	1918	1918	1918	1918
analog	r_acmphs	964	-	-	-	-	-	-	-	-	-
analog	r_acmplp	1188	-	-	-	-	-	-	-	-	-
analog	r_opamp	812	-	840	-	-	-	-	-	-	-
analog	r_sdadc	2692	-	-	-	-	-	-	-	-	-
audio_playback	sf_audio_playback	2736	2900	2900	2766	2766	2766	2766	2766	2766	2766
audio_playback_dac	sf_audio_playback_hw_dac	700	676	748	728	728	728	728	728	728	728
audio_playback_i2s	sf_audio_playback_hw_i2s	-	-	-	236	236	236	236	236	236	236
audio_record_adc	sf_audio_record_adc	-	376	-	-	360	360	360	360	360	360
ble	ble_rl78g1d_if	-	-	-	-	-	-	-	-	-	8269
ble	ble_rl78g1d_prf_anp_if	-	-	-	-	-	-	-	-	-	878
ble	ble_rl78g1d_prf_blp_if	-	-	-	-	-	-	-	-	-	1060
ble	ble_rl78g1d_prf_fmp_if	-	-	-	-	-	-	-	-	-	492
ble	ble_rl78g1d_prf_hid_if	-	-	-	-	-	-	-	-	-	1660
ble	ble_rl78g1d_prf_hrp_if	-	-	-	-	-	-	-	-	-	984
ble	ble_rl78g1d_prf_htp_if	-	-	-	-	-	-	-	-	-	1040
ble	ble_rl78g1d_prf_if	-	-	-	-	-	-	-	-	-	1732
ble	ble_rl78g1d_prf_paps_if	-	-	-	-	-	-	-	-	-	962
ble	ble_rl78g1d_prf_pxp_if	-	-	-	-	-	-	-	-	-	768
ble	ble_rl78g1d_prf_scps_if	-	-	-	-	-	-	-	-	-	528
ble	ble_rl78g1d_prf_tip_if	-	-	-	-	-	-	-	-	-	1080
ble	ble_serial	-	-	-	-	-	-	-	-	-	2328
ble	bmi	-	-	-	-	-	-	-	-	-	1408
ble	rble_api_anpc	-	-	-	-	-	-	-	-	-	560
ble	rble_api_anps	-	-	-	-	-	-	-	-	-	312
ble	rble_api_blpc	-	-	-	-	-	-	-	-	-	752
ble	rble_api_blps	-	-	-	-	-	-	-	-	-	360
ble	rble_api_cppc	-	-	-	-	-	-	-	-	-	1012
ble	rble_api_cpps	-	-	-	-	-	-	-	-	-	900
ble	rble_api_cscpc	-	-	-	-	-	-	-	-	-	822
ble	rble_api_cscps	-	-	-	-	-	-	-	-	-	348
ble	rble_api_fmpl	-	-	-	-	-	-	-	-	-	256

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
ble	rble_api_fmpt	-	-	-	-	-	-	-	-	-	132
ble	rble_api_gap	-	-	-	-	-	-	-	-	-	1940
ble	rble_api_gatt	-	-	-	-	-	-	-	-	-	1208
ble	rble_api_glpc	-	-	-	-	-	-	-	-	-	992
ble	rble_api_glps	-	-	-	-	-	-	-	-	-	612
ble	rble_api_hgbh	-	-	-	-	-	-	-	-	-	1400
ble	rble_api_hghd	-	-	-	-	-	-	-	-	-	460
ble	rble_api_hgrh	-	-	-	-	-	-	-	-	-	1552
ble	rble_api_hrpc	-	-	-	-	-	-	-	-	-	692
ble	rble_api_hrps	-	-	-	-	-	-	-	-	-	262
ble	rble_api_htpc	-	-	-	-	-	-	-	-	-	876
ble	rble_api_htpt	-	-	-	-	-	-	-	-	-	380
ble	rble_api_lnpc	-	-	-	-	-	-	-	-	-	1076
ble	rble_api_lnps	-	-	-	-	-	-	-	-	-	904
ble	rble_api_paspc	-	-	-	-	-	-	-	-	-	472
ble	rble_api_pasps	-	-	-	-	-	-	-	-	-	276
ble	rble_api_pxpm	-	-	-	-	-	-	-	-	-	488
ble	rble_api_pxpr	-	-	-	-	-	-	-	-	-	136
ble	rble_api_rscpc	-	-	-	-	-	-	-	-	-	896
ble	rble_api_rscps	-	-	-	-	-	-	-	-	-	400
ble	rble_api_sm	-	-	-	-	-	-	-	-	-	536
ble	rble_api_sppc	-	-	-	-	-	-	-	-	-	412
ble	rble_api_spps	-	-	-	-	-	-	-	-	-	212
ble	rble_api_tipc	-	-	-	-	-	-	-	-	-	632
ble	rble_api_tips	-	-	-	-	-	-	-	-	-	532
ble	rble_api_vs	-	-	-	-	-	-	-	-	-	940
ble	rble_host	-	-	-	-	-	-	-	-	-	3340
ble	rble_if_api_cb	-	-	-	-	-	-	-	-	-	25072
ble	rscip	-	-	-	-	-	-	-	-	-	252
ble	rscip_cntl	-	-	-	-	-	-	-	-	-	2898
ble	rscip_uart	-	-	-	-	-	-	-	-	-	2080
ble	sf_ble_rl78g1d	-	-	-	-	-	-	-	-	-	2536

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
ble	sf_ble_rl78g1d_onboard_profile	-	-	-	-	-	-	-	-	-	872
ble	timer	-	-	-	-	-	-	-	-	-	321
block_media_sdmmc	sf_block_media_qspi	-	-	-	-	848	-	848	848	848	848
block_media_sdmmc	sf_block_media_ram	-	-	-	-	352	-	352	-	352	352
block_media_sdmmc	sf_block_media_sdmmc	-	-	-	492	492	-	492	492	492	492
cac	r_cac	1820	1804	1804	1864	1864	1864	1860	1860	1860	1860
can	hw_can	1468	1480	1480	1292	1292	1292	1292	1292	1292	1292
can	r_can	2378	2454	2454	2320	2320	2320	2320	2320	2320	2320
cellular	cellular_serial	-	-	-	-	-	712	712	712	712	712
cellular	sf_cellular_cat1	-	-	-	-	-	5568	5568	5568	5568	5568
cellular	sf_cellular_cat1_socket	-	-	-	-	-	548	548	548	548	548
cellular	sf_cellular_cat1_socket_private	-	-	-	-	-	6552	6552	6552	6552	6552
cellular	sf_cellular_cat3	-	-	-	-	-	1744	1744	1744	1744	1744
cellular	sf_cellular_cat3_socket	-	-	-	-	-	524	524	524	524	524
cellular	sf_cellular_cat3_socket_private	-	-	-	-	-	6328	6328	6328	6328	6328
cellular	sf_cellular_common	-	-	-	-	-	1400	1400	1400	1400	1400
cellular	sf_cellular_common_private	-	-	-	-	-	10016	10016	10016	10016	10016
cellular	sf_cellular_qctlcatm1	-	-	-	-	-	-	-	-	-	-
cellular	sf_cellular_qctlcatm1_socket	-	-	-	-	-	-	-	-	-	-
cellular	sf_cellular_qctlcatm1_socket_private	-	-	-	-	-	-	-	-	-	-
cgc	hw_cgc	2490	2538	2538	2496	2496	2496	2496	2496	2496	2496
cgc	r_cgc	2924	3044	3044	2984	2984	2984	2984	2984	2984	2984
console	sf_console	-	-	-	-	-	-	-	-	-	2272
ctsu	hw_ctsu_common	-	432	-	-	392	392	392	-	392	-
ctsu	sf_touch_ctsu	-	4256	-	-	4208	4208	4208	-	4208	-
ctsu	sf_touch_ctsu_button	-	2056	-	-	1976	1976	1976	-	1976	-
ctsu	sf_touch_ctsu_slider	-	-	-	-	2648	-	2648	-	-	-
ctsu	sf_touch_panel_i2c	-	-	-	-	-	-	-	-	1065	1065
ctsu	touch_panel_i2c_ft5x06	-	-	-	-	-	-	-	-	280	280
ctsu	touch_panel_i2c_sx8654	-	-	-	-	-	-	-	-	572	572

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
ctsu	r_ctsu	-	12508	-	-	11820	11820	11812	-	11812	-
dac	r_dac	588	624	624	596	596	596	596	596	596	596
dac8	r_dac8	776	-	804	-	732	-	-	-	-	-
dave/2d	dave_64bitoperation	-	-	-	-	-	-	-	-	0	0
dave/2d	dave_base	-	-	-	-	-	-	-	-	24	24
dave/2d	dave_blit	-	-	-	-	-	-	-	-	1024	1024
dave/2d	dave_box	-	-	-	-	-	-	-	-	1972	1972
dave/2d	dave_circle	-	-	-	-	-	-	-	-	1140	1140
dave/2d	dave_context	-	-	-	-	-	-	-	-	2392	2392
dave/2d	dave_curve	-	-	-	-	-	-	-	-	612	612
dave/2d	dave_dlist	-	-	-	-	-	-	-	-	3396	3396
dave/2d	dave_driver	-	-	-	-	-	-	-	-	2780	2780
dave/2d	dave_edge	-	-	-	-	-	-	-	-	1416	1416
dave/2d	dave_errorcodes	-	-	-	-	-	-	-	-	1180	1180
dave/2d	dave_gradient	-	-	-	-	-	-	-	-	498	498
dave/2d	dave_hardware	-	-	-	-	-	-	-	-	276	276
dave/2d	dave_line	-	-	-	-	-	-	-	-	6368	6368
dave/2d	dave_math	-	-	-	-	-	-	-	-	256	256
dave/2d	dave_memory	-	-	-	-	-	-	-	-	218	218
dave/2d	dave_pattern	-	-	-	-	-	-	-	-	320	320
dave/2d	dave_perfcount	-	-	-	-	-	-	-	-	168	168
dave/2d	dave_polyline	-	-	-	-	-	-	-	-	1716	1716
dave/2d	dave_quad	-	-	-	-	-	-	-	-	2754	2754
dave/2d	dave_rbuffer	-	-	-	-	-	-	-	-	1660	1660
dave/2d	dave_render	-	-	-	-	-	-	-	-	2978	2978
dave/2d	dave_texture	-	-	-	-	-	-	-	-	2612	2612
dave/2d	dave_triangle	-	-	-	-	-	-	-	-	1942	1942
dave/2d	dave_utility	-	-	-	-	-	-	-	-	1098	1098
dave/2d	dave_viewport	-	-	-	-	-	-	-	-	568	568
dave/2d	dave_wedge	-	-	-	-	-	-	-	-	1050	1050
dave/2d	sf_tes_2d_drw_base	-	-	-	-	-	-	-	-	300	300
dave/2d	sf_tes_2d_drw_irq	-	-	-	-	-	-	-	-	336	336

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
dave/2d	sf_tes_2d_drw_memory	-	-	-	-	-	-	-	-	208	208
dmac	r_dmac	-	-	-	1760	1760	1760	1764	1764	1764	1764
doc	r_doc	832	872	872	800	800	800	796	796	796	796
dtc	r_dtc	1860	1888	1888	1960	1960	1960	1980	2108	2108	2108
el_fx	sf_el_fx	-	-	-	464	464	-	464	464	464	464
el_gx	sf_el_gx	-	-	-	-	-	-	-	-	1348	1348
el_nx	nx_hw_init	-	-	-	-	-	-	-	908	908	908
el_nx	nx_renesas_synergy	-	-	-	-	-	-	-	2224	2224	2224
el_nx	sf_el_nx_comms	-	-	-	-	-	-	-	1508	1508	1508
el_ux	sf_el_ux_comms	748	776	776	756	756	756	756	756	756	756
el_ux	usb_irq_veneer	12	16	16	12	12	12	12	12	72	72
el_ux	usbfs_irq_veneer	32	36	36	34	34	34	34	34	34	84
el_ux	ux_dcd_synergy_buffer_empty_interrupt	76	76	76	78	78	78	78	78	78	78
el_ux	ux_dcd_synergy_buffer_notready_interrupt	76	76	76	78	78	78	78	78	78	78
el_ux	ux_dcd_synergy_buffer_read	68	68	68	66	66	66	66	66	66	66
el_ux	ux_dcd_synergy_buffer_ready_interrupt	76	76	76	78	78	78	78	78	78	78
el_ux	ux_dcd_synergy_buffer_write	114	118	118	114	114	114	114	114	114	114
el_ux	ux_dcd_synergy_current_endpoint_change	166	172	172	166	166	166	166	166	166	166
el_ux	ux_dcd_synergy_data_buffersize	66	68	68	64	64	64	64	64	64	64
el_ux	ux_dcd_synergy_endpoint_create	386	386	386	448	448	448	448	448	448	448
el_ux	ux_dcd_synergy_endpoint_destroy	156	156	156	154	154	154	154	154	154	154
el_ux	ux_dcd_synergy_endpoint_nak_set	20	26	26	20	20	20	20	20	20	20
el_ux	ux_dcd_synergy_endpoint_reset	102	106	106	100	100	100	100	100	100	100
el_ux	ux_dcd_synergy_endpoint_stall	42	46	46	42	42	42	42	42	42	42
el_ux	ux_dcd_synergy_endpoint_status	32	32	32	32	32	32	32	32	32	32
el_ux	ux_dcd_synergy_fifo_port_change	54	56	56	52	52	52	52	52	52	52
el_ux	ux_dcd_synergy_fifo_read	862	890	890	858	858	858	858	858	1094	1094
el_ux	ux_dcd_synergy_fifoc_write	338	350	350	280	280	280	280	280	412	412
el_ux	ux_dcd_synergy_fifod_write	702	730	730	698	698	698	698	698	804	804
el_ux	ux_dcd_synergy_frame_number_get	20	20	20	20	20	20	20	20	20	20
el_ux	ux_dcd_synergy_function	76	94	94	82	82	82	82	82	82	82
el_ux	ux_dcd_synergy_initialize	804	808	808	752	752	752	756	756	1104	1104

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
el_ux	ux_dcd_synergy_initialize_complete	132	132	132	124	124	124	124	124	124	124
el_ux	ux_dcd_synergy_interrupt_handler	660	676	676	624	624	624	624	624	668	668
el_ux	ux_dcd_synergy_register_clear	16	16	16	16	16	16	16	16	16	16
el_ux	ux_dcd_synergy_register_read	12	12	12	10	10	10	10	10	10	10
el_ux	ux_dcd_synergy_register_set	16	16	16	14	14	14	14	14	14	14
el_ux	ux_dcd_synergy_register_write	12	12	12	10	10	10	10	10	10	10
el_ux	ux_dcd_synergy_transfer_callback	636	656	656	654	654	654	654	654	654	654
el_ux	ux_dcd_synergy_transfer_request	922	900	900	808	808	808	808	808	832	832
el_ux	ux_hcd_synergy_asynch_queue_process	196	200	200	216	216	216	216	216	216	216
el_ux	ux_hcd_synergy_asynch_queue_process_bemp	290	300	300	290	290	290	290	290	290	290
el_ux	ux_hcd_synergy_asynch_queue_process_brdy	438	442	442	428	428	428	428	428	428	428
el_ux	ux_hcd_synergy_asynch_queue_process_nrdy	194	200	200	194	194	194	194	194	194	194
el_ux	ux_hcd_synergy_asynch_queue_process_sign	54	56	56	56	56	56	56	56	56	56
el_ux	ux_hcd_synergy_asynch_schedule	66	78	78	70	70	70	70	70	70	70
el_ux	ux_hcd_synergy_asynchronous_endpoint_create	62	66	66	62	62	62	62	62	62	62
el_ux	ux_hcd_synergy_asynchronous_endpoint_destroy	116	122	122	114	114	114	114	114	114	114
el_ux	ux_hcd_synergy_buffer_empty_interrupt	76	76	76	78	78	78	78	78	78	78
el_ux	ux_hcd_synergy_buffer_notready_interrupt	76	76	76	78	78	78	78	78	78	78
el_ux	ux_hcd_synergy_buffer_read	78	78	78	76	76	76	76	76	76	76
el_ux	ux_hcd_synergy_buffer_ready_interrupt	76	76	76	78	78	78	78	78	78	78
el_ux	ux_hcd_synergy_buffer_write	112	114	114	110	110	110	110	110	110	110
el_ux	ux_hcd_synergy_bulk_endpoint_create	414	430	430	414	414	414	414	414	528	528
el_ux	ux_hcd_synergy_bulk_int_td_add	218	222	222	212	212	212	212	212	212	212
el_ux	ux_hcd_synergy_control_endpoint_create	146	152	152	138	138	138	138	138	138	138
el_ux	ux_hcd_synergy_control_td_add	334	338	338	330	330	330	330	330	330	330
el_ux	ux_hcd_synergy_controller_disable	10	10	10	10	10	10	10	10	10	10
el_ux	ux_hcd_synergy_current_endpoint_change	110	116	116	106	106	106	106	106	106	106
el_ux	ux_hcd_synergy_data_buffer_size	36	38	38	44	44	44	44	44	44	44
el_ux	ux_hcd_synergy_ed_obtain	56	56	56	56	56	56	56	56	56	56
el_ux	ux_hcd_synergy_ed_td_clean	20	20	20	20	20	20	20	20	20	20
el_ux	ux_hcd_synergy_endpoint_nak_set	20	26	26	20	20	20	20	20	20	20
el_ux	ux_hcd_synergy_endpoint_reset	8	8	8	8	8	8	8	8	8	8

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
el_ux	ux_hcd_synergy_entry	328	330	330	356	356	356	356	356	356	356
el_ux	ux_hcd_synergy_fifo_port_change	58	60	60	58	58	58	58	58	58	58
el_ux	ux_hcd_synergy_fifo_read	604	620	620	628	628	628	628	628	714	714
el_ux	ux_hcd_synergy_fifoc_write	258	270	270	252	252	252	252	252	416	416
el_ux	ux_hcd_synergy_fifod_write	460	472	472	456	456	456	456	456	580	580
el_ux	ux_hcd_synergy_frame_number_get	76	76	76	84	84	84	84	84	84	84
el_ux	ux_hcd_synergy_frame_number_set	2	2	2	2	2	2	2	2	2	2
el_ux	ux_hcd_synergy_initialize	1160	1172	1176	1124	1124	1124	1124	1124	1392	1392
el_ux	ux_hcd_synergy_interrupt_endpoint_create	412	436	436	406	406	406	406	406	406	406
el_ux	ux_hcd_synergy_interrupt_handler	940	964	964	904	904	904	904	904	944	944
el_ux	ux_hcd_synergy_iso_queue_process	136	140	140	126	126	126	126	126	126	126
el_ux	ux_hcd_synergy_iso_queue_process_bemp	202	206	206	204	204	204	204	204	204	204
el_ux	ux_hcd_synergy_iso_queue_process_brdy	258	270	270	258	258	258	258	258	258	258
el_ux	ux_hcd_synergy_iso_queue_process_nrdy	144	146	146	144	144	144	144	144	144	144
el_ux	ux_hcd_synergy_iso_schedule	66	78	78	70	70	70	70	70	70	70
el_ux	ux_hcd_synergy_iso_td_add	138	138	138	132	132	132	132	132	132	132
el_ux	ux_hcd_synergy_isochronous_endpoint_create	402	412	412	400	400	400	400	400	536	536
el_ux	ux_hcd_synergy_isochronous_td_obtain	56	56	56	56	56	56	56	56	56	56
el_ux	ux_hcd_synergy_least_traffic_list_get	62	62	62	66	66	66	66	66	66	66
el_ux	ux_hcd_synergy_periodic_endpoint_destroy	126	132	132	128	128	128	128	128	128	128
el_ux	ux_hcd_synergy_periodic_schedule	66	68	68	60	60	60	60	60	60	60
el_ux	ux_hcd_synergy_periodic_tree_create	292	296	296	292	292	292	292	292	292	292
el_ux	ux_hcd_synergy_port_disable	34	34	34	34	34	34	34	34	34	34
el_ux	ux_hcd_synergy_port_enable	70	70	70	70	70	70	70	70	70	70
el_ux	ux_hcd_synergy_port_reset	132	132	132	120	120	120	120	120	120	120
el_ux	ux_hcd_synergy_port_resume	4	4	4	4	4	4	4	4	4	4
el_ux	ux_hcd_synergy_port_status_get	90	90	90	82	82	82	82	82	82	82
el_ux	ux_hcd_synergy_port_suspend	4	4	4	4	4	4	4	4	4	4
el_ux	ux_hcd_synergy_power_down_port	4	4	4	4	4	4	4	4	4	4
el_ux	ux_hcd_synergy_power_on_port	4	4	4	4	4	4	4	4	4	4
el_ux	ux_hcd_synergy_power_root_hubs	2	2	2	2	2	2	2	2	2	2
el_ux	ux_hcd_synergy_register_clear	12	12	12	14	14	14	14	14	14	14

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
el_ux	ux_hcd_synergy_register_read	8	8	8	8	8	8	8	8	8	8
el_ux	ux_hcd_synergy_register_set	12	12	12	12	12	12	12	12	12	12
el_ux	ux_hcd_synergy_register_write	8	8	8	8	8	8	8	8	8	8
el_ux	ux_hcd_synergy_regular_td_obtain	88	88	88	92	92	92	92	92	92	92
el_ux	ux_hcd_synergy_request_bulk_transfer	194	200	200	176	176	176	176	176	284	284
el_ux	ux_hcd_synergy_request_control_transfer	448	460	460	438	438	438	438	438	438	438
el_ux	ux_hcd_synergy_request_interupt_transfer	72	74	74	68	68	68	68	68	68	68
el_ux	ux_hcd_synergy_request_isochronous_transfer	272	288	288	242	242	242	242	242	242	242
el_ux	ux_hcd_synergy_request_transfer	52	62	62	52	52	52	52	52	52	52
el_ux	ux_hcd_synergy_td_add	22	24	24	22	22	22	22	22	22	22
el_ux	ux_hcd_synergy_transfer_abort	236	240	240	230	230	230	230	230	230	230
elc	r_elc	288	296	296	304	304	304	304	304	304	304
external_irq	sf_external_irq	400	436	436	420	420	420	420	420	420	420
flash	hw_codeflash	1344	1360	1360	1348	1348	1348	1348	-	-	-
flash	hw_codeflash_extra	376	378	378	320	320	320	320	-	-	-
flash	hw_dataflash	888	900	900	876	876	876	876	-	-	-
flash	hw_flash_common	20	26	26	22	22	22	22	-	-	-
flash	hw_flash_hp	-	-	-	-	-	-	-	2726	2726	2666
flash	hw_flash_lp	1120	1128	1128	1124	1124	1124	1124	-	-	-
flash_hp	r_flash_hp	-	-	-	-	-	-	-	2834	2834	2834
flash_lp	r_flash_lp	2822	2994	2994	2846	2846	2846	2846	-	-	-
fmi	r_fmi	1476	1496	1496	1444	1444	1444	1448	1448	1448	1448
glcd	r_glcd	-	-	-	-	-	-	-	-	4916	4916
gpt	r_gpt	2292	2368	2368	2108	2108	2108	2112	2164	2164	2164
gpt_input_capture	r_gpt_input_capture	1508	1520	1520	1520	1520	1520	1520	1520	1520	1520
i2c	sf_i2c	-	-	-	-	1120	1120	1120	1120	-	1120
icu	r_icu	952	944	944	896	896	896	892	892	892	892
ioport	r_ioport	1234	1270	1270	1346	1346	1346	1346	1346	1346	1346
iwdt	r_iwdt	528	564	564	568	568	568	568	568	568	568
jpeg	r_jpeg_common	-	-	-	-	-	-	-	-	52	52
jpeg	r_jpeg_decode	-	-	-	-	-	-	-	-	2256	2256
jpeg	r_jpeg_encode	-	-	-	-	-	-	-	-	2040	2040

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
jpeg	sf_jpeg_decode	-	-	-	-	-	-	-	-	984	984
kint	r_kint	872	864	864	864	864	864	860	860	860	860
lpm	hw_lpm_common	-	100	-	-	-	-	96	-	-	96
lpm	hw_lpm_s124	-	44	-	-	-	-	-	-	-	-
lpm	hw_lpm_s3a7	-	-	-	-	-	-	42	-	-	-
lpm	hw_lpm_s7g2	-	-	-	-	-	-	-	-	-	88
lpm	hw_lpmv2_s124	0	528	-	-	-	-	0	-	-	-
lpm	hw_lpmv2_s128	-	-	528	-	-	-	0	-	-	-
lpm	hw_lpmv2_s1ja	512	-	-	-	-	-	0	-	-	-
lpm	hw_lpmv2_s3a1	-	-	-	528	-	-	0	-	-	-
lpm	hw_lpmv2_s3a3	-	-	-	-	528	-	0	-	-	-
lpm	hw_lpmv2_s3a6	-	-	-	-	-	512	0	-	-	-
lpm	hw_lpmv2_s3a7	-	-	-	-	-	-	528	-	-	-
lpm	hw_lpmv2_s5d5	-	-	-	-	-	-	0	936	-	-
lpm	hw_lpmv2_s5d9	-	-	-	-	-	-	0	-	936	-
lpm	hw_lpmv2_s7g2	-	-	-	-	-	-	0	-	-	936
lpm	r_lpm	-	708	-	-	-	-	712	-	-	1024
lpmv2	r_lpmv2	220	232	232	220	220	220	220	220	220	220
lvd	r_lvd	1768	1772	1772	1660	1660	1660	1656	1656	1656	1656
message	sf_message	1244	1304	1304	1248	1248	1248	1248	1248	1248	1248
pdc	r_pdc	-	-	-	-	-	-	-	1680	-	1680
power_profiles	sf_power_profiles	-	648	-	-	-	-	636	-	-	636
power_profiles	sf_power_profiles_v2	-	-	-	-	-	-	512	-	-	-
qspi	r_qspi	-	-	-	-	980	-	980	980	980	980
riic	r_riic	4608	4868	4868	4712	4712	4712	4704	4704	4704	4704
riic_slave	r_riic_slave	2206	2306	2306	2228	2228	2228	2224	2224	2224	2224
rspi	r_rspi	3484	3776	3776	3480	3480	3480	3480	3480	3480	3480
rtc	r_rtc	-	4064	-	-	-	-	3676	-	-	3676
sci_i2c	r_sci_i2c	4152	4260	4260	4320	4320	4320	4312	4312	4312	4312
sci_spi	r_sci_spi	2912	3052	3052	2964	2964	2964	2960	2960	2960	2960
sci_uart	r_sci_uart	3892	4144	4144	3916	3916	3916	3912	3912	3912	3912
sdmmc	r_sdmmc	-	-	-	6280	6280	-	6284	6284	6284	6284

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
sf_audio_record_i2s	sf_audio_record_i2s	-	-	-	540	540	540	540	540	540	540
slcdc	r_slcdc	-	-	-	1336	1336	1336	1336	-	-	-
spi	sf_spi	-	-	-	-	-	-	-	-	-	-
spi	spi_hcd	-	-	-	-	-	-	-	-	-	2572
ssi	r_ssi	-	-	-	3980	3980	3980	3976	3976	3976	3976
system	startup_S124	-	80	-	-	-	-	-	-	-	-
system	startup_S128	-	-	5212	-	-	-	-	-	-	-
system	startup_S1JA	92	-	-	-	-	-	-	-	-	-
system	startup_S3A1	-	-	-	9296	-	-	-	-	-	-
system	startup_S3A3	-	-	-	-	9296	-	-	-	-	-
system	startup_S3A6	-	-	-	-	-	80	-	-	-	-
system	startup_S3A7	-	-	-	-	-	-	9296	-	-	-
system	startup_S5D5	-	-	-	-	-	-	-	80	-	-
system	startup_S5D9	-	-	-	-	-	-	-	-	80	-
system	startup_S7G2	-	-	-	-	-	-	-	-	-	80
system	storerecall	-	-	-	-	-	-	-	-	-	0
system	system_S124	-	414	-	-	-	-	-	-	-	-
system	system_S128	-	-	366	-	-	-	-	-	-	-
system	system_S1JA	342	-	-	-	-	-	-	-	-	-
system	system_S3A1	-	-	-	406	-	-	-	-	-	-
system	system_S3A3	-	-	-	-	486	-	-	-	-	-
system	system_S3A6	-	-	-	-	-	990	-	-	-	-
system	system_S3A7	-	-	-	-	-	-	486	-	-	-
system	system_S5D5	-	-	-	-	-	-	-	1402	-	-
system	system_S5D9	-	-	-	-	-	-	-	-	1506	-
system	system_S7G2	-	-	-	-	-	-	-	-	-	1658
telnet	sf_comms_telnet	-	-	-	-	-	-	-	1400	1400	1400
thread_monitor	sf_thread_monitor	976	1032	1032	1028	1028	1028	1028	1028	1028	1028
uart	sf_uart_comms	980	1056	1056	1004	1004	1004	1004	1004	1004	1004
ul_gx	gx_display_driver_synergy_dave2d	-	-	-	-	-	-	-	-	10488	10488
ul_gx	gx_display_driver_synergy_dave2d_8bit_palette	-	-	-	-	-	-	-	-	5268	5268
wdt	r_wdt	708	748	748	752	752	752	752	752	752	752

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
wifi	api_init	-	-	-	-	-	-	-	-	-	460
wifi	api_ioctl	-	-	-	-	-	-	-	-	-	2816
wifi	api_stack_offload	-	-	-	-	-	-	-	-	-	12880
wifi	api_txrx	-	-	-	-	-	-	-	-	-	716
wifi	api_wmi_rx	-	-	-	-	-	-	-	-	-	1876
wifi	cust_api_init	-	-	-	-	-	-	-	-	-	420
wifi	cust_api_ioctl	-	-	-	-	-	-	-	-	-	5126
wifi	cust_api_stack_offload	-	-	-	-	-	-	-	-	-	3904
wifi	cust_api_stack_txrx	-	-	-	-	-	-	-	-	-	224
wifi	cust_api_txrx	-	-	-	-	-	-	-	-	-	244
wifi	cust_api_wmi_rx	-	-	-	-	-	-	-	-	-	1472
wifi	cust_driver_main	-	-	-	-	-	-	-	-	-	416
wifi	cust_driver_netbuf	-	-	-	-	-	-	-	-	-	1149
wifi	cust_spi_hcd	-	-	-	-	-	-	-	-	-	616
wifi	custom_qcom_api	-	-	-	-	-	-	-	-	-	396
wifi	driver_diag	-	-	-	-	-	-	-	-	-	528
wifi	driver_init	-	-	-	-	-	-	-	-	-	1320
wifi	driver_main	-	-	-	-	-	-	-	-	-	9464
wifi	driver_netbuf	-	-	-	-	-	-	-	-	-	76
wifi	driver_txrx	-	-	-	-	-	-	-	-	-	508
wifi	dset	-	-	-	-	-	-	-	-	-	616
wifi	dset_api	-	-	-	-	-	-	-	-	-	840
wifi	enet_irq_veneer	-	-	-	-	-	-	-	44	44	44
wifi	ether_phy	-	-	-	-	-	-	-	800	800	828
wifi	gt202_debug	-	-	-	-	-	-	-	-	-	896
wifi	gt202_util	-	-	-	-	-	-	-	-	-	308
wifi	gt202_wifi_ctrl	-	-	-	-	-	-	-	-	-	1482
wifi	gt202_wifi_task_thread	-	-	-	-	-	-	-	-	-	1324
wifi	htc	-	-	-	-	-	-	-	-	-	1560
wifi	hw_api	-	-	-	-	-	-	-	-	-	288
wifi	osal	-	-	-	-	-	-	-	-	-	8896
wifi	qcom_api	-	-	-	-	-	-	-	-	-	6070

Group	Module or Component	s1ja	s124	s128	s3a1	s3a3	s3a6	s3a7	s5d5	s5d9	s7g2
wifi	qcom_legacy	-	-	-	-	-	-	-	-	-	60
wifi	rcv_aggr	-	-	-	-	-	-	-	-	-	356
wifi	ring_buffer	-	-	-	-	-	-	-	-	-	296
wifi	sf_wifi_gt202	-	-	-	-	-	-	-	-	-	2912
wifi	sf_wifi_gt202_onchip_stack	-	-	-	-	-	-	-	-	-	364
wifi	sf_wifi_gt202_socket	-	-	-	-	-	-	-	-	-	1008
wifi	sf_wifi_nsal_nx	-	-	-	-	-	-	-	-	-	1230
wifi	util	-	-	-	-	-	-	-	-	-	184
wifi	wmi	-	-	-	-	-	-	-	-	-	2852

16.3 Cryptography Memory Size Estimation

Table 16-3 Cryptography (r_sce) ROM results compiled for various MCU Groups:

lib crypto gcc cm4_s7g2	
Name	ROM
aes	580
aes128	1656
aes192	1472
aes256	1656
arc4	376
dsa	48
dsa1024	208
dsa2048	416
ecc	48
ecc192	544
ecc192hrk	400
ecc256	544
ecc256hrk	396
hash	24
hrk	5800
interface	12
key	1120
md5	80
r_sce	196
rsa	504
rsa1024	528
rsa2048	476
s1	0
s3	0
s5	0
s7	740
sha1	76
sha256	76
tdes	108
tdes192	212

lib crypto iar cm4_s7g2	
Name	ROM
aes	672
aes128	1808
aes192	1616
aes256	1808
arc4	376
dsa	48
dsa1024	246
dsa2048	492
ecc	48
ecc192	536
ecc192hrk	384
ecc256	536
ecc256hrk	384
hash	24
hrk	6136
interface	14
key	1076
md5	74
r_sce	188
rsa	488
rsa1024	504
rsa2048	512
s1	0
s3	0
s5	0
s7	704
sha1	80
sha256	80
tdes	108
tdes192	246

libcrypto gcc cm4_s5d9	
Name	ROM
aes	580
aes128	1656
aes192	1472
aes256	1656
arc4	376
dsa	48
dsa1024	208
dsa2048	416
ecc	48
ecc192	544
ecc192hrk	400
ecc256	544
ecc256hrk	396
hash	24
hrk	5800
interface	12
key	1120
md5	80
r_sce	196
rsa	504
rsa1024	528
rsa2048	476
s1	0
s3	0
s5	740
s7	0
sha1	76
sha256	76
tdes	108
tdes192	212

lib crypto iar cm4_s5d9	
Name	ROM
aes	672
aes128	1808
aes192	1616
aes256	1808
arc4	376
dsa	48
dsa1024	246
dsa2048	492
ecc	48
ecc192	536
ecc192hrk	384
ecc256	536
ecc256hrk	384
hash	24
hrk	6136
interface	14
key	1076
md5	74
r_sce	188
rsa	488
rsa1024	504
rsa2048	512
s1	0
s3	0
s5	704
s7	0
sha1	80
sha256	80
tdes	108
tdes192	246

lib crypto gcc cm4_s5d5	
Name	ROM
aes	580
aes128	1656
aes192	1472
aes256	1656
arc4	376
dsa	48
dsa1024	208
dsa2048	416
ecc	48
ecc192	544
ecc192hrk	400
ecc256	544
ecc256hrk	396
hash	24
hrk	5800
interface	12
key	1120
md5	80
r_sce	196
rsa	504
rsa1024	528
rsa2048	476
s1	0
s3	0
s5	740
s7	0
sha1	76
sha256	76
tdes	108
tdes192	212

lib crypto iar cm4_s5d5	
Name	ROM
aes	672
aes128	1808
aes192	1616
aes256	1808
arc4	376
dsa	48
dsa1024	246
dsa2048	492
ecc	48
ecc192	536
ecc192hrk	384
ecc256	536
ecc256hrk	384
hash	24
hrk	6136
interface	14
key	1076
md5	74
r_sce	188
rsa	488
rsa1024	504
rsa2048	512
s1	0
s3	0
s5	704
s7	0
sha1	80
sha256	80
tdes	108
tdes192	246

lib crypto gcc cm4_s7g2	
trng	292

lib crypto iar cm4_s7g2	
trng	252
sc32	

libcrypto gcc cm4_s5d9	
trng	292

lib crypto iar cm4_s5d9	
trng	252

lib crypto gcc cm4_s5d5	
trng	292

lib crypto iar cm4_s5d5	
trng	252

lib crypto gcc cm4_s3a7	
Name	**ROM**
aes	580
aes128	1656
aes256	1656
ecc	48
ecc192	544
ecc192hrk	400
ecc256	544
ecc256hrk	396
hrk	3912
interface	12
key	908
r_sce	196
s1	0
s3	380
s5	0
s7	0
trng	292

lib crypto iar cm4_s3a7	
Name	**ROM**
aes	672
aes128	1808
aes256	1808
ecc	48
ecc192	536
ecc192hrk	384
ecc256	536
ecc256hrk	384
hrk	4238
interface	14
key	872
r_sce	188
s1	0
s3	356
s5	0
s7	0
trng	252

lib crypto gcc cm4_s3a6	
Name	**ROM**
aes	580
aes128	1656
aes256	1656
ecc	48
ecc192	544
ecc192hrk	400
ecc256	544
ecc256hrk	396
hrk	3912
interface	12
key	908
r_sce	196
s1	0
s3	380
s5	0
s7	0
trng	292

lib crypto iar cm4_s3a6	
Name	**ROM**
aes	672
aes128	1808
aes256	1808
ecc	48
ecc192	536
ecc192hrk	384
ecc256	536
ecc256hrk	384
hrk	4238
interface	14
key	872
r_sce	188
s1	0
s3	356
s5	0
s7	0
trng	252

lib crypto gcc cm4_s3a3	
Name	**ROM**
aes	580
aes128	1656
aes256	1656
ecc	48
ecc192	544
ecc192hrk	400
ecc256	544
ecc256hrk	396
hrk	3912
interface	12
key	908
r_sce	196
s1	0
s3	380
s5	0
s7	0
trng	292

lib crypto iar cm4_s3a3	
Name	**ROM**
aes	672
aes128	1808
aes256	1808
ecc	48
ecc192	536
ecc192hrk	384
ecc256	536
ecc256hrk	384
hrk	4238
interface	14
key	872
r_sce	188
s1	0
s3	356
s5	0
s7	0
trng	252

lib crypto gcc cm0_s128	
Name	**ROM**
aes	440
aes128	392
aes192	392
aes256	392
ecc	56
ecc192	624
ecc192hrk	456

lib crypto iar cm0_s128	
Name	**ROM**
aes	608
aes128	436
aes256	436
ecc	56
ecc192	848
ecc192hrk	584
ecc256	848

lib crypto gcc cm0_s124	
Name	**ROM**
aes	440
aes128	392
aes192	392
aes256	392
ecc	56
ecc192	624
ecc192hrk	456

lib crypto iar cm0_s124	
Name	**ROM**
aes	608
aes128	436
aes256	436
ecc	56
ecc192	848
ecc192hrk	584
ecc256	848

libcrypto iar cm23_s1ja	
Name	**ROM**
aes	592
aes128	440
aes256	440
ecc	56
ecc192	832
ecc192hrk	576
ecc256	832

lib crypto gcc cm0_s128	
ecc256	624
ecc256hrk	448
hrk	1744
interface	20
key	856
r_sce	220
s1	148
s3	0
s5	0
s7	0
trng	292

lib crypto iar cm0_s128	
ecc256hrk	584
hrk	1804
interface	24
key	764
r_sce	224
s1	224
s3	0
s5	0
s7	0
trng	268

lib crypto gcc cm0_s124	
ecc256	624
ecc256hrk	448
hrk	1744
interface	20
key	856
r_sce	220
s1	148
s3	0
s5	0
s7	0
trng	292

lib crypto iar cm0_s124	
ecc256hrk	584
hrk	1804
interface	24
key	764
r_sce	224
s1	224
s3	0
s5	0
s7	0
trng	268

libcrypto iar cm23_s1ja	
ecc256hrk	576
hrk	1808
interface	20
key	756
r_sce	224
s1	200
s3	0
s5	0
s7	0
trng	268

16.4 DSP Library Memory Size Estimation

Table 16-4 DSP (Arm®) Library Results Compiled for Various MCU Groups:

DSP_Lib gcc cm4	
Name	**ROM**
abs	692
add	512
biquad	6368
bitreversal	500
bitreversal2	192
cfft	14844
cmplx	5372
common	216778
const	1056
conv	20868
copy	332
correlate	10012
cos	276
dct4	328216
dot	680
fill	240

DSP_Lib iar cm4	
Name	**ROM**
abs	476
add	368
biquad	5174
bitreversal	490
bitreversal2	192
cfft	12894
cmplx	3858
common	216780
const	1056
conv	14942
copy	216
correlate	6986
cos	268
dct4	327888
dot	460
fill	190

DSP_Lib gcc cm0 plus	
Name	**ROM**
abs	196
add	280
biquad	4324
bitreversal	552
bitreversal2	80
cfft	21404
cmplx	2304
common	216778
const	1056
conv	17444
copy	92
correlate	8728
cos	312
dct4	327532
dot	332
fill	56

DSP_Lib ia rcm0 plus	
Name	**ROM**
abs	178
add	256
biquad	4030
bitreversal	470
bitreversal2	80
cfft	17834
cmplx	1838
common	216780
const	1056
conv	14318
copy	72
correlate	7020
cos	288
dct4	327314
dot	198
fill	62

DSP_Lib iar cm23	
Name	**ROM**
abs	176
add	254
biquad	4028
bitreversal	466
bitreversal2	80
cfft	17810
cmplx	1828
common	216780
const	1056
conv	14278
copy	64
correlate	6998
cos	288
dct4	327314
dot	190
fill	52

DSP_Lib gcc cm4	
fir	18192
float	624
iir	1932
lms	4508
mat	8088
max	720
mean	384
min	720
mult	700
negate	424
offset	428
pid	276
power	488
q15	360
q31	328
q7	388
rfft	166896
rms	408
scale	880
shift	932
sin	1136
sqrt	484
std	620
sub	516
var	644

DSP_Lib iar cm4	
fir	13524
float	560
iir	1464
lms	3326
mat	6214
max	492
mean	334
min	492
mult	504
negate	334
offset	322
pid	218
power	342
q15	248
q31	256
q7	262
rfft	166578
rms	314
scale	652
shift	636
sin	1076
sqrt	464
std	512
sub	366
var	530

DSP_Lib gcc cm0 plus	
fir	10908
float	248
iir	1168
lms	3080
mat	7276
max	248
mean	196
min	248
mult	288
negate	184
offset	272
pid	440
power	212
q15	88
q31	80
q7	88
rfft	167904
rms	324
scale	364
shift	356
sin	1332
sqrt	560
std	548
sub	280
var	480

DSP_Lib ia rcm0 plus	
fir	9122
float	224
iir	1016
lms	2584
mat	6364
max	180
mean	164
min	180
mult	264
negate	172
offset	250
pid	348
power	178
q15	84
q31	78
q7	84
rfft	167468
rms	274
scale	340
shift	300
sin	1264
sqrt	564
std	460
sub	258
var	362

DSP_Lib iar cm23	
fir	8932
float	224
iir	1006
lms	2554
mat	6312
max	180
mean	152
min	180
mult	262
negate	168
offset	250
pid	340
power	170
q15	80
q31	74
q7	80
rfft	167462
rms	270
scale	332
shift	292
sin	1264
sqrt	548
std	452
sub	258
var	358

16.5 X-Ware Memory Size Estimation

Table 16-5 X-Ware Results Compiled for Various MCU Groups

FileX (fx)

fx gcc cm0plus	
Name	**ROM**
directory	12932
fault	0
file	10088
media	7755
partition	392
ram	220
system	986
trace	0
unicode	4720
utility	5796

fx iar cm0plus	
Name	**ROM**
directory	11764
fault	0
file	9402
media	6806
partition	382
ram	160
system	636
trace	0
unicode	4442
utility	5136

fx gcc cm4	
Name	**ROM**
directory	12100
fault	0
file	9424
media	7471
partition	376
ram	204
system	870
trace	0
unicode	4484
utility	5440

fx iar cm4	
Name	**ROM**
directory	10890
fault	0
file	8238
media	6322
partition	354
ram	156
system	592
trace	0
unicode	4272
utility	4530

fx iar cm23	
Name	**ROM**
directory	11246
fault	0
file	9118
media	6592
partition	348
ram	160
system	600
trace	0
unicode	4220
utility	4898

FileX Error Checking (fxe)

fxe gcc cm0plus	
Name	**ROM**
directory	1044
fault	0
file	1616
media	1048
system	228
unicode	648

fxe iar cm0plus	
Name	**ROM**
directory	980
fault	0
file	1452
media	988
system	168
unicode	640

fxe gcc cm4	
Name	**ROM**
directory	924
fault	0
file	1420
media	916
system	184
unicode	564

fxe iar cm4	
Name	**ROM**
directory	934
fault	0
file	1404
media	944
system	138
unicode	672

fxe iar cm23	
Name	**ROM**
directory	866
fault	0
file	1356
media	892
system	144
unicode	572

GUIX (gx)

gx gcc cm0plus	
Name	**ROM**
accordion	1440
animation	2912
binres	4160
brush	40
button	856

gx iar cm0plus	
Name	**ROM**
accordion	1330
animation	2434
binres	3680
brush	36
button	652

gx gcc cm4	
Name	**ROM**
accordion	1444
animation	2736
binres	3304
brush	48
button	756

gx iar cm4	
Name	**ROM**
accordion	1326
animation	2334
binres	3572
brush	36
button	648

gx iar cm23	
Name	**ROM**
accordion	1296
animation	2360
binres	3656
brush	36
button	652

gx gcc cm0plus	
canvas	7508
checkbox	604
circular	1256
context	608
display	116353
drop	788
horizontal	3240
icon	848
image	12881
line	556
menu	778
monochrome	88
multi	9158
numeric	472
pixelmap	1576
popup	92
progress	886
prompt	336
radial	2022
radio	480
screen	144
scroll	2532
scrollbar	2208
single	5188
slider	1416
sprite	592
string	248
system	25540
text	2432
touch	1228
tree	2556
utility	36440

gx iar cm0plus	
canvas	6324
checkbox	608
circular	1170
context	590
display	100020
drop	736
horizontal	2770
icon	774
image	12644
line	510
menu	764
monochrome	84
multi	8048
numeric	450
pixelmap	1486
popup	100
progress	796
prompt	342
radial	1766
radio	486
screen	136
scroll	2326
scrollbar	1992
single	4570
slider	1282
sprite	574
string	230
system	24844
text	2198
touch	1552
tree	2278
utility	28330

gx gcc cm4	
canvas	7032
checkbox	596
circular	1240
context	588
display	108572
drop	756
horizontal	2852
icon	808
image	11933
line	544
menu	842
monochrome	88
multi	8182
numeric	460
pixelmap	1592
popup	96
progress	858
prompt	340
radial	1830
radio	472
screen	144
scroll	2544
scrollbar	2152
single	4836
slider	1444
sprite	604
string	252
system	25428
text	2276
touch	1220
tree	2368
utility	34000

gx iar cm4	
canvas	6104
checkbox	578
circular	1130
context	552
display	92784
drop	730
horizontal	2662
icon	762
image	12046
line	492
menu	754
monochrome	84
multi	7520
numeric	436
pixelmap	1472
popup	100
progress	798
prompt	330
radial	1728
radio	464
screen	140
scroll	2302
scrollbar	1950
single	4464
slider	1308
sprite	546
string	236
system	24614
text	2096
touch	1390
tree	2212
utility	27292

gx iar cm23	
canvas	6130
checkbox	590
circular	1130
context	556
display	98766
drop	722
horizontal	2722
icon	762
image	12374
line	496
menu	744
monochrome	84
multi	7880
numeric	434
pixelmap	1430
popup	98
progress	762
prompt	328
radial	1738
radio	472
screen	136
scroll	2222
scrollbar	1958
single	4536
slider	1272
sprite	556
string	220
system	24598
text	2144
touch	1518
tree	2226
utility	27914

gx gcc cm0plus	
vertical	3252
widget	6800
window	1948

gx iar cm0plus	
vertical	2812
widget	6008
window	1690

gx gcc cm4	
vertical	2864
widget	6284
window	1696

gx iar cm4	
vertical	2686
widget	5824
window	1602

gx iar cm23	
vertical	2764
widget	5826
window	1638

GUIX Error Checking (gxe)

gxe gcc cm0plus	
Name	**ROM**
accordion	304
animation	336
binres	48
brush	168
button	316
canvas	2716
checkbox	412
circular	468
context	1564
display	672
drop	564
horizontal	900
icon	516
image	292
line	324
menu	400
multi	1896
numeric	856
pixelmap	828
progress	868
prompt	704
radial	672
radio	192
screen	348
scroll	1060
scrollbar	352

gxe iar cm0plus	
Name	**ROM**
accordion	280
animation	316
binres	40
brush	160
button	286
canvas	2600
checkbox	368
circular	408
context	1492
display	620
drop	532
horizontal	856
icon	420
image	268
line	284
menu	372
multi	1744
numeric	776
pixelmap	772
progress	784
prompt	668
radial	620
radio	160
screen	340
scroll	976
scrollbar	348

gxe gcc cm4	
Name	**ROM**
accordion	288
animation	296
binres	32
brush	160
button	268
canvas	2584
checkbox	372
circular	452
context	1428
display	640
drop	572
horizontal	912
icon	468
image	268
line	308
menu	412
multi	1892
numeric	884
pixelmap	836
progress	856
prompt	704
radial	664
radio	180
screen	320
scroll	1032
scrollbar	344

gxe iar cm4	
Name	**ROM**
accordion	264
animation	280
binres	30
brush	148
button	262
canvas	2492
checkbox	354
circular	398
context	1356
display	592
drop	500
horizontal	820
icon	406
image	264
line	276
menu	364
multi	1686
numeric	744
pixelmap	754
progress	760
prompt	664
radial	592
radio	154
screen	312
scroll	944
scrollbar	328

gxe iar cm23	
Name	**ROM**
accordion	272
animation	270
binres	28
brush	164
button	266
canvas	2596
checkbox	362
circular	400
context	1516
display	624
drop	524
horizontal	844
icon	408
image	264
line	284
menu	372
multi	1728
numeric	768
pixelmap	736
progress	780
prompt	664
radial	616
radio	150
screen	340
scroll	968
scrollbar	348

gxe gcc cm0plus	
single	1316
slider	916
sprite	472
string	316
system	2192
text	1328
tree	732
utility	968
vertical	928
widget	4340
window	1692

gxe iar cm0plus	
single	1248
slider	860
sprite	428
string	292
system	2084
text	1216
tree	688
utility	870
vertical	888
widget	4160
window	1612

gxe gcc cm4	
single	1296
slider	888
sprite	448
string	320
system	2064
text	1284
tree	732
utility	896
vertical	940
widget	4232
window	1608

gxe iar cm4	
single	1164
slider	840
sprite	400
string	288
system	1952
text	1188
tree	660
utility	812
vertical	848
widget	4008
window	1536

gxe iar cm23	
single	1248
slider	844
sprite	416
string	288
system	2100
text	1216
tree	684
utility	782
vertical	876
widget	4156
window	1592

NetX (nx)

nx gcc cm0plus	
Name	**ROM**
arp	3812
icmp	1368
igmp	1904
ip	8905
packet	1692
ram	856
rarp	768
system	178
tcp	14976
trace	0
udp	3560

nx iar cm0plus	
Name	**ROM**
arp	3360
icmp	1258
igmp	2056
ip	7588
packet	1516
ram	880
rarp	724
system	156
tcp	13102
trace	0
udp	3254

nx gcc cm4	
Name	**ROM**
arp	3520
icmp	1344
igmp	1704
ip	8113
packet	1636
ram	712
rarp	720
system	182
tcp	14392
trace	0
udp	3392

nx iar cm4	
Name	**ROM**
arp	3138
icmp	1254
igmp	1892
ip	6942
packet	1460
ram	828
rarp	676
system	156
tcp	12438
trace	0
udp	3046

nx iar cm23	
Name	**ROM**
arp	3230
icmp	1220
igmp	2012
ip	7290
packet	1456
ram	852
rarp	688
system	156
tcp	12694
trace	0
udp	3124

NetX (nxe) Error Checking

nxe gcc cm0plus	
Name	**ROM**
arp	868
icmp	284
igmp	676
ip	2604
packet	1000
rarp	232
tcp	2768
udp	1856

nxe iar cm0plus	
Name	**ROM**
arp	856
icmp	272
igmp	660
ip	2528
packet	880
rarp	228
tcp	2700
udp	1802

nxe gcc cm4	
Name	**ROM**
arp	796
icmp	248
igmp	600
ip	2476
packet	924
rarp	196
tcp	2624
udp	1724

nxe iar cm4	
Name	**ROM**
arp	816
icmp	252
igmp	584
ip	2338
packet	828
rarp	200
tcp	2496
udp	1618

nxe iar cm23	
Name	**ROM**
arp	796
icmp	248
igmp	604
ip	2404
packet	792
rarp	212
tcp	2484
udp	1668

NetX Duo (nxd) IPV4 and IPV6

nxd gcc cm4	
Name	**ROM**
arp	3884
icmp	1440
icmpv4	832
icmpv6	5488
igmp	1696
invalidate	100
ip	11073
ipv4	1016
ipv6	3788
nd	1200
packet	1704
ram	1168
rarp	768
system	281
tcp	14364
trace	0
udp	2956

nxd iar cm4	
Name	**ROM**
arp	3460
icmp	1318
icmpv4	780
icmpv6	5092
igmp	1714
invalidate	96
ip	9692
ipv4	826
ipv6	3408
nd	1188
packet	1600
ram	1156
rarp	708
system	240
tcp	12790
trace	0
udp	2756

NetX Duo (nxd) IPV6 only

nxd gcc cm4	
Name	**ROM**
icmp	264
icmpv6	36
ip	184
ipv6	3820
nd	340
tcp	548
udp	612

nxd iar cm4	
Name	**ROM**
icmp	250
icmpv6	34
ip	180
ipv6	3426
nd	314
tcp	496
udp	578

NetX Duo IPV6 Error Checking (nxde)

nxde gcc cm4	
Name	**ROM**
icmp	328
icmpv6	68
ip	404
ipv6	832
nd	388
tcp	220
udp	608

nxde iar cm4	
Name	**ROM**
icmp	332
icmpv6	68
ip	384
ipv6	812
nd	368
tcp	216
udp	576

NetX Duo IPV4 Error Checking (nxe)

nxe gcc cm4	
Name	**ROM**
arp	944
icmp	248
igmp	688
ip	3092
ipv4	184
packet	948
rarp	196
tcp	2700
udp	1732

nxe iar cm4	
Name	**ROM**
arp	976
icmp	252
igmp	704
ip	2924
ipv4	192
packet	852
rarp	200
tcp	2572
udp	1630

el iar cm23	
Name	**ROM**
md5	2468

ThreadX (tx)

tx gcc cm0plus	
Name	**ROM**
block	992
byte	1404
event	1280
initialize	376
isr	8
misra	0
mutex	1720
queue	2036
semaphore	1004
thread	3965
time	40
timer	1416
trace	120

tx iar cm0plus	
Name	**ROM**
block	936
byte	1332
event	1150
initialize	360
isr	4
misra	0
mutex	1596
queue	1832
semaphore	922
thread	3386
time	40
timer	1230
trace	120

tx gcc cm4	
Name	**ROM**
block	1004
byte	1352
event	1180
initialize	392
isr	8
misra	0
mutex	1700
queue	1876
semaphore	940
thread	3573
time	40
timer	1368
trace	104

tx iar cm4	
Name	**ROM**
block	900
byte	1288
event	1080
initialize	280
isr	4
misra	0
mutex	1538
queue	1756
semaphore	896
thread	3358
time	40
timer	1198
trace	104

tx iar cm23	
Name	**ROM**
block	896
byte	1284
event	1100
initialize	360
isr	4
misra	0
mutex	1544
queue	1776
semaphore	888
thread	3296
time	40
timer	1182
trace	96

ThreadX (txe) Error Checking

txe gcc cm0plus	
Name	**ROM**
block	448
byte	488
event	408
mutex	460
queue	596
semaphore	452
thread	912
timer	432

txe iar cm0plus	
Name	**ROM**
block	404
byte	456
event	372
mutex	432
queue	544
semaphore	408
thread	812
timer	396

txe gcc cm4	
Name	**ROM**
block	404
byte	444
event	360
mutex	420
queue	548
semaphore	396
thread	824
timer	408

txe iar cm4	
Name	**ROM**
block	388
byte	436
event	356
mutex	404
queue	532
semaphore	372
thread	792
timer	376

txe iar cm23	
Name	**ROM**
block	384
byte	432
event	352
mutex	408
queue	524
semaphore	368
thread	764
timer	372

USBX (ux)

ux gcc cm0plus	
Name	**ROM**
device	4755
host	6930
system	382
trace	0
utility	1784

ux iar cm0plus	
Name	**ROM**
device	4268
host	6356
system	376
trace	0
utility	1698

ux gcc cm4	
Name	**ROM**
device	4367
host	6750
system	382
trace	0
utility	1812

ux iar cm4	
Name	**ROM**
device	4032
host	6180
system	400
trace	0
utility	1632

ux iar cm23	
Name	**ROM**
device	4184
host	6174
system	376
trace	0
utility	1614

ux_device_class_cdc_acm gcc cm0plus	
Name	**ROM**
device	1475

ux_device_class_cdc_acm iar cm0plus	
Name	**ROM**
device	1296

ux_device_class_cdc_acm gcc cm4	
Name	**ROM**
device	1327

ux_device_class_cdc_acm iar cm4	
Name	**ROM**
device	1268

ux_device_class_cdc_acm iar cm23	
Name	**ROM**
device	1270

ux_device_class_cdc_ecm gcc cm0plus	
Name	**ROM**
device	2326

ux_device_class_cdc_ecm iar cm0plus	
Name	**ROM**
device	2050

ux_device_class_cdc_ecm gcc cm4	
Name	**ROM**
device	2102

ux_device_class_cdc_ecm iar cm4	
Name	**ROM**
device	1962

ux_device_class_cdc_ecm iar cm23	
Name	**ROM**
device	2010

ux_device_class_hid gcc cm0plus	
Name	**ROM**
device	1339

ux_device_class_hid iar cm0plus	
Name	**ROM**
device	1196

ux_device_class_hid gcc cm4	
Name	**ROM**
device	1187

ux_device_class_hid iar cm4	
Name	**ROM**
device	1176

ux_device_class_hid iar cm23	
Name	**ROM**
device	1176

ux_device_class_rndis gcc cm0plus	
Name	**ROM**
device	3580

ux_device_class_rndis iar cm0plus	
Name	**ROM**
device	3054

ux_device_class_rndis gcc cm4	
Name	**ROM**
device	3272

ux_device_class_rndis iar cm4	
Name	**ROM**
device	2878

ux_device_class_rndis iar cm23	
Name	**ROM**
device	2994

ux_device_class_storage gcc cm0plus	
Name	**ROM**
device	7242

ux_device_class_storage iar cm0plus	
Name	**ROM**
device	6446

ux_device_class_storage gcc cm4	
Name	**ROM**
device	6926

ux_device_class_storage iar cm4	
Name	**ROM**
device	6420

ux_device_class_storage iar cm23	
Name	**ROM**
device	6416

ux_network_driver gcc cm0plus	
Name	**ROM**
network	760

ux_network_driver iar cm0plus	
Name	**ROM**
network	770

ux_host_class_audio gcc cm4	
Name	**ROM**
host	3439

ux_host_class_audio iar cm4	
Name	**ROM**
host	3154

ux_network_driver iar cm23	
Name	**ROM**
network	716

Website and Support

Support: https://synergygallery.renesas.com/support

Technical Contact Details:

- America: https://www.renesas.com/en-us/support/contact.html
- Europe: https://www.renesas.com/en-eu/support/contact.html
- Japan: https://www.renesas.com/ja-jp/support/contact.html

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Apr 12, 2018	—	Initial release
1.01	Apr 13, 2018	—	Minor changes and corrections
1.02	May 18, 2018	—	Miscellaneous changes and corrections

Synergy Software Package (SSP) v1.4.0 Datasheet

Publication Date: Rev.1.02 May 18, 2018

Published by: Renesas Electronics Corporation

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709 Quantum Plaza, No.27 ZhichunLu, Haidian District, Beijing, 100191 P. R. China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, 200333 P. R. China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

Colophon 5.0

Renesas Synergy™ Platform
Synergy Software Package (SSP) v1.4.0
Software Descriptive Datasheet

Renesas Electronics Corporation

R01DS0329EU0102