Datasheet

Synergy Software Package (SSP)

Datasheet

Renesas Synergy™ Platform
Synergy Software
SSP v1.2.0

Renesas Electronics
www.renesas.com

Rev.1.38 May 2017

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 disputes 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, drawing, chart, program, algorithm, 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 otherwise misappropriate 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, copy or otherwise misappropriation of Renesas Electronics products.
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; and 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.

   Renesas Electronics products are neither intended nor 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 damages (space and 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 third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. When using the 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 radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident arising out of the use of Renesas Electronics products beyond 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. Further, Renesas Electronics products are not subject to radiation resistance design. Please ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of 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 by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, please evaluate 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. Please investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use 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 not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.
10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.
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 majority-owned subsidiaries.

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

Renesas Synergy™ Platform

Synergy Software Package (SSP) v1.2.0

R01DS0272EU0138
Rev 1.38
May 5, 2017

Contents

List of Figures

1. Description of Renesas Synergy™ Software Package

The Synergy Software Package (SSP), the heart of the Synergy Platform, is a complete integrated software package that was created using industry best practices and tested to commercial standards.

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.

Bundled in the Renesas SSP are additional libraries, a large set of Application Frameworks, Hardware Abstraction Layer (HAL) drivers, and Board Support Packages (BSPs) that are optimized for use with Synergy Microcontrollers (MCU). Developed according to the IEC/ISO/IEEE-12207 Software Life Cycle Process standard while using the MISRA C: 2012 coding guidelines, the complete SSP has been integrated, tested and validated. The SSP is supported and maintained by Renesas on a continuous basis, and Renesas warrants the SSP to include the features as specified in this datasheet.

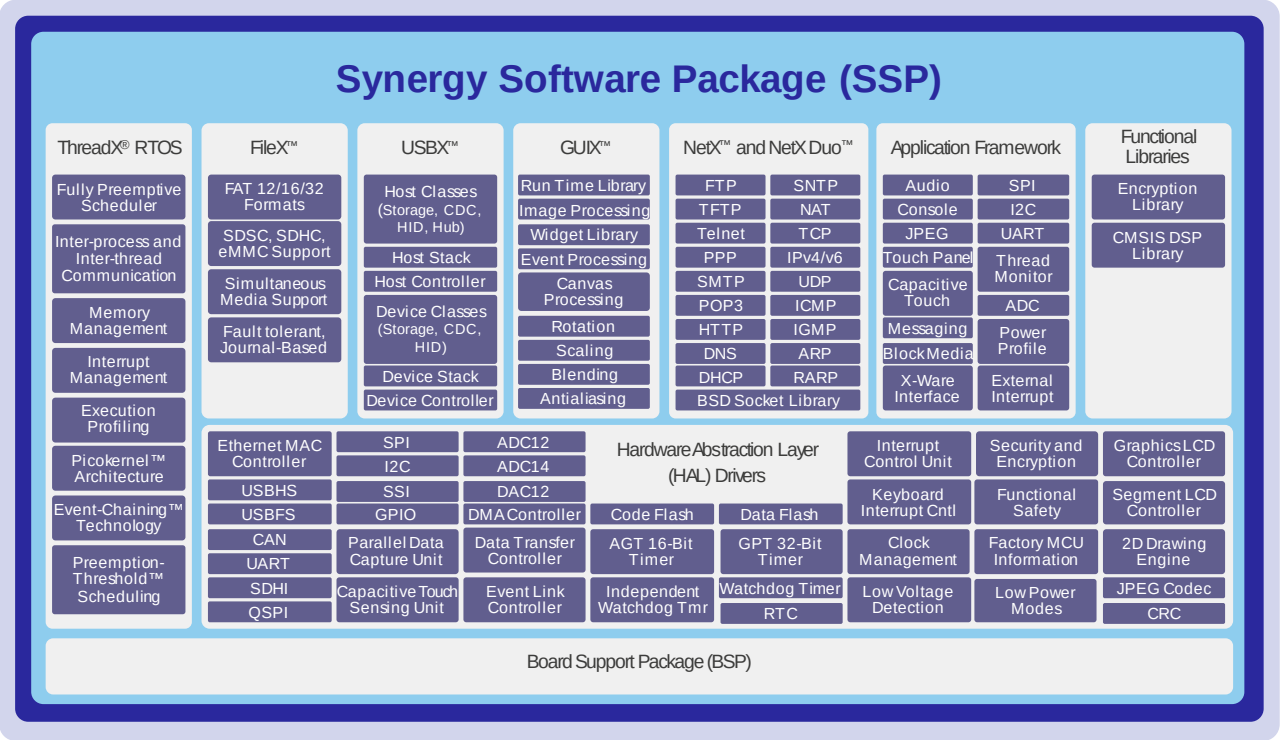


Figure 1.1 Renesas Synergy™ Software Package

SSP works with Renesas e^2 studio integrated development environment, featuring the GCC C/C++ Compiler Tool Chain, augmented and fortified with unique ISDE Software Configurators, and the IAR Embedded Workbench for Renesas Synergy – both are available as downloads at no additional charge from the Renesas Gallery.

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™ with 2D Drawing Engine

- 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 stack

- USB 2.0 Full Speed and High Speed support
- Device class: MSC, HID, CDC
- Host class: MSC, HID, CDC ACM
- Supports fast DMA 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)
- RFC 1112 Host Extensions for IP Multicasting

Application Frameworks

- Periodic Sampling ADC framework

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
- SNTP Client
- PPP

NetX Duo™ Applications (IPv4/v6 Networking Services)

- DHCP Client and Server
- DNS Client
- HTTP Client and Webserver
- TFTP Client and Server
- Telnet Client and Server
- Auto IP
- NAT
- SMTP Client
- POP3 Client and Server
- SNTP Client
- PPP

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)

- Audio Playback framework
- Audio Record framework
- Audio Playback HW DAC framework
- Audio Playback HW I²S framework
- Audio Recording HW ADC framework
- Capacitive Touch Sensing Unit framework
- Capacitive Touch Sensing Unit Button framework
- Capacitive Touch Sensing Unit Slider framework
- Console framework
- External Interrupt framework
- I²C framework
- Inter-Thread Messaging framework
- JPEG Decode framework
- Power Mode Profile framework
- Synergy FileX® Interface framework
- Synergy GUIX™ Interface framework
- Synergy NetX Communication framework
- NetX Telnet Communications framework
- Synergy USBX Communication framework
- Thread Monitor framework
- UART framework
- SPI Framework
- Block Media Interface for SD Multi Media Card
- TES D/AVE 2D over I²C framework
- Touch Panel framework

Security Cryptographic Library

- True RNG (TRNG)
- SHA1, SHA224/SHA256
- AES 128, 192, and 256-bits
- 3DES, 192-bit key
- ARC4
- RSA up to 2048-bit keys
- DLP, DSA up to 2048-bit keys

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

Board Support Package (BSP)

- Supports S124, S3A7, S5D9 and S7G2 Group MCUs
- Supports PE-HMI1, DK-S7G2, DK-S3A7, DK-S124, PK-S5D9, and SK-S7G2 Kits
- Creation of custom BSPs using e² studio
- System initialization and configuration during startup
- Software and hardware access control
- Register Write Protection

Hardware Abstract Layer (HAL) Driver Modules

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

System and Power Management

- Clock Frequency Accuracy Measurement Circuit (CAC)
- Low Power Modes driver
- Power Profiles
- RTC driver with calendar support
- Event Link Controller (ELC)
- DMA Controller (DMAC)
- Data Transfer Controller (DTC)
- Clock Generation and Management
- GPIO
- Unique ID
- Stack Pointer Overflow Monitor
- Messaging Configurator

GPIO and Key Interrupts

- GPIO module
- Key Interrupts module

1.2 Introduction to this Software Datasheet

This SSP Software Datasheet includes functional descriptions and provides performance data for the major software modules included in the Synergy Software Package (SSP).

This Software 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 (especially helpful for the Application Frameworks).
- Estimated Memory Requirements for each component.
- Tables indicating what the MCU hardware supports against what is supported (currently) in SSP for the Hardware Abstraction Layer (HAL) components.
- Performance Data and Characterizations.

Resources available in addition to this Software Datasheet are:

- SSP README.FIRST
- SSP Upgrade Guide
- SSP Software User's Manual
- SSP v1.2.0 Release Notes
- e^2 Studio Release Notes
- IAR Embedded Workbench for Synergy Release Notes
- Synergy Software Configurator Release Notes
- Application Projects with Application Notes.

The estimated memory requirements in this document specify the estimated memory consumption for each module. The requirements are as follows:

- Compiler: **GNU ARM® compiler** eabi-4_9-2015q3-20150921-win32 (Option: -O2)
- Compiler: **IAR Compiler** version 7.71.1 Embedded Workbench for Renesas Synergy (Option: -Oh)

The memory usage for the modules includes:

- Flash Memory Usage = .text + .data

All performance tests in this document are included in Section 0, and were conducted and measured on specific Synergy hardware systems, typically a Synergy Development Kit. This section also specifies the test environment for each performance test that includes:

- SSP version
- Synergy hardware (Development Kit)
- When the Synergy hardware is specific, it also identifies which Synergy MCU is used, the operating frequency, and the MCU configuration settings
- Toolchain version (including complier optimization levels).

1.2.1 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 so that they can judge how much Flash memory is required given the inclusion of certain features.

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

Name	Flash
Event	1156

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

Name	Flash
Event	360

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

It would be nearly impossible from a datasheet perspective to turn on, or off, error checking for every single function, even if that is an option Developers have to fine-tune their own code. Therefore, memory size values shown are generally worst case in that they are the sum of all of the files under each directory. Please note that by default, error checking is turned on.

Also note that different 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 MCU devices from S3A7, S5D9, and S7G2 Synergy MCU Groups.

ARM Cortex M0+ means devices from S124 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
```

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.

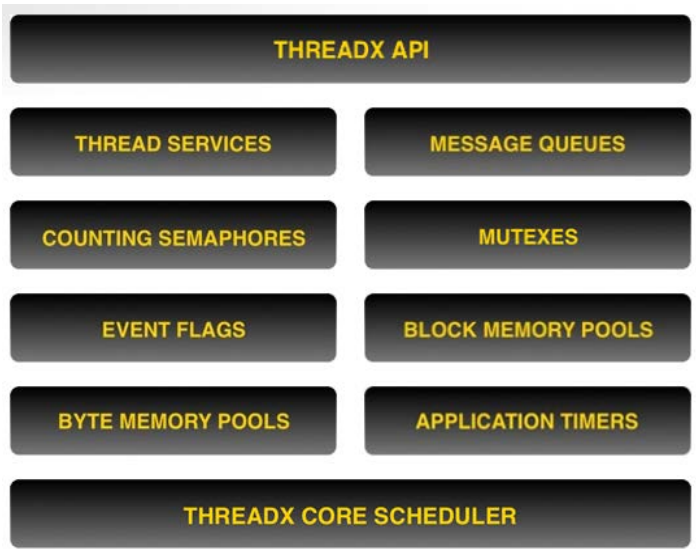
For an overview, please see: https://www.renesas.com/en-us/products/synergy/software/quality.html

The latest Synergy Software Quality Handbook and the latest Software Quality Summary can also be found on that page.

2. ThreadX® RTOS

2.1 Introduction to ThreadX and X-ware

ThreadX 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®	5.7
NetX™	5.9
NetX Duo™	5.10 SP1
NetX Application bundle	5.9 SP2
NetX Duo Application bundle	5.10 SP1
USBX™ Host	5.7 SP7
USBX™ Device	5.7 SP4
FileX®	5.4 SP2
GUIX™	5.3.2
TraceX®	5.2.0
GUIX™ Studio	5.3.2.2

Figure 2.1 ThreadX® Features and Versions

2.2 Component Introduction

At the core of Synergy Software Package (SSP) is the Express Logic, Inc. ThreadX® RTOS. Optimized for MCUs in the Synergy Microcontroller family 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 and 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, and 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

Dynamic block pool creation, and no limits on the number of block pools (except for physical memory limits), there are 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

Dynamic byte pool creation, and no limits on the number of byte pools (except for physical memory limits), there are 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, no limits on the number of timers, and periodic or one-shot timers are supported. 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 2KB flash footprint, 1KB 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.

2.3 Estimated Memory Requirements

ThreadX® ARM Cortex M4 with GCC		ThreadX® ARM Cortex M4 with IAR		ThreadX® ARM Cortex M0+ with GCC		ThreadX® ARM Cortex M0+ with IAR	
Name	**Flash**	**Name**	**Flash**	**Name**	**Flash**	**Name**	**Flash**
block	964	block	864	block	976	block	912
byte	1,344	byte	1,276	byte	1,392	byte	1,324
event	1,156	event	1,056	event	1,232	event	1,114
initialize	364	initialize	252	initialize	360	initialize	340
isr	8	isr	4	isr	8	isr	4
mutex	1,672	mutex	1,516	mutex	1,680	mutex	1,576
queue	1,784	queue	1,708	queue	1,872	queue	1,796
semaphore	904	semaphore	868	semaphore	928	semaphore	918
thread	3,257	thread	3,040	thread	3,253	thread	3,084
time	40	time	40	time	40	time	40
timer	1,328	timer	1,130	timer	1,372	timer	1,164
trace	104	trace	104	trace	120	trace	120
In addition, if error checking is turned on, add the following values to the previous data:							
ThreadX® ARM Cortex M4 with GCC		**ThreadX® ARM Cortex M4 with IAR**		**ThreadX® ARM Cortex M0+ with GCC**		**ThreadX® ARM Cortex M0+ with IAR**	
Name	**Flash**	**Name**	**Flash**	**Name**	**Flash**	**Name**	**Flash**
block	404	block	396	block	448	block	404
byte	444	byte	440	byte	488	byte	452
event	360	event	360	event	408	event	376
mutex	420	mutex	408	mutex	460	mutex	432
queue	548	queue	540	queue	596	queue	544
semaphore	396	semaphore	376	semaphore	452	semaphore	408
thread	824	thread	800	thread	912	thread	824
timer	408	timer	384	timer	432	timer	396

3. NetX™ Embedded TCP/IP and UDP Stacks

3.1 Component Introduction

SSP includes an optimized embedded TCP/IP-IPv4-compliant protocol stack, NetX™, for enabling Internet of Things (IoT) and 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 SSP v1.2.0, both e^2 studio and IAR Embedded Workbench for Synergy have Configurators to assist Developers in creating code for NetX ™.

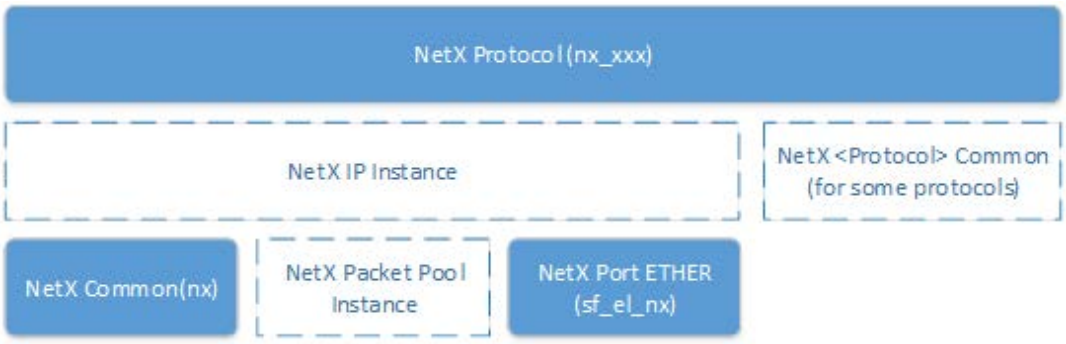


Figure 3.1 NetX ™ Stack Configuration

4. NetX Duo™ Dual IPv4/IPv6 Stack

4.1 Component Introduction

For applications requiring IPv6 support, 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 features and capabilities available with NetX™, which 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 providing flexibility to the system designer to incorporate only the applications necessary for the target application.

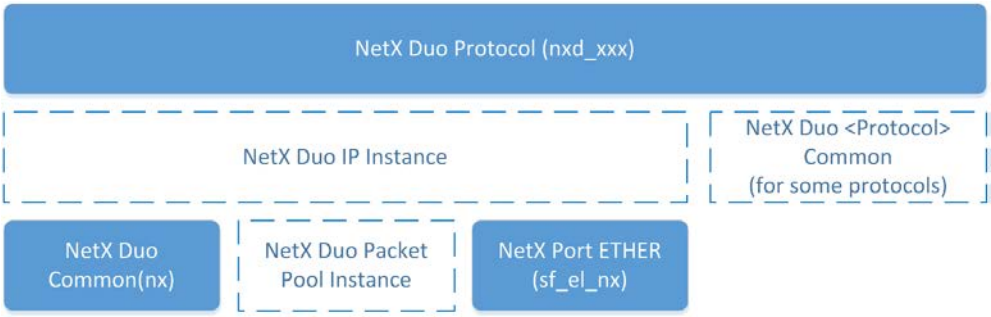


Figure 4.1 NetX Duo™ Stack Configuration

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.

4.2 Estimated Memory Requirements

NetX Duo™ ARM Cortex M4 with GCC		NetX Duo™ ARM Cortex M0+ with IAR	
Name	**Flash**	**Name**	**Flash**
arp	3,884	arp	3,498
icmp	1,704	icmp	1,590
icmpv4	868	icmpv4	794
icmpv6	5,484	icmpv6	5,184
igmp	1,696	igmp	1,700
invalidate	100	invalidate	96
ip	11,297	ip	9,854
ipv4	1,016	ipv4	820
ipv6	7,612	ipv6	6,708
nd	4,128	nd	3,696
packet	1,704	packet	1,658
ram	1,168	ram	1,180
rarp	768	rarp	706
system	278	system	236
tcp	14,744	tcp	13,294
udp	3,572	udp	3,292
In addition, if error checking is turned on, add the following values to the previous data:			
NetX Duo™ ARM Cortex M4 with GCC		**NetX Duo™ ARM Cortex M0+ with IAR**	
Name	**Flash**	**Name**	**Flash**
icmp	328	icmp	324
icmpv6	68	icmpv6	68
ip	404	ip	382
ipv6	832	ipv6	812
nd	388	nd	368
tcp	220	tcp	216
udp	608	udp	580

In SSP v1.2.0, both e^2 studio and IAR Embedded Workbench for Synergy have Configurators to assist Developers in creating code for NetX Duo™.

5. NetX™ Applications Bundle

5.1 Component Introduction

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 2616 Hypertext Transfer Protocol/1.1
  - 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.

5.2 Estimated Memory Requirements

NetX™ ARM Cortex M4 with GCC		NetX™ ARM Cortex M4 with IAR		NetX™ ARM Cortex M0+ with GCC		NetX™ ARM Cortex M0+ with IAR	
Name	**Flash**	**Name**	**Flash**	**Name**	**Flash**	**Name**	**Flash**
arp	3,456	arp	3,118	arp	3,844	arp	3,298
icmp	1,336	icmp	1,230	icmp	1,340	icmp	1,280
igmp	1,468	igmp	1,494	igmp	1,680	igmp	1,644
ip	7,980	ip	6,958	ip	8,800	ip	7,654
packet	1,632	packet	1,490	packet	1,688	packet	1,506
ram	780	ram	828	ram	868	ram	876
rarp	712	rarp	670	rarp	808	rarp	724
server	712	server	700	server	856	server	760
system	180	system	152	system	176	system	152
tcp	14,052	tcp	12,496	tcp	14,712	tcp	13,162
udp	3,376	udp	3,008	udp	3,524	udp	3,238

In addition, if error checking is turned on, add the following values to the previous data:

NetX™ ARM Cortex M4 with GCC		NetX™ ARM Cortex M4 with IAR		NetX™ ARM Cortex M0+ with GCC		NetX™ ARM Cortex M0+ with IAR	
Name	**Flash**	**Name**	**Flash**	**Name**	**Flash**	**Name**	**Flash**
arp	768	arp	792	arp	868	arp	852
icmp	256	icmp	244	icmp	284	icmp	272
igmp	592	igmp	568	igmp	664	igmp	664
ip	2,456	ip	2,298	ip	2564	ip	2,484
packet	864	packet	830	packet	956	packet	874
rarp	204	rarp	200	rarp	232	rarp	228
tcp	2,596	tcp	2,400	tcp	2,736	tcp	2,644
udp	1,704	udp	1,594	udp	1,844	udp	1,782

6. NetX Duo™ Applications Bundle

6.1 Component Introduction

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 2616 Hypertext Transfer Protocol/1.1
  - 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.

6.2 Estimated Memory Requirements

NetX Duo™ ARM Cortex M4 with GCC		NetX Duo™ ARM Cortex M4 with IAR	
Name	**Flash**	**Name**	**Flash**
arp	3,884	arp	3,498
icmp	1,704	icmp	1,590
icmpv4	868	icmpv4	794
icmpv6	5,484	icmpv6	5,184
igmp	1,696	igmp	1,700
invalidate	100	invalidate	96
ip	11,297	ip	9,854
ipv4	1,016	ipv4	820
ipv6	7,612	ipv6	6,708
nd	4,128	nd	3,696
packet	1,704	packet	1,658
ram	1,168	ram	11,80
rarp	768	rarp	706
system	278	system	236
tcp	14,744	tcp	13,294
udp	3,572	udp	3,292
In addition, if error checking is turned on, add the following values to the previous data:			
NetX Duo™ ARM Cortex M4 with GCC		**NetX Duo™ ARM Cortex M4 with IAR**	
Name	**Flash**	**Name**	**Flash**
icmp	328	icmp	324
icmpv6	68	icmpv6	68
ip	404	ip	382
ipv6	832	ipv6	812
nd	388	nd	368
tcp	220	tcp	216
udp	608	udp	580

7. FileX® Embedded File System

7.1 Component Introduction

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

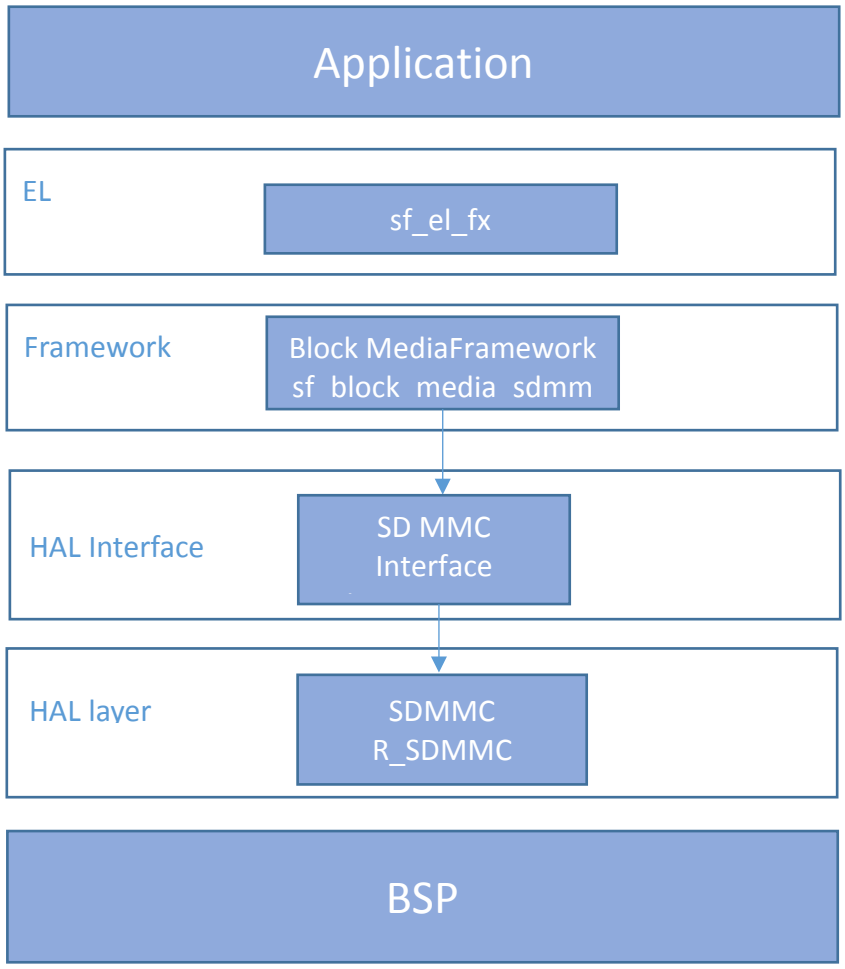


Figure 7.1 FileX® Embedded File System

7.2 Estimated Memory Requirements

FileX® ARM Cortex M4 with GCC		FileX® ARM Cortex M4 with IAR		FileX® ARM Cortex M0+ with GCC		FileX® ARM Cortex M0+ with IAR	
Name	**Flash**	**Name**	**Flash**	**Name**	**Flash**	**Name**	**Flash**
directory	12,476	directory	10,576	directory	13,456	directory	11,646
file	9,312	file	8,112	file	10,172	file	8,934
media	7,479	media	6,246	media	7,755	media	6,926
partition	376	partition	360	partition	392	partition	362
ram	204	ram	158	ram	220	ram	160
system	846	system	576	system	970	system	624
unicode	4,476	unicode	4,104	unicode	4,684	unicode	4226
utility	5,380	utility	4,506	utility	5,820	utility	5,150
In addition, if error checking is turned on, add the following values to the previous data:							
FileX® ARM Cortex M4 with GCC		**FileX® ARM Cortex M4 with IAR**		**FileX® ARM Cortex M0+ with GCC**		**FileX® ARM Cortex M0+ with IAR**	
Name	**Flash**	**Name**	**Flash**	**Name**	**Flash**	**Name**	**Flash**
directory	924	directory	916	directory	1,044	directory	968
file	1,444	file	1,404	file	1624	file	1,452
media	980	media	912	media	1,108	media	984
system	192	system	148	system	232	system	184
unicode	396	unicode	424	unicode	432	unicode	400

8. GUIX™ GUI Development Toolkit

8.1 Component Introduction

GUIX™ is SSP's graphical user interface framework. GUIX™ includes a full-featured runtime UI library and a matching GUI design application for desktop PCs, **GUIX™ Studio**. GUIX™ is fully integrated within SSP and, like ThreadX®, is designed to have a small footprint and high performance, making it ideal for today's deeply embedded applications. GUIX™ uses the same high-performance design and coding methods as ThreadX® and has been designed to meet the growing need for dynamic user interfaces with limited hardware resources. GUIX™ has minimal function call layering, and optimized clipping, drawing, and event handling making it extremely fast and responsive.

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
- 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)

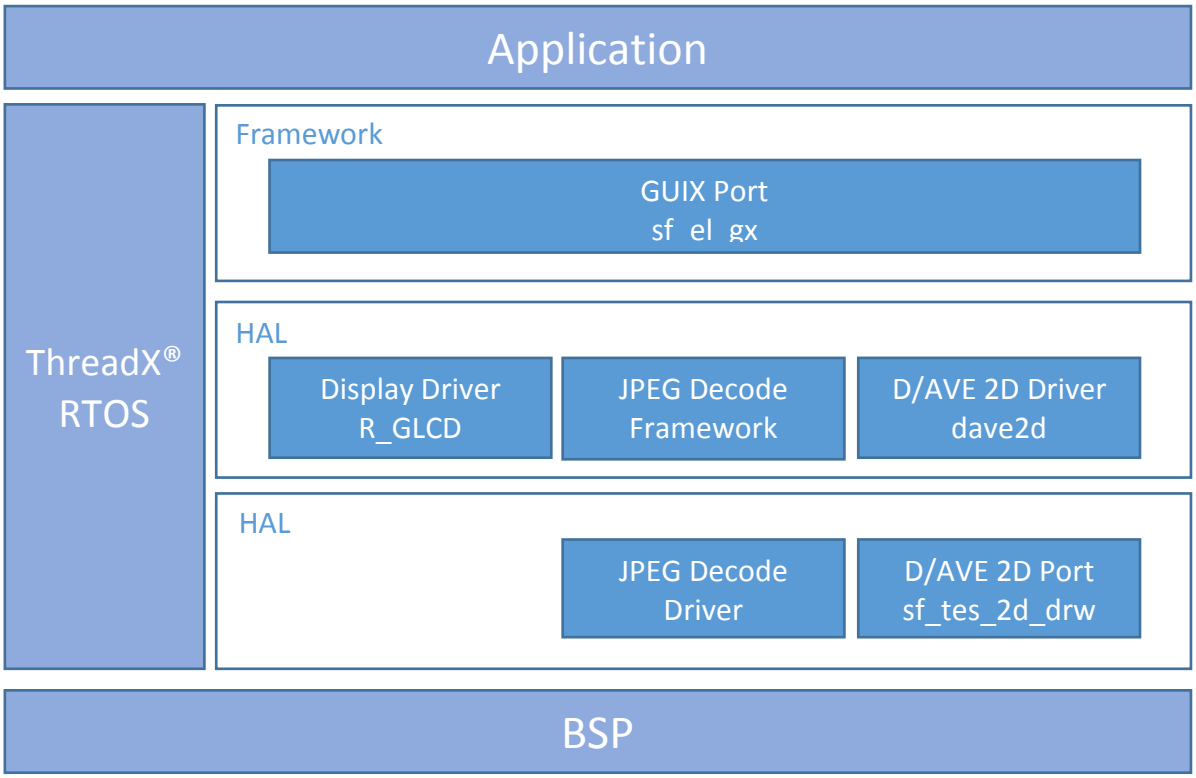


Figure 8.1 GUIX™ Runtime Library

8.2 Estimated Memory Requirements

GUIX™ ARM Cortex M4 with GCC		GUIX™ ARM Cortex M4 with IAR		GUIX™ ARM Cortex M0+ with GCC		GUIX™ ARM Cortex M0+ with IAR	
Name	**Flash**	**Name**	**Flash**	**Name**	**Flash**	**Name**	**Flash**
animation	1010	animation	932	animation	1,122	animation	1,008
brush	48	brush	36	brush	40	brush	36
button	648	button	590	button	776	button	602
canvas	6,316	canvas	5,400	canvas	6,616	canvas	5,576
checkbox	580	checkbox	564	checkbox	588	checkbox	588
circular	1,216	circular	1152	circular	1,248	circular	1,196
context	588	context	560	context	608	context	598
display	80,004	display	63,920	display	85,856	display	69,178
drop	796	drop	748	drop	820	drop	762
horizontal	2,696	horizontal	2,534	horizontal	3,076	horizontal	2,720
icon	688	icon	640	icon	716	icon	658
image	11,565	image	11,750	image	12,477	image	12,332
line	544	line	478	line	560	line	490
multi	5,760	multi	5,208	multi	6,368	multi	5,680
pixelmap	1,572	pixelmap	1,456	pixelmap	1,560	pixelmap	1,484
popup	72	popup	66	popup	68	popup	66
progress	866	progress	800	progress	894	progress	806
prompt	284	prompt	274	prompt	284	prompt	284
radial	1,258	radial	1,200	radial	1,366	radial	1,234
radio	468	radio	458	radio	496	radio	480
screen	144	screen	140	screen	144	screen	136
scroll	688	scroll	614	scroll	648	scroll	610
scrollbar	2,148	scrollbar	1,864	scrollbar	2,208	scrollbar	1,908
single	4064	single	3,712	single	4,300	single	3,892
slider	1,452	slider	1,298	slider	1,432	slider	1,304
sprite	596	sprite	550	sprite	592	sprite	568
system	24,634	system	24,008	system	24,738	system	24,160
text	392	text	376	text	388	text	386
utility	24,152	utility	17,782	utility	25,920	utility	18,684
vertical	2,704	vertical	2,554	vertical	3,084	vertical	2,758
widget	5,908	widget	5,470	widget	6,384	widget	5,702
window	1,616	window	1,560	window	1,820	window	1,628

In addition, if error checking is turned on, add the following values to the previous data:

GUIX™ ARM Cortex M4 with GCC		GUIX™ ARM Cortex M4 with IAR		GUIX™ ARM Cortex M0+ with GCC		GUIX™ ARM Cortex M0+ with IAR	
Name	**Flash**	**Name**	**Flash**	**Name**	**Flash**	**Name**	**Flash**
animation	348	animation	346	animation	384	animation	388
brush	160	brush	148	brush	168	brush	160
button	476	button	474	button	548	button	514
canvas	2,032	canvas	1,940	canvas	2,136	canvas	2,028
checkbox	476	checkbox	460	checkbox	528	checkbox	482
circular	640	circular	582	circular	672	circular	616

GUIX™ ARM Cortex M4 with GCC		GUIX ™ ARM Cortex M4 with IAR		GUIX™ ARM Cortex M0+ with GCC		GUIX™ ARM Cortex M0+ with IAR	
Name	**Flash**	**Name**	**Flash**	**Name**	**Flash**	**Name**	**Flash**
context	1,428	context	1,356	context	1,564	context	1,492
display	540	display	508	display	580	display	536
drop	568	drop	504	drop	572	drop	532
horizontal	912	horizontal	820	horizontal	900	horizontal	860
icon	620	icon	558	icon	692	icon	588
image	268	image	264	image	292	image	268
line	536	line	480	line	576	line	524
multi	1,904	multi	1,686	multi	1,916	multi	1,776
pixelmap	1,140	pixelmap	1,034	pixelmap	1,156	pixelmap	1,088
progress	944	progress	832	progress	964	progress	884
prompt	880	prompt	828	prompt	888	prompt	856
radial	892	radial	808	radial	920	radial	856
radio	284	radio	260	radio	304	radio	270
screen	320	screen	312	screen	348	screen	340
scroll	284	scroll	284	scroll	304	scroll	292
scrollbar	456	scrollbar	440	scrollbar	472	scrollbar	464
single	1,372	single	1,232	single	1,400	single	1,328
slider	1,144	slider	1,082	slider	1,204	slider	1,136
sprite	444	sprite	388	sprite	464	sprite	416
system	1,744	system	1,648	system	1,836	system	1,748
text	1,032	text	956	text	1,072	text	992
utility	608	utility	552	utility	640	utility	606
vertical	968	vertical	880	vertical	956	vertical	920
widget	4,936	widget	4660	widget	5,068	widget	4,876
window	1,696	window	1,632	window	1,784	window	1,692

9. USBX™

9.1 Component Introduction

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. USB Isochronous is not supported. Some of the key features include:

- Supports most of the standard USB class drivers including Mass Storage, HID, and CDC-ACM.
- 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.

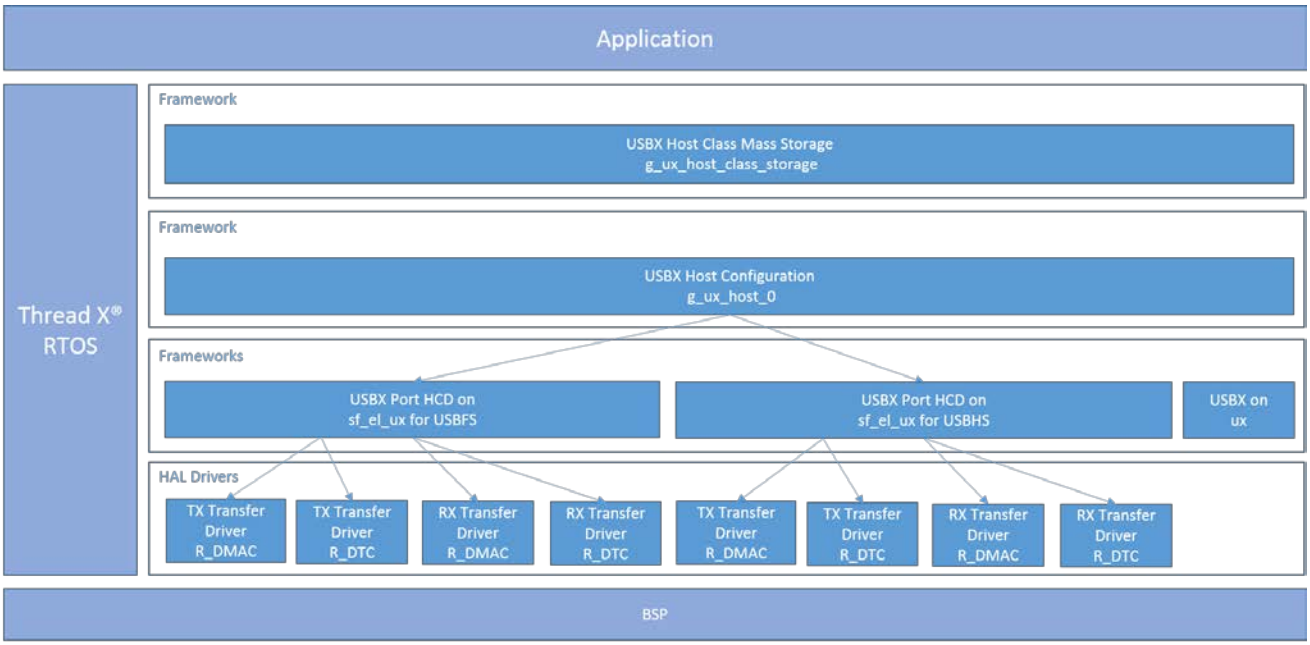


Figure 9.1 USBX™ Host Stacks for Mass Storage (MSC)

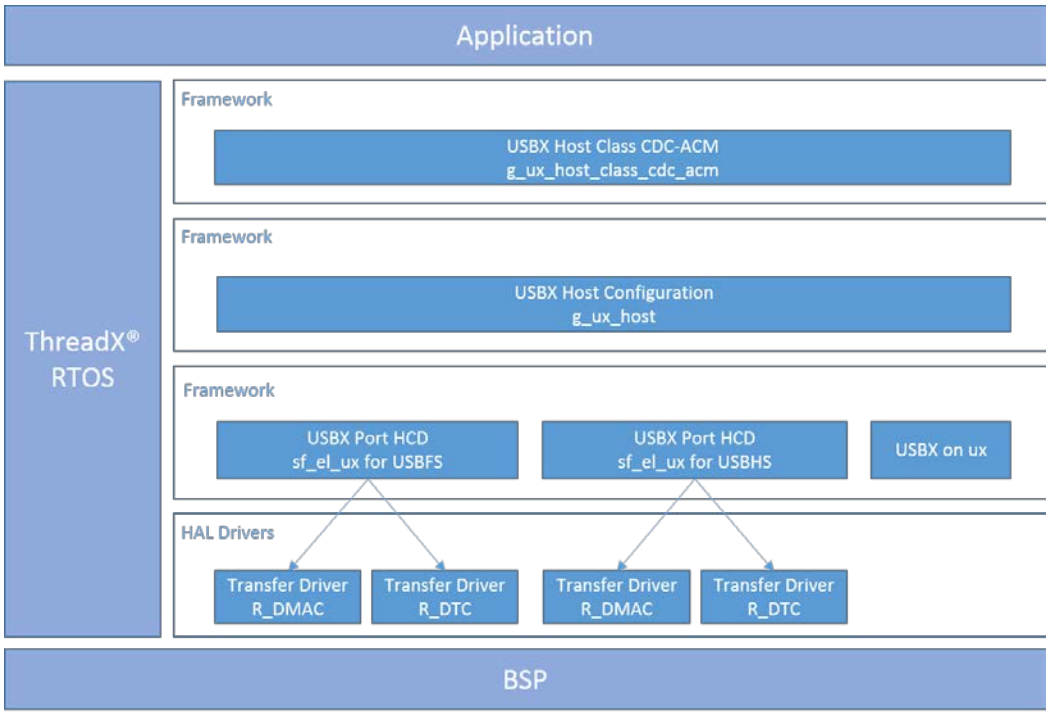


Figure 9.2 USBX™ Host Stacks for CDC ACM

	USBX Mass Storage Class				USBX HID Class				USBX CDC/ACM			
	Host		Device		Host		Device		Host		Device	
	High Speed	Full Speed	High Speed	Full Speed	High Speed	Full Speed	High Speed	Full Speed	High Speed	Full Speed	High Speed	Full Speed
S124	N/A	N/A	N/A	✓	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	✓
S5D9	✓	✓	✓	✓	✓	☒	✓	✓	✓	✓	✓	✓
S7G2	✓	✓	✓	✓	✓	☒	✓	✓	✓	✓	✓	✓

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

In SSP v1.2.0, both e^2 studio and IAR Embedded Workbench for Synergy have Configurators to assist Developers in creating code for USBX.

9.2 Estimated Memory Requirements for USBX Stack

Estimated memory requirements. Note: USBX Host Class on M0+ based MCU's is not supported.

USBX ARM Cortex M4 with GCC		**USBX ARM Cortex M4 with IAR**		**USBX ARM Cortex M0+ with GCC**		**USBX ARM Cortex M0+ with IAR**	
Name	**Flash**	**Name**	**Flash**	**Name**	**Flash**	**Name**	**Flash**
device	4046	device	3672	device	4294	device	3688
host	5709	host	5226	host	5901	host	5370
system	1273	system	1292	system	534	system	540
utility	1608	utility	1374	utility	1592	utility	1458
In addition, if error checking is turned on, add the following values to the previous data:							
ux_device_class_cdc_acm gcc cm4		ux_device_class_cdc_acm iar cm4		ux_device_class_cdc_acm gcc cm0plus		ux_device_class_cdc_acm iar cm0plus	
Name	**Flash**	**Name**	**Flash**	**Name**	**Flash**	**Name**	**Flash**
device	1015	device	1022	device	1175	device	1046
ux_device_class_cdc_ecm gcc cm4		ux_device_class_cdc_ecm iar cm4		ux_device_class_cdc_ecm gcc cm0plus		ux_device_class_cdc_ecm iar cm0plus	

Name	Flash	Name	Flash	Name	Flash	Name	Flash
device	2074	device	1932	device	2294	device	1962
ux_device_class_hid gcc cm4		ux_device_class_hid iar cm4		ux_device_class_hid gcc cm0plus		ux_device_class_hid iar cm0plus	
Name	**Flash**	**Name**	**Flash**	**Name**	**Flash**	**Name**	**Flash**
device	1027	device	1026	device	1163	device	1036
ux_device_class_hid gcc cm4		ux_device_class_hid iar cm4		ux_device_class_hid gcc cm0plus		ux_device_class_hid iar cm0plus	
Name	**Flash**	**Name**	**Flash**	**Name**	**Flash**	**Name**	**Flash**
device	3240	device	2856	device	3544	device	3006
ux_device_class_storage gcc cm4		ux_device_class_storage iar cm4		ux_device_class_storage gcc cm0plus		ux_device_class_storage iar cm0plus	
Name	**Flash**	**Name**	**Flash**	**Name**	**Flash**	**Name**	**Flash**
device	3274	device	2690	device	3598	device	2636
USBX ARM Cortex M4 with GCC		USBX ARM Cortex M4 with GCC		ux_host_class_cdc_acm gcc cm4		ux_host_class_cdc_acm iar cm4	
host	3351	host	3038	network	760	network	770
ux_host_class_cdc_acm gcc cm4		ux_host_class_acm iar cm4					
Name	**Flash**	**Name**	**Flash**				
host	2169	host	2006				
ux_host_class_gser gcc cm4		ux_host_class_gser iar cm4					
Name	**Flash**	**Name**	**Flash**				
host	1730	host	1614				
ux_host_class_hid gcc cm4		ux_host_class_hid iar cm4					
Name	**Flash**	**Name**	**Flash**				
host	6683	host	6274				
ux_host_class_hub gcc cm4		ux_host_class_hub iar cm4					
Name	**Flash**	**Name**	**Flash**				
host	1845	host	1718				
ux_host_class_printer gcc cm4		ux_host_class_printer iar cm4					
Name	**Flash**	**Name**	**Flash**				
host	1553	host	1424				
ux_host_class_prolific gcc cm4		ux_host_class_prolific iar cm4					
Name	**Flash**	**Name**	**Flash**				
host	2562	host	2262				
ux_host_class_storage gcc cm4		ux_host_class_storage iar cm4					
Name	**Flash**	**Name**	**Flash**				
host	4296	host	3958				
ux_host_class_swar gcc cm4		ux_host_class_swar iar cm4					
Name	**Flash**	**Name**	**Flash**				
host	1302	host	1188				
ux_network_driver gcc cm4		ux_network_driver iar cm4					
Name	**Flash**	**Name**	**Flash**				
network	668	network	760				

10. Application Frameworks

10.1 Introduction

Application Frameworks are a key subsystem in Renesas SSP, which, along with the Hardware Abstraction Layer (HAL), abstracts hardware peripherals as well as linking software features together to accomplish one or more tasks. Application Frameworks provides a uniform and consistent programming interface with standardized (Applications Programming Interface) APIs for system designers and Developers. This allows them to access and use most major features in SSP without having to worry about the complexity of the underlying low-level device interfaces in the platform.

These Applications Frameworks give the Developer the flexibility to program at a higher level of abstraction (saving a lot of tedious programming work) as well as not prohibiting direct access to the HAL layer, or the BSP.

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

Framework API's are controlled, engineered by Renesas, maintained, and provide consistency. If the Developer wishes to access HAL driver directly (peripheral drivers), they may do so. However, HAL drivers don't use any RTOS objects nor make any RTOS API calls. However, HAL drivers can be used with, or without, the RTOS.

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

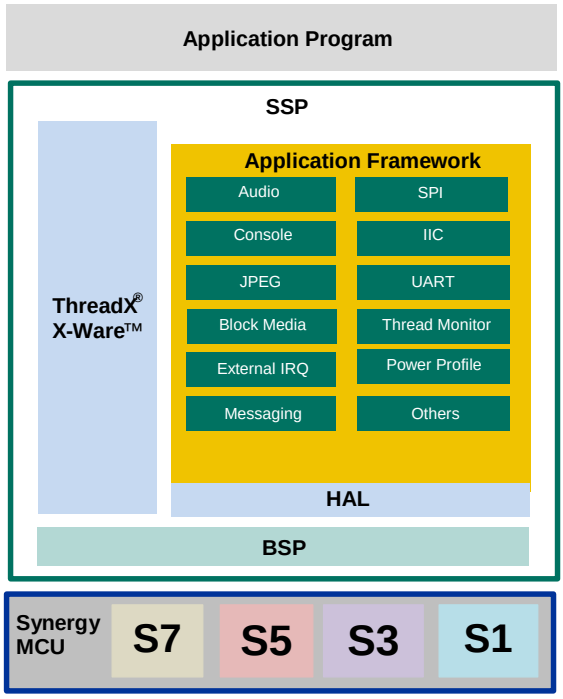


Figure 10.1 Application Frameworks

10.2 Audio Playback Framework

10.2.1 Overview of the SSP Audio Playback Stack

The audio playback framework supports multiple streams on a single hardware port. A block diagram of the modules required if two streams are used is shown in the image on the next page:

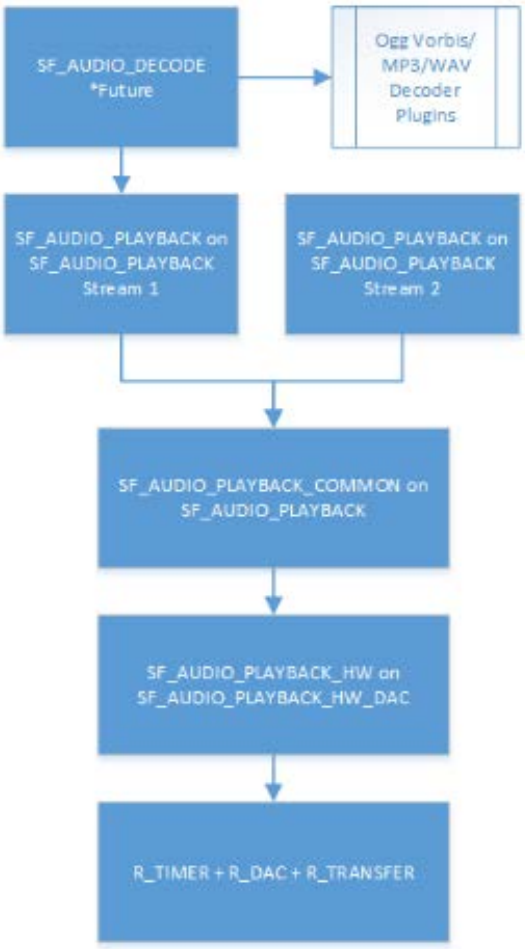


Figure 10.2 Audio Playback Stack

10.2.2 Component Introduction

The Audio Playback Framework in SSP provides standardized, C-callable, high-level APIs for playback of audio content. The framework handles the integration and synchronization of multiple HAL peripherals like timers, DMA, and DAC to facilitate audio playback. The Audio Playback Framework APIs are thread-safe and abstract underlying MCU hardware features; for example, timers, Sampling Rate Convertor, and DACs. The Audio Playback Framework supports 16-bit mono uncompressed (linear) PCM samples and lets Developers plug in custom components. The framework supports single instantiation.

Playback control features provided with Audio Framework:

- Open audio device for audio playback.
- Close audio device opened for audio playback.
- Start audio playback.
- Stop audio playback.
- Pause audio playback. Resume audio playback.
- Set software volume control.

Audio Playback Framework supports following output peripherals:

- DAC
- PWM

Other codecs may be supported in the future or can be added by the Developer.

Audio Playback Framework features:

- Plays long buffers by splitting the data into smaller manageable blocks.
- Repeat/loop playback of supplied audio data until ThreadX® timeout.
- Can request next data using callback after last buffer playback begins.
- Scaling to playback signed 16-bit PCM samples through unsigned 12-bit DAC.

- Basic mixing of multiple streams.
- The Audio Playback Framework does not support reading files in a file system and decoding audio. These functions are performed outside of the Audio Playback Framework by user-written software.

10.2.3 Estimated Memory Requirements

Table 10.1 Memory Usage for Audio Playback Framework - GCC Compiler

sf_audio_playback	Flash (Bytes)
S124 MCU Group	3,459
S3A7 MCU Group	3,263
S5D9 MCU Group	3,263
S7G2 MCU Group	3,263

Table 10.2 Memory Usage for Audio Playback Framework - IAR Compiler

sf_audio_playback	Flash (Bytes)
S124 MCU Group	3,376
S3A7 MCU Group	3,168
S5D9 MCU Group	3,168
S7G2 MCU Group	3,168

10.3 Inter-Thread Messaging Framework

10.3.1 Component Introduction

The Inter-Thread Messaging Framework in SSP provides easy to use high-level APIs for inter-thread communication and synchronization. The Inter-Thread Messaging Framework implements a lightweight, event-driven message-passing mechanism that lets applications pass messages between two or more threads. The Messaging Framework makes it simple to use the ThreadX® message queue mechanism for passing messages, and provides the following additional features beyond the basic RTOS message queue services:

- Message management: The framework supports buffer control blocks to manage each message.
- Message buffering: The framework manages buffer allocation and release for messages.
- Message publish/subscribe mechanism: The framework allows multiple threads to listen to an Event Class without the message producer thread knowing who is subscribing to a message for the Event Class and the subscribers not knowing who produces the message.
- Handshaking: The framework provides an option for handshakes between a message producer and a consumer thread by invoking a specified callback function of the producer thread from a consumer 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 provides the capability to post high priority messages that receive precedence for delivery.
- Messaging Framework provides user applications a buffer that is allocated in the memory pool to store the message header and payload. The framework also has the provision for user applications to release the buffer.
- Framework supports unicast and multicast messaging.

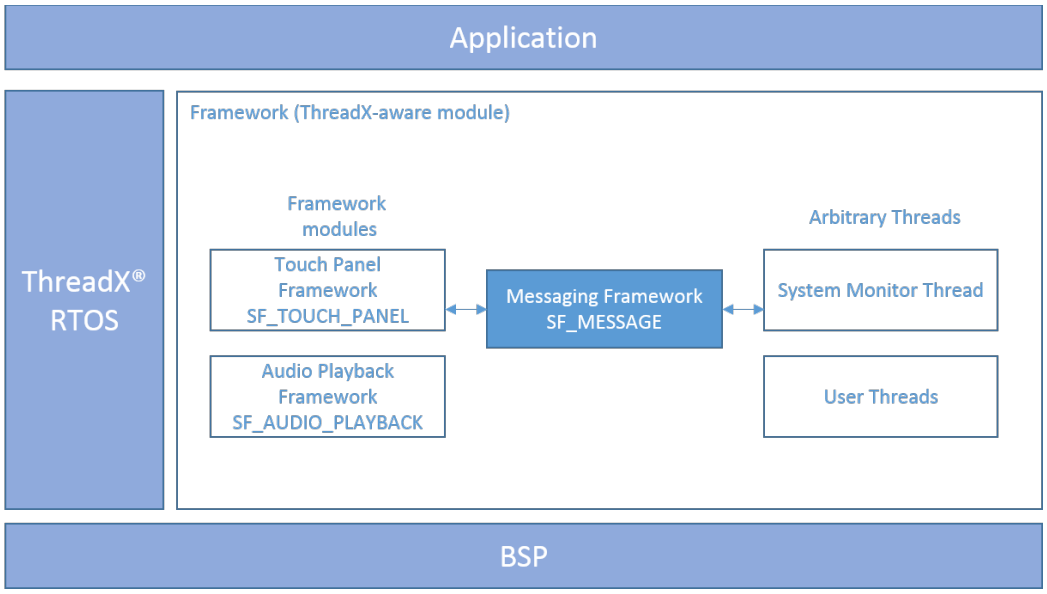


Figure 10.3 Inter-Thread Messaging Framework

10.3.2 Estimated Memory Requirements

Table 10.3 Memory Usage for Inter-Thread Messaging Framework - GCC Compiler

sf_message	Flash (Bytes)
S124 MCU Group	2,073
S3A7 MCU Group	1,981
S5D9 MCU Group	1,981
S7G2 MCU Group	1,981

Table 10.4 Memory Usage for Inter-Thread Messaging Framework – IAR Compiler

sf_message	Flash (Bytes)
S124 MCU Group	1,760
S3A7 MCU Group	1,692
S5D9 MCU Group	1,692
S7G2 MCU Group	1,692

10.4 I²C Framework

10.4.1 Component Introduction

The I²C Framework in SSP abstracts the software interface for the I²C driver. It provides a simple, high-level, C-callable API for seamless and thread-safe access of the I²C interface from multiple application threads.

The I²C framework is ThreadX® aware and handles the integration and synchronization of multiple I²C peripherals on an I²C bus. The I²C Framework enables the user to create one or more I²C buses and connect multiple I²C peripherals to the buses. The I²C Framework makes use of low-level I²C driver modules and SCI common driver modules to communicate with I²C peripherals. The I²C Framework provides Mutual Exclusion and Synchronization services to manage simultaneous multiple access requests. Internally the framework uses ThreadX® objects like mutex bus locking/unlocking for blocking, and synchronization techniques like event flags for completion of transactions. The I²C Framework supports the handling of restart condition and provides common interface for both SCI I²C and RIIC peripherals.

- The framework blocks access to a specific channel while in use with mutex.
- The framework driver handles callback to notify application of events.
- The framework maintains a counter that tracks how many devices are currently on the bus.
- The framework also provides timeout parameter to the write/read API functions.

- The framework provides lock options to lock a bus for a device and provides an unlock API to unlock the locked bus by device.
- The framework supports all channels available with the device.
- Framework supports opening of multiple devices on the same bus.

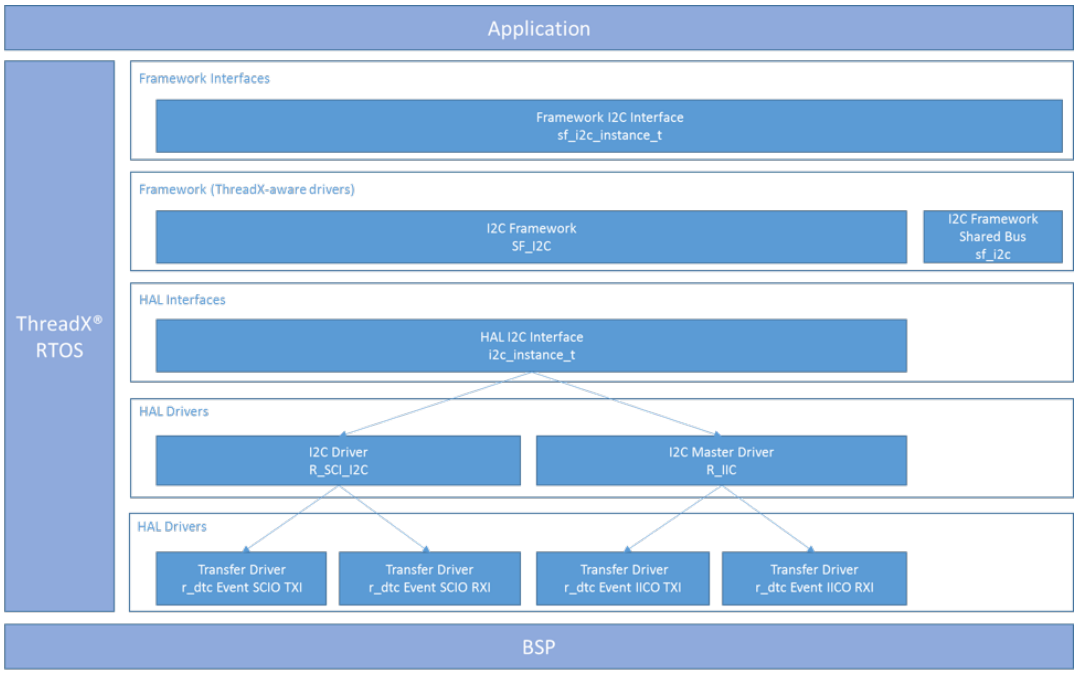


Figure 10.4 I²C Framework

10.4.2 Estimated Memory Requirements

Table 10.5 Memory Use for I2C Framework – GCC Compiler

sf_i2c	Flash (Bytes)
S3A7 MCU Group	1,485
S5D9 MCU Group	1,485
S7G2 MCU Group	1,485

Table 10.6 Memory Use for I2C Framework – IAR Compiler

sf_i2c	Flash (Bytes)
S3A7 MCU Group	1,280
S5D9 MCU Group	1,280
S7G2 MCU Group	1,280

10.5 Touch Panel I²C Framework

10.5.1 Component Introduction

The Touch Panel I²C Framework in SSP provides a high-level, C-callable programmable interface for interfacing with external touch screen controllers. Internally the Touch Panel I²C Framework uses I²C to interface with the touch screen controller. The Touch Panel I²C Framework API provides seamless and thread-safe software interface for touch screens from SSP.

The Touch Panel I²C Framework is commonly used for capturing touch input for GUI applications. It produces touch data messages with position information (X and Y coordinates) and event type information which are posted to event queue(s) using the Messaging Framework. The touch panel framework also creates a thread to poll the touch driver and post touch data to the Messaging Framework.

The Touch Panel I²C Framework sequentially processes UI input events in the order they are received from the I²C interface. The framework supports receiving input events from multiple touch screens. Internally the framework uses

mutex for synchronization between multiple application threads. The framework uses external interrupt interface for synchronization.

The API has provisions for:

- Specifying the display resolution.
- Limiting the rate of touch messages published for positioning information.
- Interrupt driven I²C touch controller chips.
- Configuring minimum period between touch messages for repeat events (hold/move).
- A calibrate function for ADC interface calibration.
- An option to start and stop touch processing. When stopped the Framework terminates all underlying low level drivers used by the touch panel framework. Additionally the framework provides a reset function for runtime reset.

The touch panel framework supports touch controllers available on all Renesas Product Examples and Development Kits.

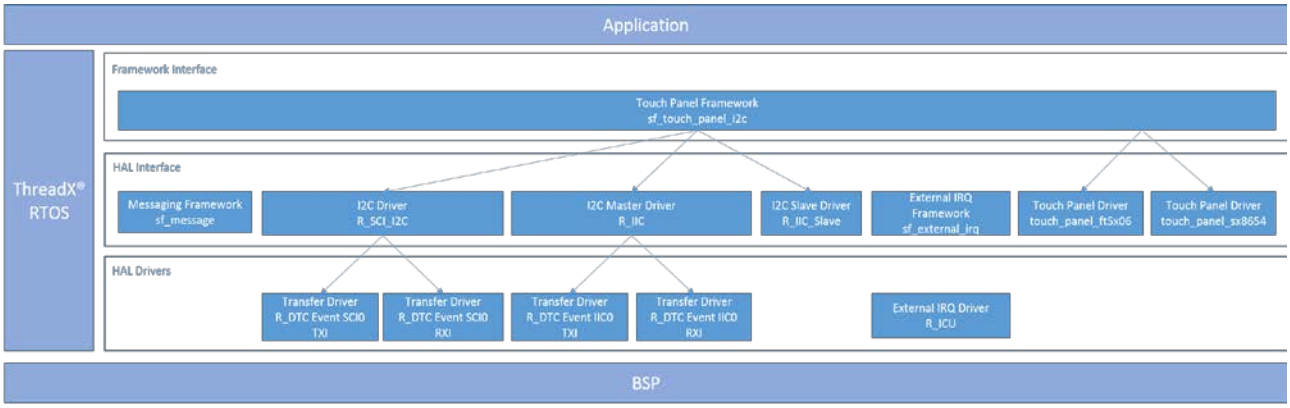


Figure 10.5 Touch Panel I²C Framework

10.5.2 Estimated Memory Requirements

Table 10.7 Memory Usage for Touch Panel I2C Framework – GCC Compiler

sf_touch_panel_i2c	Flash (Bytes)
S5D9 MCU Group	1,526
S7G2 MCU Group	1,526

Table 10.8 Memory Usage for Touch Panel I2C Framework – IAR Compiler

sf_touch_panel_i2c	Flash (Bytes)
S5D9 MCU Group	1,428
S7G2 MCU Group	1,428

10.6 External Interrupt Framework

10.6.1 Component Introduction

The External Interrupt Framework provides a high-level, C-callable interface for scheduling event driven execution of threads in the SSP. When the IRQ is raised by applications, it causes a thread to pend until an external pin interrupt is received or a timeout occurs. Common use cases include waiting for a switch press from a user or waiting for a signal from an external hardware device.

The interrupt framework is ThreadX® aware and provides generic external interrupt handling capability. The APIs for Interrupt Framework are thread-safe and internally the framework uses ThreadX® objects like mutex for blocking, and synchronization techniques like semaphores for interrupt handling by multiple application threads. Some of the key features include:

- Supports unique pending IRQ requests for multiple threads.
- Supports IRQ requests for up to 16 hardware channels.
- Provides programmable timeout for wait function through the APIs.

- Thread can be suspended while waiting for external IRQ request.
- Provides provision to specify timeouts.
- Configures the driver to block access to specific low level external IRQ when it is being accessed using mutex.
- Provides API which waits for an external IRQ to be triggered.
- Makes use of lower level HAL driver for interfacing to the external IRQ hardware.

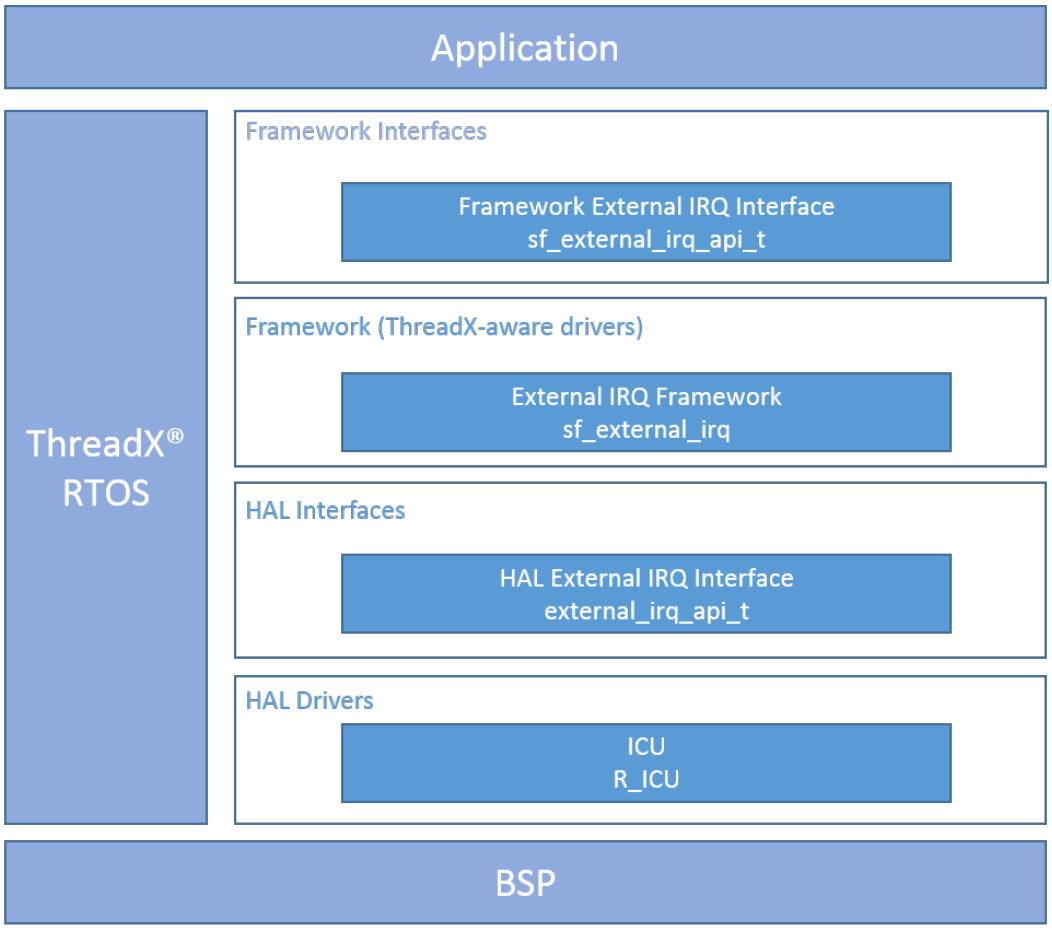


Figure 10.6 External Interrupt Framework

10.6.2 Estimated Memory Requirements

Table 10.9 Memory Usage for External Interrupt Framework – GCC Compiler

sf_external_irq	Flash (Bytes)
S124 MCU Group	772
S3A7 MCU Group	764
S5D9 MCU Group	764
S7G2 MCU Group	764

Table 10.10 Memory Usage for External Interrupt Framework – IAR Compiler

sf_external_irq	Flash (Bytes)
S124 MCU Group	656
S3A7 MCU Group	636
S5D9 MCU Group	636
S7G2 MCU Group	636

10.7 JPEG Decode Framework

10.7.1 Component Introduction

The JPEG Decode Framework module in SSP abstracts the on-chip JPEG codec and provides a simple high-level, C-callable API for seamless and device independent integration of JPEG decoder application threads. The JPEG codec allows for high-speed compression of raw images and decoding of JPEG images. The codec conforms to the JPEG baseline compression and decompression standard, JPEG Part 2, ISO-IEC 10918-2.

The JPEG Decode Framework is ThreadX® aware and provides primary JPEG decoder functionality. The JPEG APIs are thread-safe and internally the framework uses ThreadX® objects like mutex for blocking, and synchronization techniques, like event flags for completion of JPEG data decompression by multiple application threads.

The JPEG Decode Framework APIs handles the decode tasks by taking application specific encoded data in an input buffer and allocating an output buffer pointer to store the decoded image frame. Alternatively, the API can handle streaming encoded data into JPEG decoder module. This feature allows an application to read encoded JPEG image from a file or from network without buffering the entire image. The framework allows the application to specify the number of image lines to decode so that the application can decode the image on the fly without buffering the entire frame.

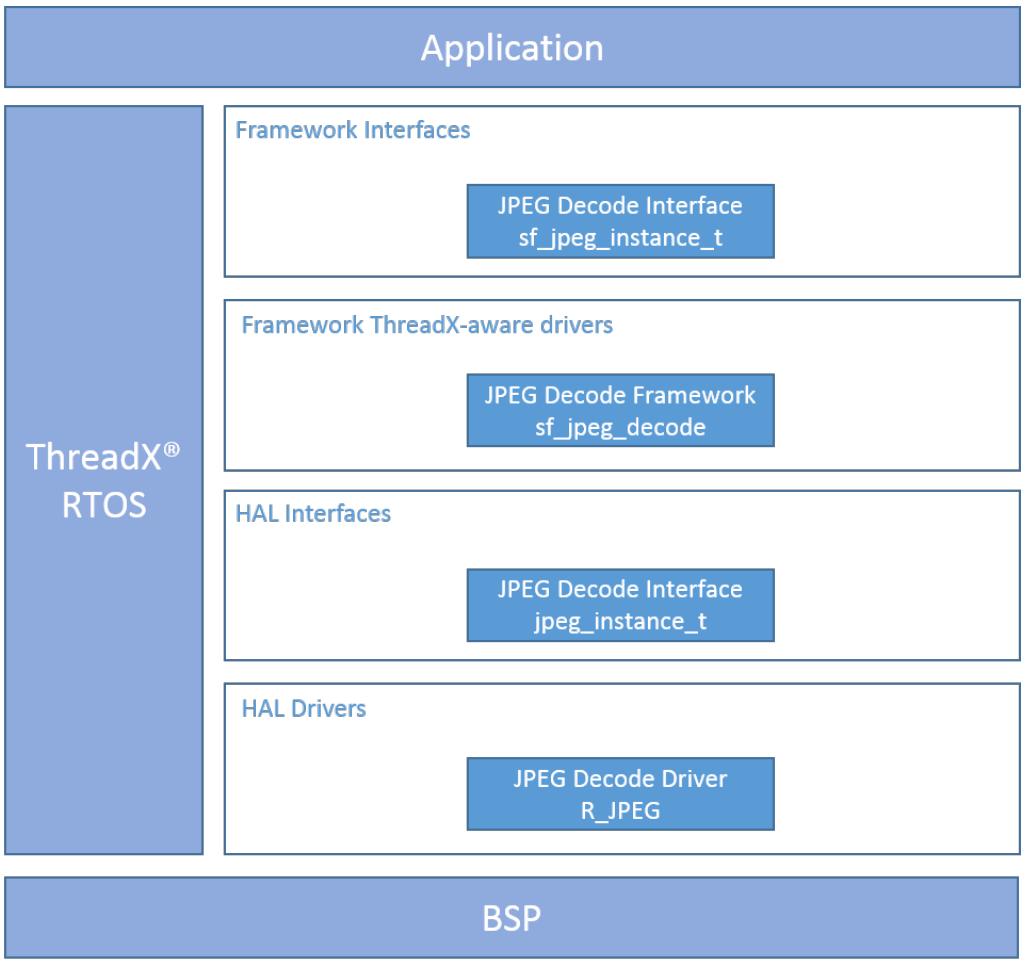


Figure 10.7 JPEG Decode Framework

10.7.2 Estimated Memory Requirements

Table 10.11 Memory Usage for JPEG Decode Framework – GCC Compiler

sf_jpeg_decode	Flash (Bytes)
S5D9 MCU Group	1,888
S7G2 MCU Group	1,888

Table 10.12 Memory Usage for JPEG Decode Framework – IAR Compiler

sf_jpeg_decode	Flash (Bytes)
S5D9 MCU Group	1,416
S7G2 MCU Group	1,416

10.8 UART Framework

10.8.1 Component Introduction

The UART Framework module in SSP abstracts the serial communication peripherals and provides a simple high-level, C-callable API for seamless and device independent integration of serial ports from multiple application threads.

The UART Framework is ThreadX® aware and provides generic full-duplex UART communication. Internally the framework uses ThreadX® objects like mutex for blocking and synchronization techniques like event flags to manage simultaneous multiple access requests. The SSP UART driver designates the HAL driver to call the framework's UART callback function and handles events generated by UART hardware. The UART Framework module can be implemented by several hardware peripherals at the HAL layer. The connection to the HAL layer is established by passing in a driver structure at initialization time. Both SCI and USBX UART modules are supported in this version.

The UART Framework in Synergy implements the Synergy Communications Interface.

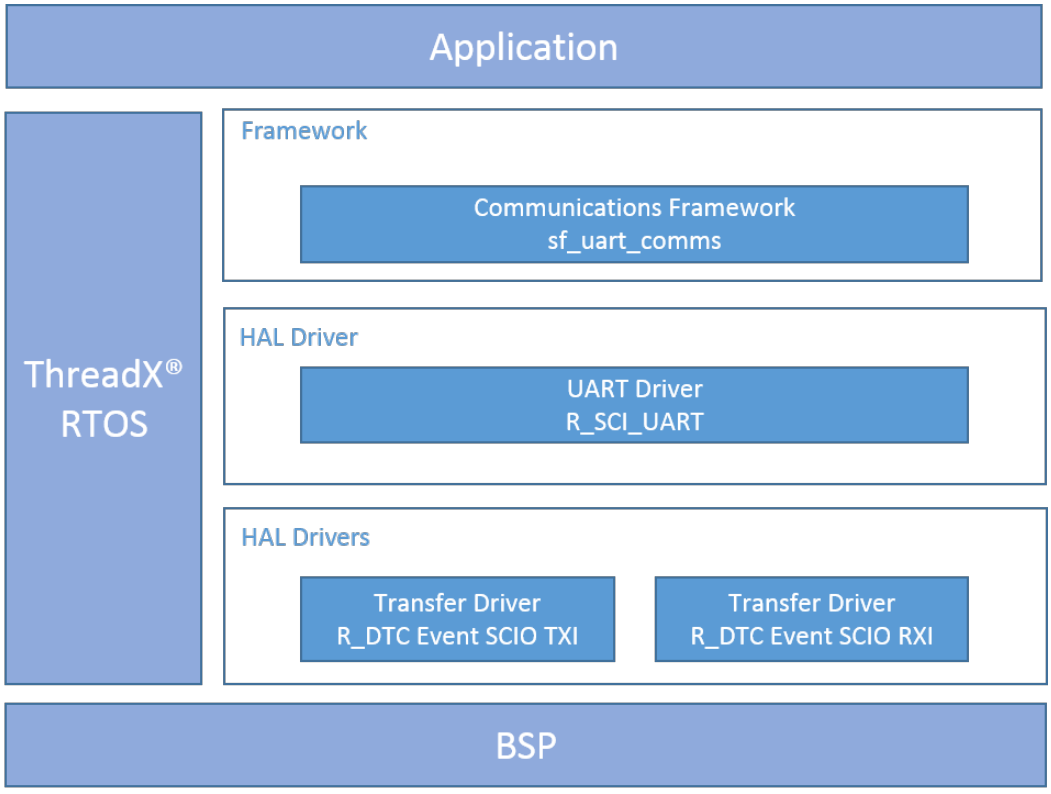


Figure 10.8 UART Framework

10.8.2 Estimated Memory Requirements

Table 10.13 Memory Usage for UART Framework – GCC Compiler

sf_uart_comms	Flash (Bytes)
S124 MCU Group	1,688
S3A7 MCU Group	1,700
S5D9 MCU Group	1,700
S7G2 MCU Group	1,700

Table 10.14 Memory Usage for UART Framework – IAR Compiler

sf_uart_comms	Flash (Bytes)
S124 MCU Group	1,420
S3A7 MCU Group	1,408
S5D9 MCU Group	1,408
S7G2 MCU Group	1,408

10.9 Console Framework

10.9.1 Component Introduction

The Console Framework module in SSP provides easy to use high-level, C-callable APIs for implementing a CLI (command line interface). The framework defines command names and callback function for each command, and uses the Communications Framework to receive commands and input strings, parse the content, and invokes the relevant command handler routine.

The Console Framework can handle inputs from other serial interfaces as well, for example, USB CDC. The parser within the framework provides support for nested menus, standard commands to return to the root menu ("~") and to back out to the previous menu (". ."). It also supports arrow key input, backspace, and delete keys.

The Console Framework supports hierarchical menus and parsing of input command based on predefined command list, and provides notification when a command is selected and provides an error code if the command isn't found.

The Console Framework supports:

- Input without parsing
- Numerical input
- Echo option to echo input to transmitter
- Backspace and arrow key navigation
- UART driver and other serial drivers

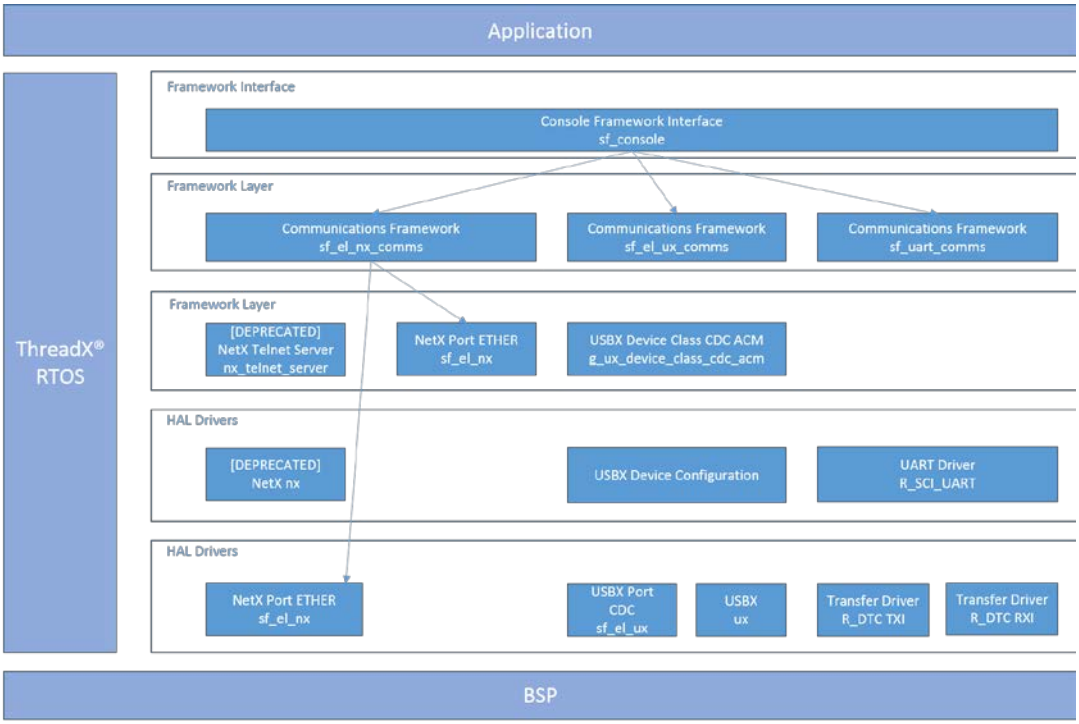


Figure 10.9 Console Framework

10.9.2 Estimated Memory Requirements

Table 10.15 Memory Usage for Console Framework – GCC Compiler

sf_console	Flash (Bytes)
S124 MCU Group	2,794
S3A7 MCU Group	2,750
S5D9 MCU Group	2,750
S7G2 MCU Group	2,750

Table 10.16 Memory Usage for Console Framework – IAR Compiler

sf_console	Flash (Bytes)
S124 MCU Group	2,496
S3A7 MCU Group	2,516
S5D9 MCU Group	2,516
S7G2 MCU Group	2,516

10.10 Thread Monitor Framework

10.10.1 Component Introduction

The Thread Monitor Framework monitors RTOS threads using a Watchdog Timer (WDT). The Thread Monitor forces a watchdog reset of the microcontroller when any of the monitored threads misbehave.

The Thread Monitor operates as follows:

A thread registers a counter variable with the Thread Monitor Framework along with minimum and maximum expected values for this counter variable. The thread which is monitored increments the counter variable while it runs. At a period of half the watchdog timeout period, the Thread Monitor checks the counter variables of registered threads. If any fall outside of the minimum and maximum values, the Watchdog Timer is allowed to reset the microcontroller.

If the counter variables fall within their expected range, the Watchdog Timer is refreshed and the counter values are cleared to zero. In profiling mode, the minimum and maximum counter values for registered threads can be determined. In profiling mode, the (WDT) is always refreshed and therefore does not reset the device. The framework supports both the Watchdog Timer and the Independent Watchdog Timer (IWDT) HAL modules.

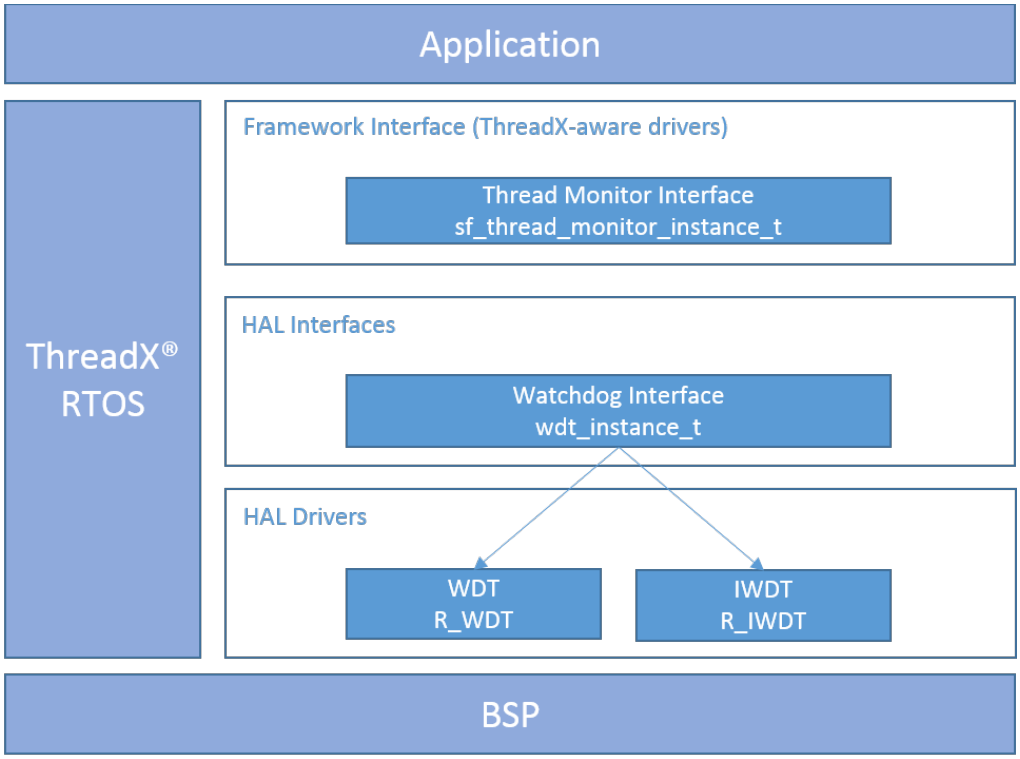


Figure 10.10 Thread Monitor Framework

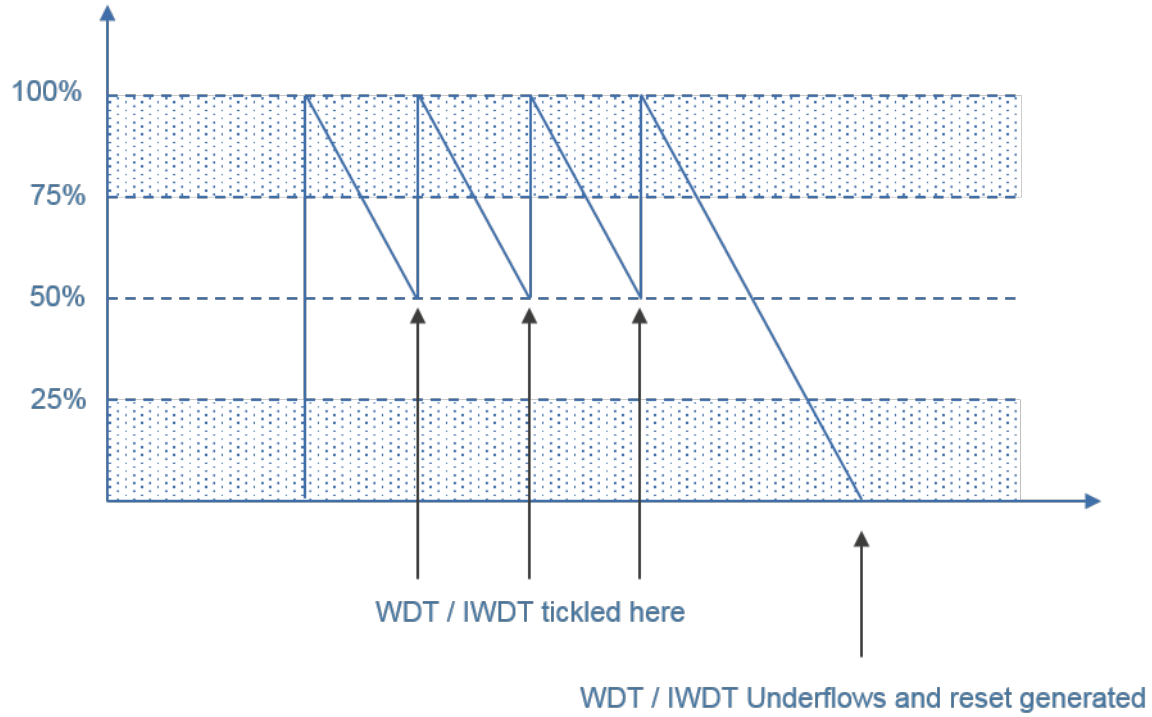


Figure 10.11 Thread Monitor Timing Chart

10.10.2 Estimated Memory Requirements

Table 10.17 Memory Usage for Thread Monitor Framework – GCC Compiler

sf_thread_monitor	Flash (Bytes)
S3A7 MCU Group	1,413
S5D9 MCU Group	1,413
S7G2 MCU Group	1,413

Table 10.18 Memory Usage for Thread Monitor Framework – IAR Compiler

sf_thread_monitor	Flash (Bytes)
S3A7 MCU Group	1,276
S5D9 MCU Group	1,276
S7G2 MCU Group	1,276

10.11 ADC Periodic Framework

10.11.1 Component Introduction

The ADC Periodic Framework Interface samples and buffers ADC data. The Framework notifies the application once the configured number of samples are buffered.

The Periodic Sampling ADC Framework provides C-callable, generic and thread-safe APIs for applications to sample data over available ADC channels. Key features of the Framework include:

- Configurable sampling rate and iterations.
- Samples and buffers data from ADC channels.
- Notifies applications when the configured number of samples are ready.
- Uses callback mechanism to notify availability of data.
- Framework uses GPT or AGT timer interface for timing functions.
- Framework uses DMA or DTC for efficient transfer of data from framework to application.

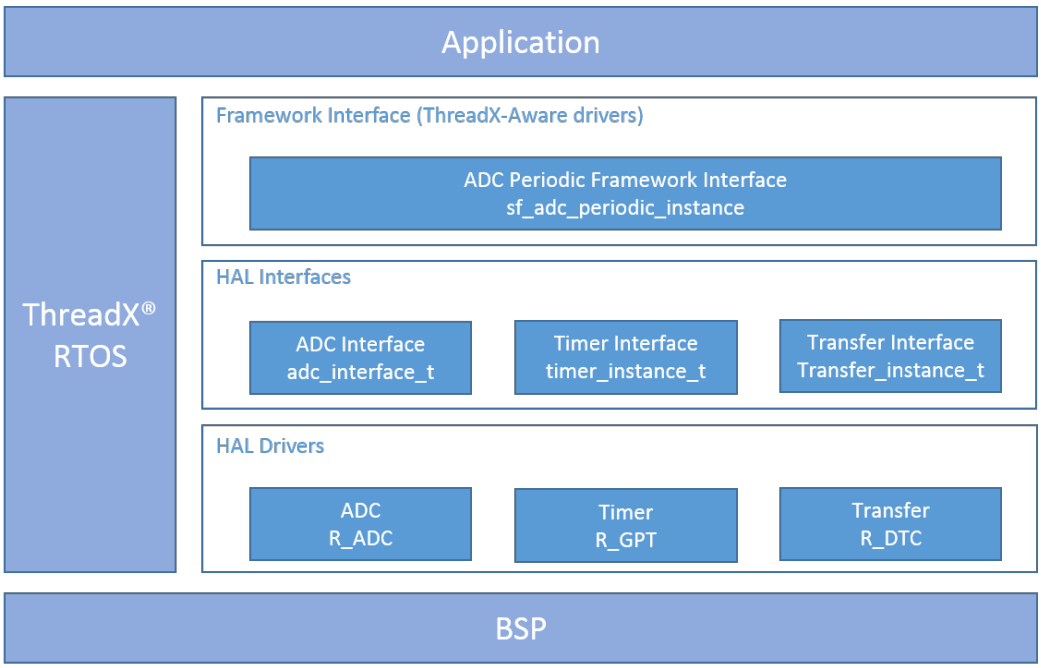


Figure 10.12 Periodic Sampling ADC Framework

10.11.2 Estimated Memory Requirements

Table 10.19 Memory Usage for Periodic Sampling ADC Framework – GCC Compiler

sf_adc_periodic	Flash (Bytes)
S124 MCU Group	1,468
S3A7 MCU Group	1,468
S5D9 MCU Group	1,468
S7G2 MCU Group	1,468

Table 10.20 Memory Usage for Periodic Sampling ADC Framework – IAR Compiler

sf_adc_periodic	Flash (Bytes)
S124 MCU Group	1,224
S3A7 MCU Group	1,224
S5D9 MCU Group	1,224
S7G2 MCU Group	1,224

10.12 Audio Playback HW DAC Framework

10.12.1 Component Introduction

The Audio Playback HW DAC Framework handles the integration and synchronization of multiple HAL peripherals like timers, DMA, and DAC to facilitate audio playback. This light-weight framework provides basic audio playback functionality and can be used for short tones or chirps. For advanced audio applications, this framework is typically used with the Audio Playback Framework.

Audio Playback HW DAC Framework features include:

- Plays a single buffer of pre-scaled 12-bit unsigned PCM audio samples.
- Provides information about the required data type (12-bit unsigned).
- Provides callback mechanism to notify application when buffer has finished playing.
- Configures the audio hardware based on application settings.
- 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.
- Scaling to move signed 16-bit PCM data into range of the unsigned 12-bit DAC.
- Basic mixing for multiple streams.

The framework supports playing back of multiple streams on a single hardware port.

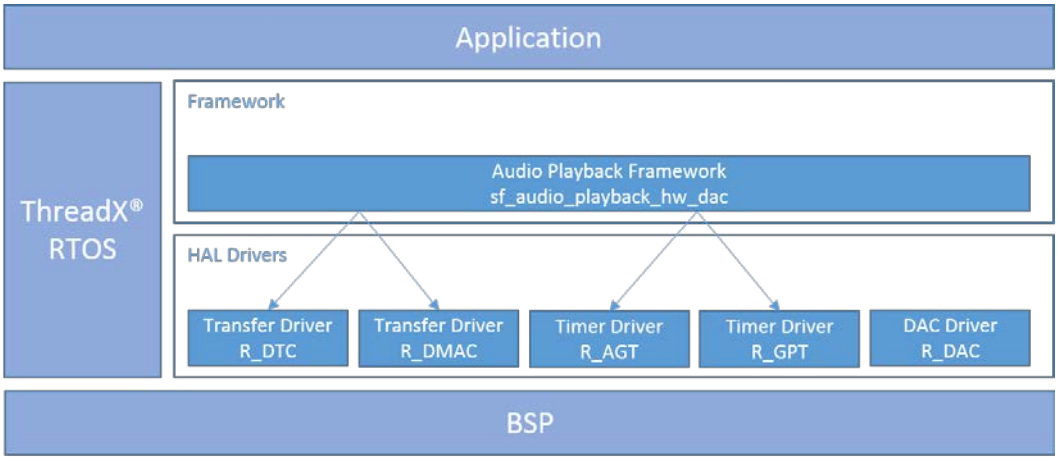


Figure 10.13 Audio Playback HW DAC Framework

10.12.2 Estimated Memory Requirements

Table 10.21 Memory Usage for Audio Playback HW DAC Framework – GCC Compiler

sf_audio_playback_hw_dac	Flash (Bytes)
S124 MCU Group	1,355
S3A7 MCU Group	1,423
S5D9 MCU Group	1,423
S7G2 MCU Group	1,423

Table 10.22 Memory Usage for Audio Playback HW DAC Framework – IAR Compiler

sf_audio_playback_hw_dac	Flash (Bytes)
S124 MCU Group	968
S3A7 MCU Group	1,068
S5D9 MCU Group	1,068
S7G2 MCU Group	1,068

10.13 Audio Playback HW I²S Framework

10.13.1 Component Introduction

The Audio Playback HW I²S Framework handles the integration and synchronization of multiple HAL peripherals like timers, DMA, and I²S to facilitate audio playback. This light-weight framework provides basic audio playback functionality and can be used for short tones or chirps. For advanced audio applications, this framework is typically used with the Audio Playback Framework.

Audio Playback HW I²S Framework features include:

- Plays a single buffer of pre-scaled 12 bit unsigned PCM audio samples.
- Provides information about the required data type (12 bit unsigned).
- Provides callback mechanism to notify application when buffer has finished playing.
- Configures the audio hardware based on application settings
- 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.
- Scaling to move signed 16-bit PCM data into range of the unsigned 12-bit DAC.
- Basic mixing for multiple streams.

The framework supports playing back of multiple streams on a single hardware port.

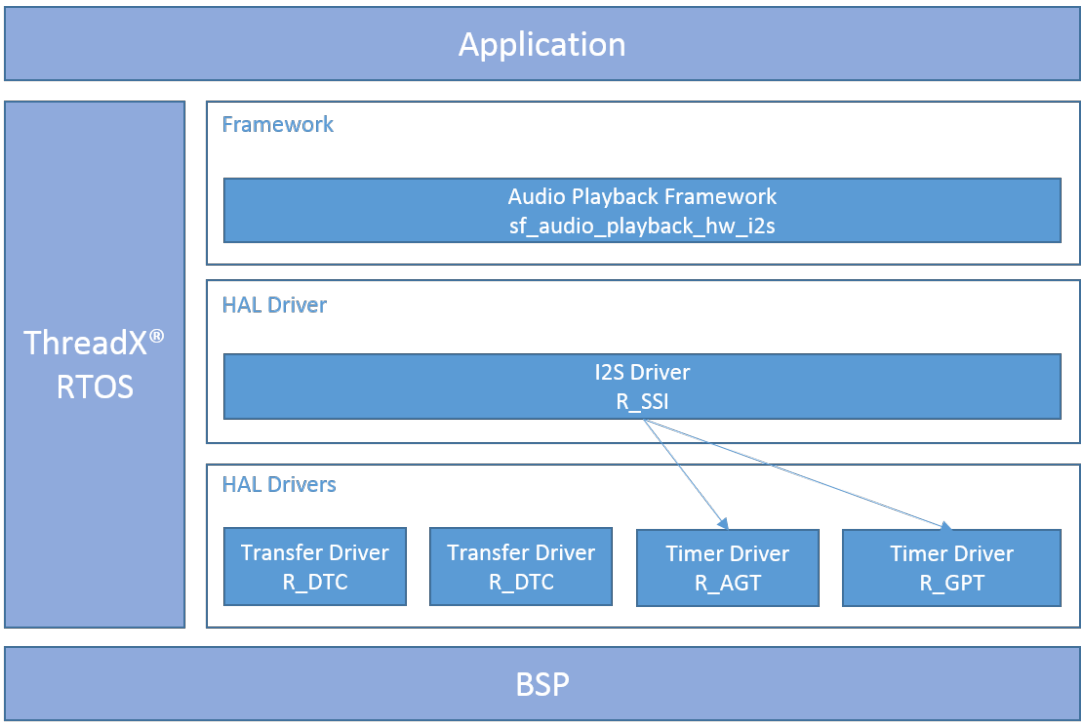


Figure 10.14 Audio Playback HW I2S Framework

10.13.2 Estimated Memory Requirements

Table 10.23 Memory Usage for Audio Playback HW I2S Framework – GCC Compiler

sf_audio_playback_hw_i2s	Flash (Bytes)
S124 MCU Group	631
S3A7 MCU Group	631
S5D9 MCU Group	631
S7G2 MCU Group	631

Table 10.24 Memory Usage for Audio Playback HW I2S Framework – IAR Compiler

sf_audio_playback_hw_i2s	Flash (Bytes)
S3A7 MCU Group	468
S5D9 MCU Group	468
S7G2 MCU Group	468

10.14 Audio Record HW ADC Framework

10.14.1 Component Introduction

The Audio Record ADC Framework records audio Pulse-code modulation samples using the ADC Periodic Framework. You can configure audio recording parameters via the Audio Record ADC Framework, providing you with the recorded or captured data for further processing.

Audio Record HW Framework features include:

- The Audio Record ADC framework provides system services for applications to record audio data through the ADC peripheral on Synergy MCUs
- Framework APIs are thread-safe and independent of underlying MCU hardware features
- Audio record ADC framework provides standardized, C-callable, high-level APIs
- Framework manages the integration and synchronization between multiple hardware peripherals like timers, DMA, and ADC to facilitate audio record
- The audio record framework supports 16-bit Mono & Stereo uncompressed (linear) PCM samples.
- The audio record ADC framework supports the following controls:
  - Start record
  - Stop record
  - Pause record
  - Resume record
- The framework does not introduce very large delays in the record
- It supports recording unsigned 12-bit data format.

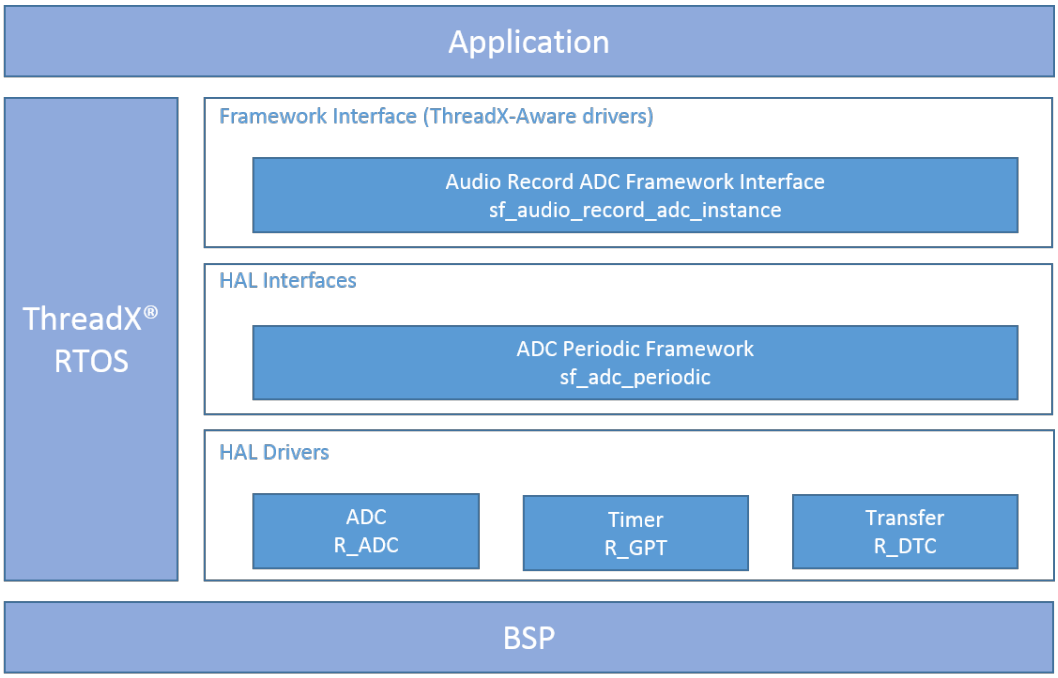


Figure 10.15 Audio Record HW ADC Framework

Table 10.25 Memory Usage for Audio Record HW ADC Framework – GCC Compiler

sf_audio_record_adc	Flash (Bytes)
S3A7 MCU Group	744
S5D9 MCU Group	744
S7G2 MCU Group	744

Table 10.26 Memory Usage for Audio Record HW ADC Framework – IAR Compiler

sf_audio_record_adc	Flash (Bytes)
S3A7 MCU Group	532
S5D9 MCU Group	532
S7G2 MCU Group	532

10.15 Capacitive Touch Sensing Unit (CTSU) Framework

10.15.1 Component Introduction

The Capacitive Touch Sensing Unit Framework module in SSP provides a hardware agnostic, high-level, abstracted interface for capacitive touch user interface elements like button, slider and wheel. The Framework uses the Capacitive Touch Sensing HAL driver to provide a thread-safe and hardware agnostic system services for Capacitive Touch applications using ThreadX® RTOS. The CTSU Framework provides simple, generic and high-level, C-callable APIs.

The CTSU Framework creates a thread which drives a hardware scan of a capacitive touch panel and updates the panel at a periodic rate. The Framework Module reads the scanned results using the HAL Layer CTSU driver.

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

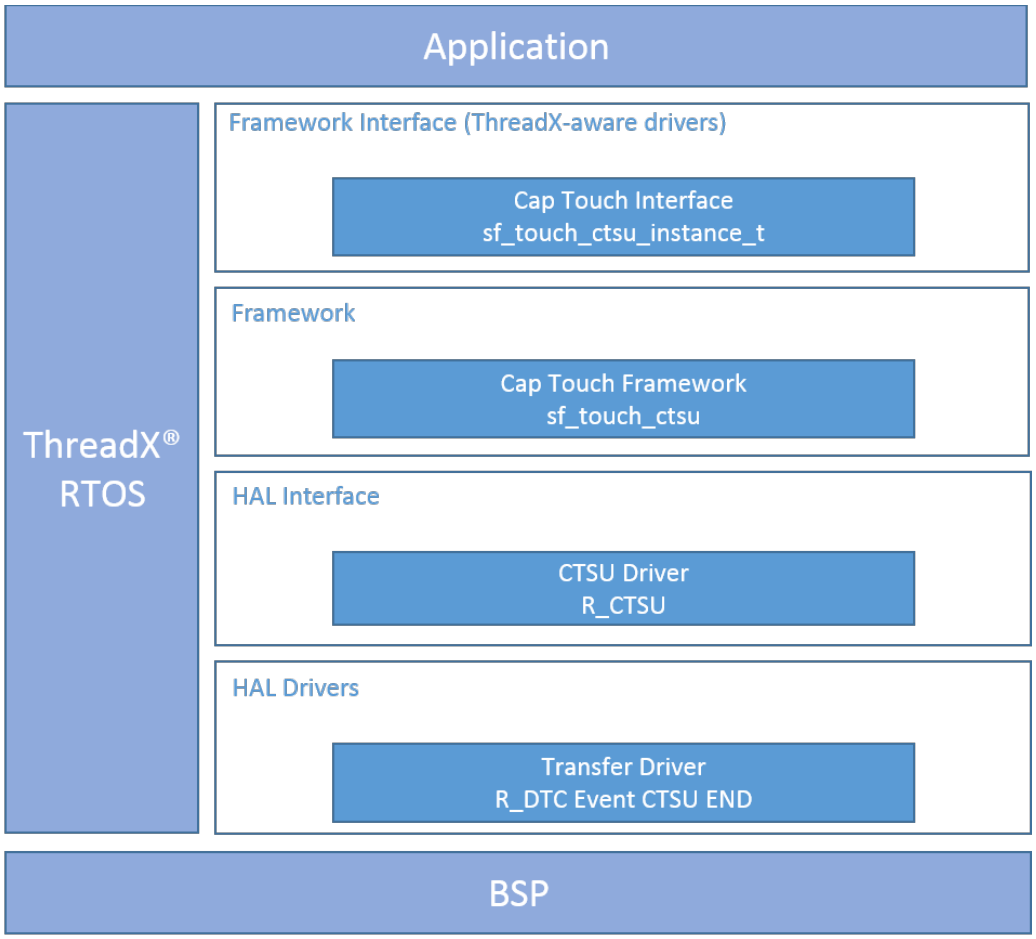


Figure 10.16 Capacitive Touch Sensing Unit Framework

10.15.2 Estimated Memory Requirements

Table 10.27 Memory Usage for Capacitive Touch Sensing Unit Framework – GCC Compiler

sf_touch_ctsu	Flash (Bytes)
S124 MCU Group	4829
S3A7 MCU Group	4,829
S5D9 MCU Group	4,829
S7G2 MCU Group	4,829

Table 10.28 Memory Usage for Capacitive Touch Sensing Unit Framework – IAR Compiler

sf_touch_ctsu_slider	Flash (Bytes)
S124 MCU Group	2,724
S3A7 MCU Group	2,816
S5D9 MCU Group	2,816
S7G2 MCU Group	2,816

10.16 Capacitive Touch Sensing Unit Button Framework

10.16.1 Component Introduction

The CTSU Button Framework provides simple, consistent and generic C-callable APIs for applications to add capacitive touch button interfaces to their user interfaces. The Button Framework is thread-safe and utilizes the CTSU Framework and the CTSU HAL driver modules in SSP.

The CTSU Button Framework interprets the data received from the CTSU Framework for all buttons that are present in the system. It also initializes the CTSU Framework layer and registers a callback with the CTSU Framework layer which will be called each time processed data is available. The CTSU Button Framework then uses this processed data to perform de-bouncing and determine which of the configured events (Press, Release, Long Touch, etc.) has occurred for each button and calls the callback for each button in the order in which they are present in the button configuration table.

CTSU Button Framework supports the following gestures for the button interface:

- Pressed State
- Released State
- Long Touch
- Short Touch
- Multi Touch
- Button is stuck in pressed state

The framework is designed to be used together with configuration data generated by the Capacitive Touch Workbench for Renesas Synergy tool, available on the Gallery.

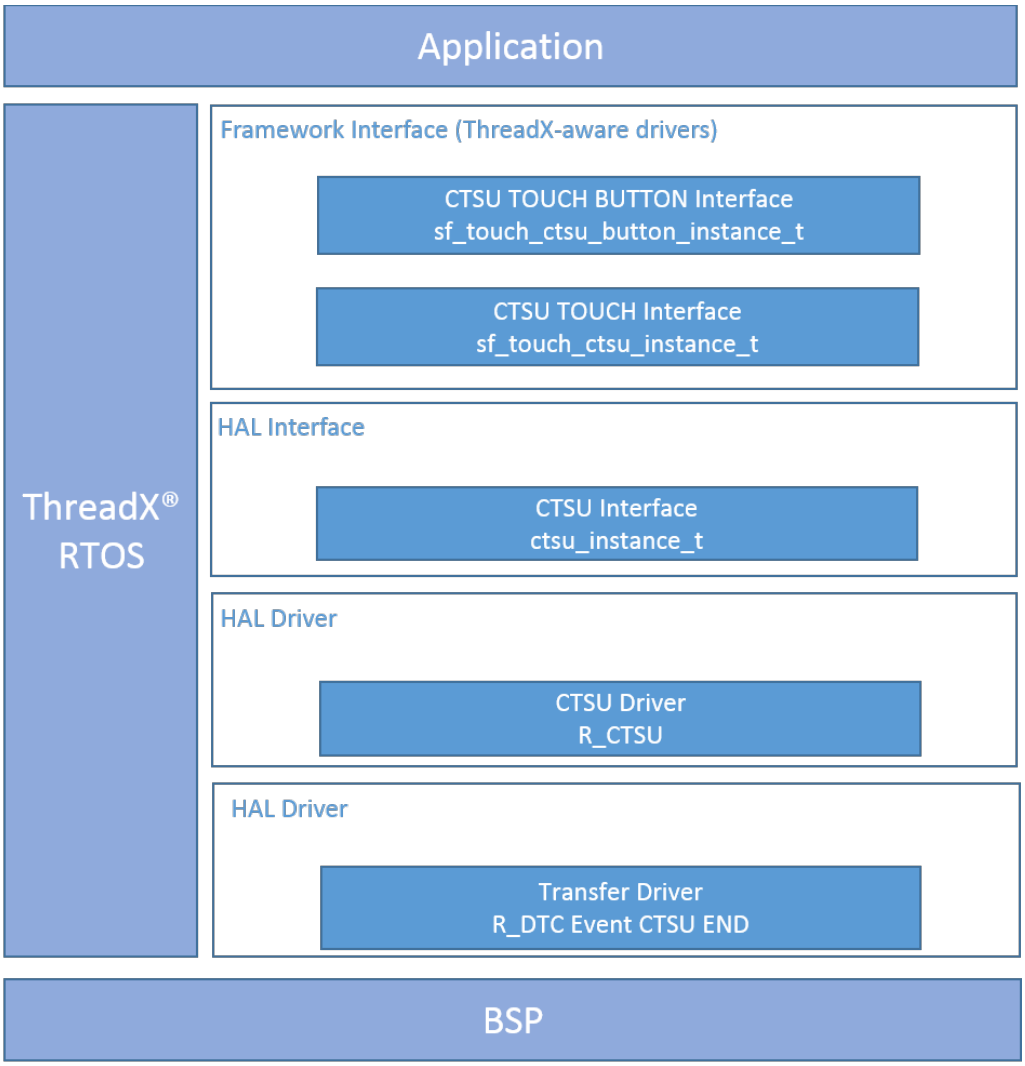


Figure 10.17 Capacitive Touch Sensing Unit Button Framework

10.16.2 Estimated Memory Requirements

Table 10.29 Memory Usage for Capacitive Touch Sensing Unit Button Framework – GCC Compiler

sf_touch_ctsu_button	Flash (Bytes)
S124 MCU Group	2,327
S3A7 MCU Group	2,423
S5D9 MCU Group	2,423
S7G2 MCU Group	2,423

Table 10.30 Memory Usage for Capacitive Touch Sensing Unit Button Framework – IAR Compiler

sf_touch_ctsu_button	Flash (Bytes)
S124 MCU Group	4,632
S3A7 MCU Group	4,576
S5D9 MCU Group	4,576
S7G2 MCU Group	4,576

10.17 Capacitive Touch Sensing Unit Slider Framework

10.17.1 Component Introduction

The CTSU Slider Framework provides simple, consistent and generic C-callable APIs for applications to add capacitive touch slider and wheel interfaces to their user interfaces. The Slider Framework is thread-safe and utilizes the CTSU Framework and the CTSU HAL driver modules in SSP.

The CTSU Slider Framework interprets the data received from CTSU Framework for all slider configurations initialized by the system. The Slider Framework uses this data to determine if a touch or release occurred and, if so, where it occurred. If there's a state change, the Framework calls the callback for each slider, in the order in which they are present in the slider configuration table, with the event and the position. The slider framework executes the callback at the update rate between the touch and release events. Applications can use these callbacks to track the position along the slider.

The slider also supports multi-touch detection, where the user application gets a callback with a multi-touch event if any widgets other than the slider are touched at the same time as the slider. This feature can be optionally disabled at build-time.

The framework is designed to be used together with configuration data generated by the Capacitive Touch Workbench for Renesas Synergy tool.

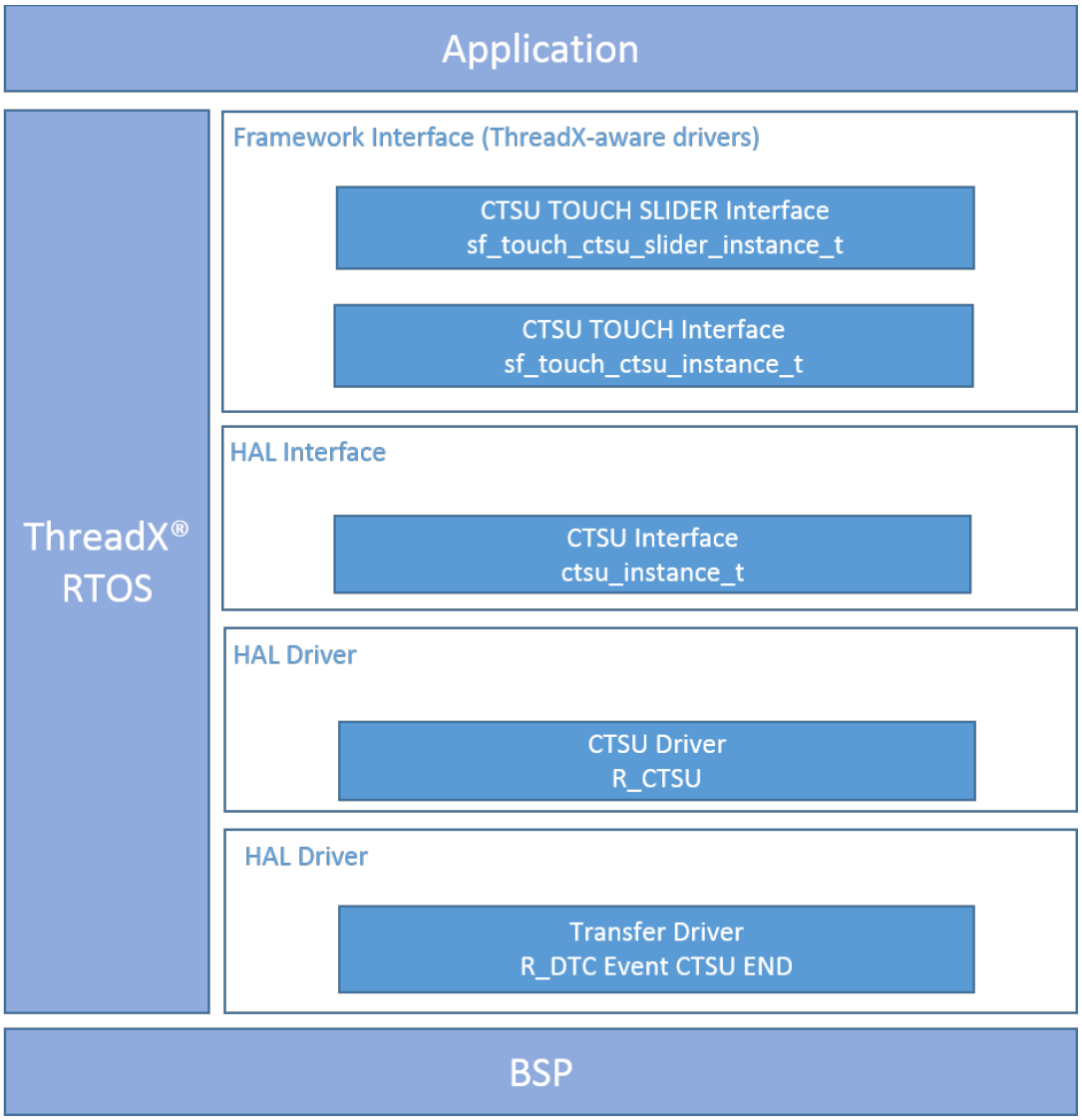


Figure 10.18 Capacitive Touch Sensing Unit Slider Framework

10.17.2 Estimated Memory Requirements

Table 10.31 Memory Usage for Capacitive Touch Sensing Unit Slider Framework – GCC Compiler

sf_touch_ctsu_slider	Flash (Bytes)
S124 MCU Group	3,603
S3A7 MCU Group	3,451
S5D9 MCU Group	3,451
S7G2 MCU Group	3,451

Table 10.32 Memory Usage for Capacitive Touch Sensing Unit Slider Framework – IAR Compiler

sf_touch_ctsu_slider	Flash (Bytes)
S124 MCU Group	2,724
S3A7 MCU Group	2,816
S5D9 MCU Group	2,816
S7G2 MCU Group	2,816

10.18 Serial Peripheral Interface (SPI) Framework

10.18.1 Component Introduction

The SPI Framework module in SSP handles the integration and synchronization of multiple SPI peripherals on an SPI bus. The SPI Framework provides simple, high-level, C-callable APIs for SPI interfaces that can be used to create one or more SPI buses and connect multiple peripherals to the SPI bus.

The SPI Framework driver complies with the layered driver architecture of the SSP. It uses either the SCI in SPI mode together with the SCI common lower-level driver modules or the RSPI lower-level driver module to communicate with the SPI peripherals on the Synergy microcontroller.

The SPI Framework is ThreadX® aware and provides common framework for SPI interfaces. The Framework integrates with existing SPI driver interfaces like SCI SPI and supports:

- Single bus or multiple buses
- Connecting multiple slave devices to a single bus
- Bus locking for a device for a given amount of time
- Operating without a manual chip select control
- Configuring of bus, device and low level drivers through ISDE

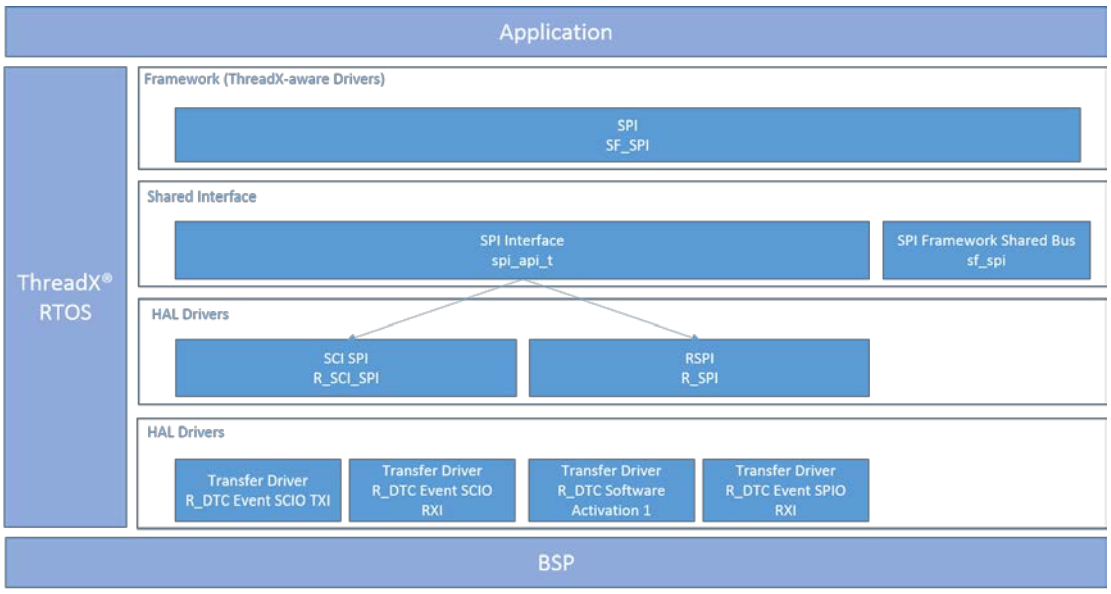


Figure 10.19 **Serial Peripheral Interface (SPI) Framework**

10.18.2 Estimated Memory Requirements

Table 10.33 Memory Usage for Serial Peripheral Interface (SPI) Framework – GCC Compiler

sf_spi	Flash (Bytes)
S124 MCU Group	1,843
S3A7 MCU Group	1,851
S5D9 MCU Group	1,851
S7G2 MCU Group	1,851

Table 10.34 Memory Usage for Serial Peripheral Interface (SPI) Framework – IAR Compiler

sf_spi	Flash (Bytes)
S124 MCU Group	1,756
S3A7 MCU Group	1,796
S5D9 MCU Group	1,796
S7G2 MCU Group	1,796

10.19 Power Mode Profile Framework

10.19.1 Component Introduction

The Power Profiles framework provides pre-configured power states for the MCU to be placed in lower power Software Standby mode. The Framework provides high-level, C-callable APIs which can be used to configure the MCU power state and place it in lower power Software Standby mode. Power profiles can be used with ThreadX® RTOS applications and RTOS-independent HAL level applications

The module can be configured at run-time in one of three operating modes:

- Run
- RTC
- External Interrupt

These modes determine which clocks and peripherals are disabled during Software Standby mode, as well as what the output pin states are prior to and after exiting Software Standby mode.

The Interface uses the RTC, LPM, IOPORT and CGC peripherals on the Synergy microcontroller hardware and provides an easy-to-use software interface to access the low power operating modes.

Currently supported Power Profiles:

- Software Standby
- Wakeup

10.19.2 Estimated Memory Requirements

Table 10.35 Memory Usage for Power Mode Profile Framework – GCC Compiler

sf_power_profiles	Flash (Bytes)
S124 MCU Group	1,338
S3A7 MCU Group	1,310
S5D9 MCU Group	1,306
S7G2 MCU Group	1,306

Table 10.36 Memory Usage for Power Mode Profile Framework – IAR Compiler

sf_power_profiles	Flash (Bytes)
S124 MCU Group	1,176
S3A7 MCU Group	1,184
S5D9 MCU Group	1,184
S7G2 MCU Group	1,184

10.20 Synergy FileX® Interface Framework

10.20.1 Component Introduction

Synergy FileX® Interface Framework provides an adaptation layer for integrating block media device drivers with FileX®. The framework provides I/O calls for FileX® to access Synergy Media drivers through the Block Media Interface and adaptation layers.

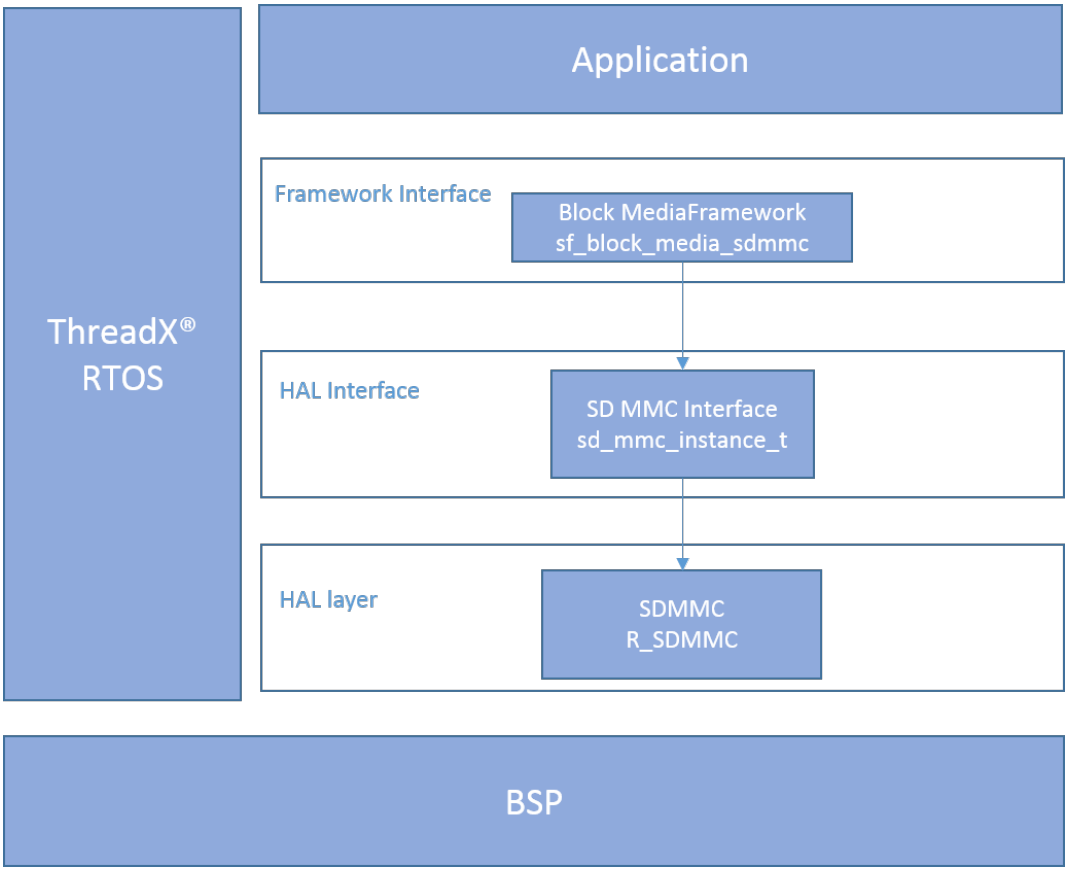


Figure 10.20 Synergy FileX® Interface Framework

10.20.2 Estimated Memory Requirements

Table 10.37 Memory Usage for Synergy FileX® Interface Framework – GCC Compiler

sf_el_fx	Flash (Bytes)
S124 MCU Group	416
S3A7 MCU Group	416
S5D9 MCU Group	416
S7G2 MCU Group	416

Table 10.38 Memory Usage for Synergy FileX® Interface Framework – IAR Compiler

sf_el_fx	Flash (Bytes)
S124 MCU Group	348
S3A7 MCU Group	348
S5D9 MCU Group	348
S7G2 MCU Group	348

10.21 Synergy GUIX™ Interface Framework

10.21.1 Component Introduction

The GUIX™ Framework module ties Synergy graphics device drivers to GUIX™ though the GUIX™ Display Drivers interface. The module uses ThreadX® service calls for mutual exclusion of device access and for timing synchronization between rendering and displaying operation of image data for graphics.

The module uses RTOS aware device drivers for 2DG and JPEG modules and the Display HAL device driver (typically the GLCDC module). The figure below shows the components for a Synergy graphics solution and the flow of graphics data.

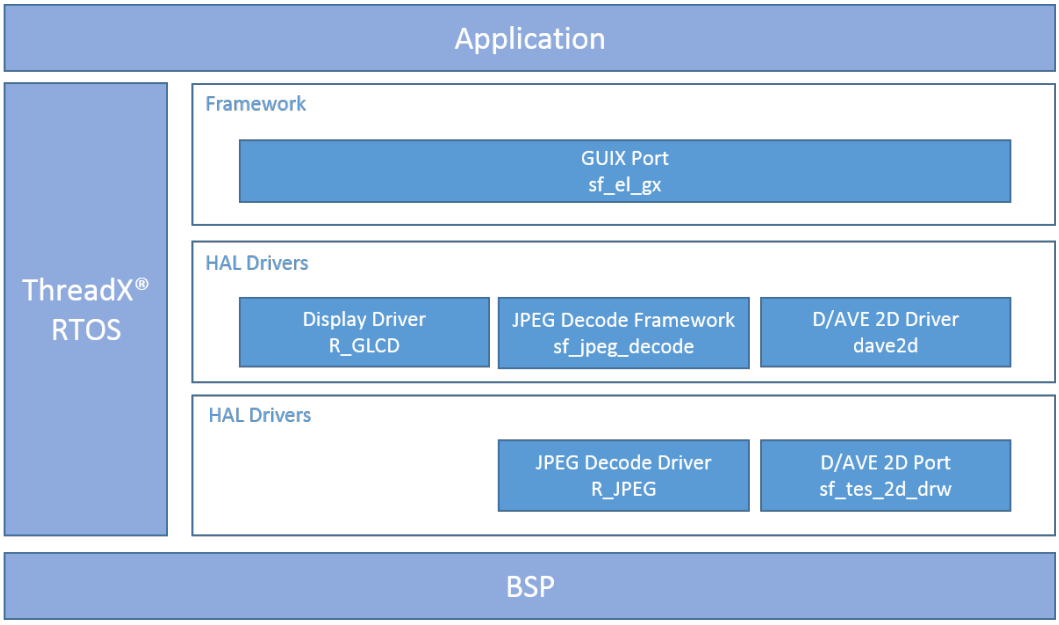


Figure 10.21 Synergy GUIX™ Interface Framework

The Framework supports the following functions:

- Adapts GUIX™ on top of SSP.
- Attaches SSP DISPLAY Interface driver to GUIX™ display driver Interface.
- Allows GUIX™ to draw widgets accelerated by Synergy D2W (2DG) engine.
- Allows GUIX™ to draw widgets accelerated by Synergy JPEG engine.
- Supports double-buffer toggling control for screen transitions without tearing.
- Supports for user callback functions.
- Supports image rotation drawn by GUIX™ for use cases where GUI screen design requires landscape orientation but the display panel hardware has a portrait screen
    - Supports configurable 90, 180, 270 degree rotation in counter clockwise direction

10.21.2 Estimated Memory Requirements

Table 10.39 Memory Usage for Synergy GUIX™ Interface Framework – GCC Compiler

sf_el_gx	Flash (Bytes)
S5D9 MCU Group	2,087
S7G2 MCU Group	2,087

Table 10.40 Memory Usage for Synergy GUIX™ Interface Framework – IAR Compiler

sf_el_gx	Flash (Bytes)
S5D9 MCU Group	1,780
S7G2 MCU Group	1,780

10.22 Synergy NetX™ Communication Framework

10.22.1 Component Introduction

The Synergy NetX Communications Interface Frameworks provides generic, high-level, C-callable APIs for applications using NetX Telnet Server. The Framework is ThreadX® aware and uses ThreadX® objects like mutex for blocking and synchronization techniques like event flags for the completion of a transaction.

The NetX Communication Framework in Synergy implements the Synergy Communications Interface.

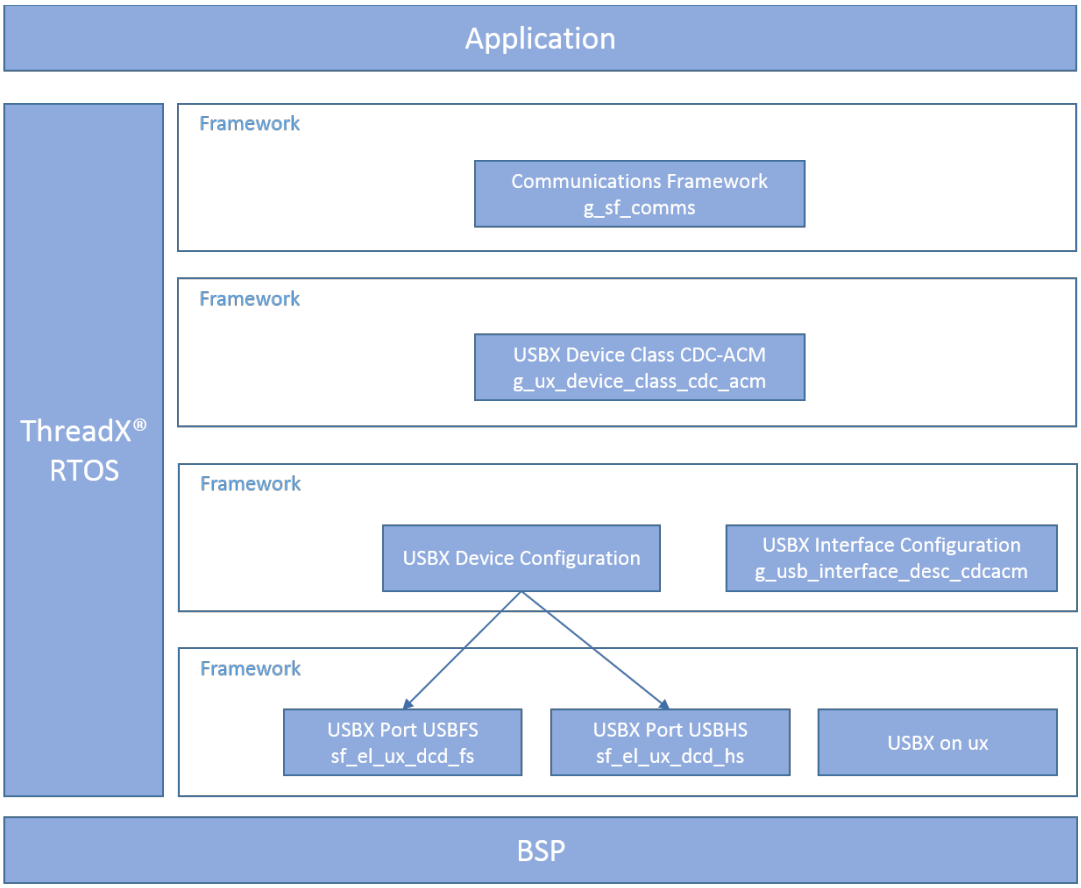


Figure 10.24 Synergy NETX™ Interface Framework

10.22.2 Estimated Memory Requirements

Table 10.41 Memory Usage for Synergy Communication Interface Framework – GCC Compiler

sf_el_nx_comms	Flash (Bytes)
S5D9 MCU Group	2,056
S7G2 MCU Group	2,056

Table 10.42 Memory Usage for Synergy Communication Interface Framework – IAR Compiler

sf_el_nx_comms	Flash (Bytes)
S5D9 MCU Group	2,056
S7G2 MCU Group	2,056

10.23 Synergy USBX Communication Framework

10.23.1 Component Introduction

Synergy USBX Communication Framework is a RTOS-aware interface for adding USBX CDC ACM capability to applications based on SSP. The Framework supports the following device procedures:

- Open
- Close
- Read
- Write

The USBX Communication Framework (CDC ACM) in Synergy implements the Synergy Communications Interface.

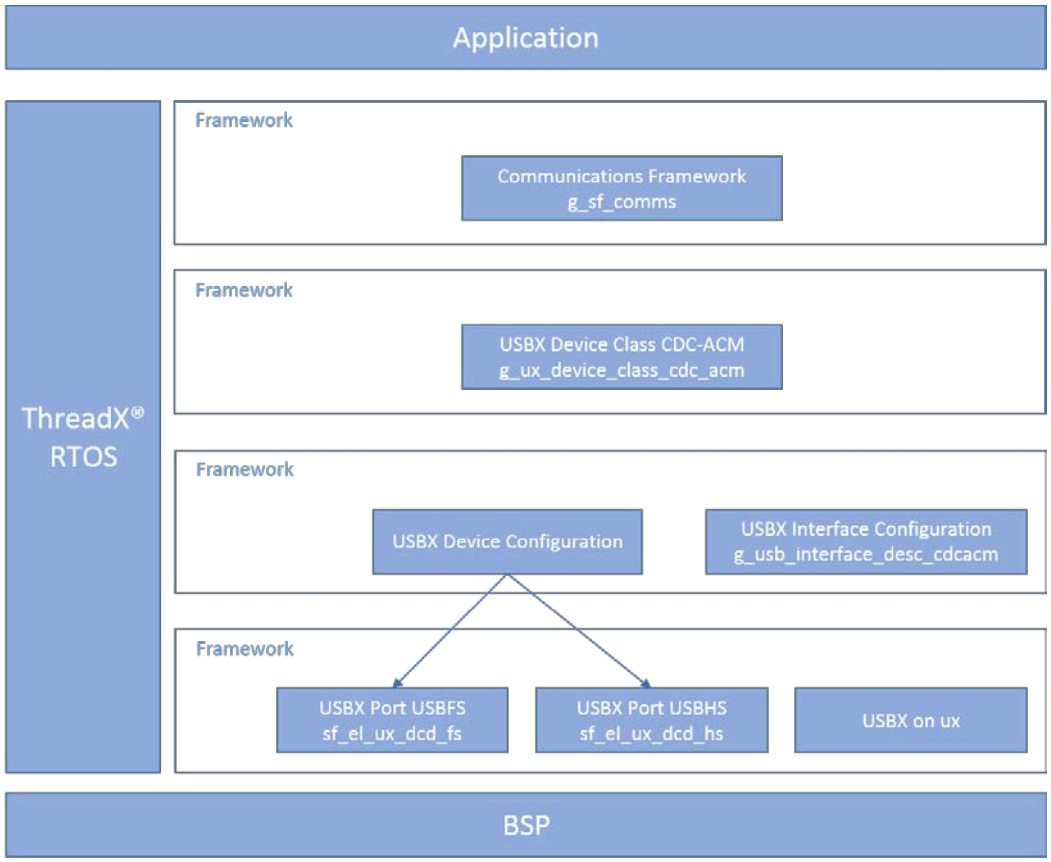


Figure 10.25 Synergy NETX™ Interface Framework

10.23.2 Estimated Memory Requirements

Table 10.43 Memory Usage for Synergy USBX Communication Framework – GCC Compiler

sf_el_ux_comms	Flash (Bytes)
S124 MCU Group	1,061
S3A7 MCU Group	1,061
S5D9 MCU Group	1,061
S7G2 MCU Group	1,061

Table 10.44 Memory Usage for Synergy USBX Communication Framework – IAR Compiler

sf_el_ux_comms	Flash (Bytes)
S124 MCU Group	860
S3A7 MCU Group	860
S5D9 MCU Group	860
S7G2 MCU Group	860

10.24 Block Media Interface for SD/Multi Media Card

10.24.1 Component Introduction

The Framework Block Media Interface is an abstract interface using function pointers instead of direct function calls. Functions are called between FileX® and the Synergy block media drivers, such as SD/MMC and SPI Flash. The interface remains the same for any media driver so that all media drivers appear functionally identical at file I/O layer and can be interchanged with one another without changing code.

Device adaptation drivers, such as `r_block_media_sdmmc`, are accessed through the Block Media Interface and provide device specific code needed to perform media I/O operations. Configuration and control structures passed through block media function calls are generally device specific as well.

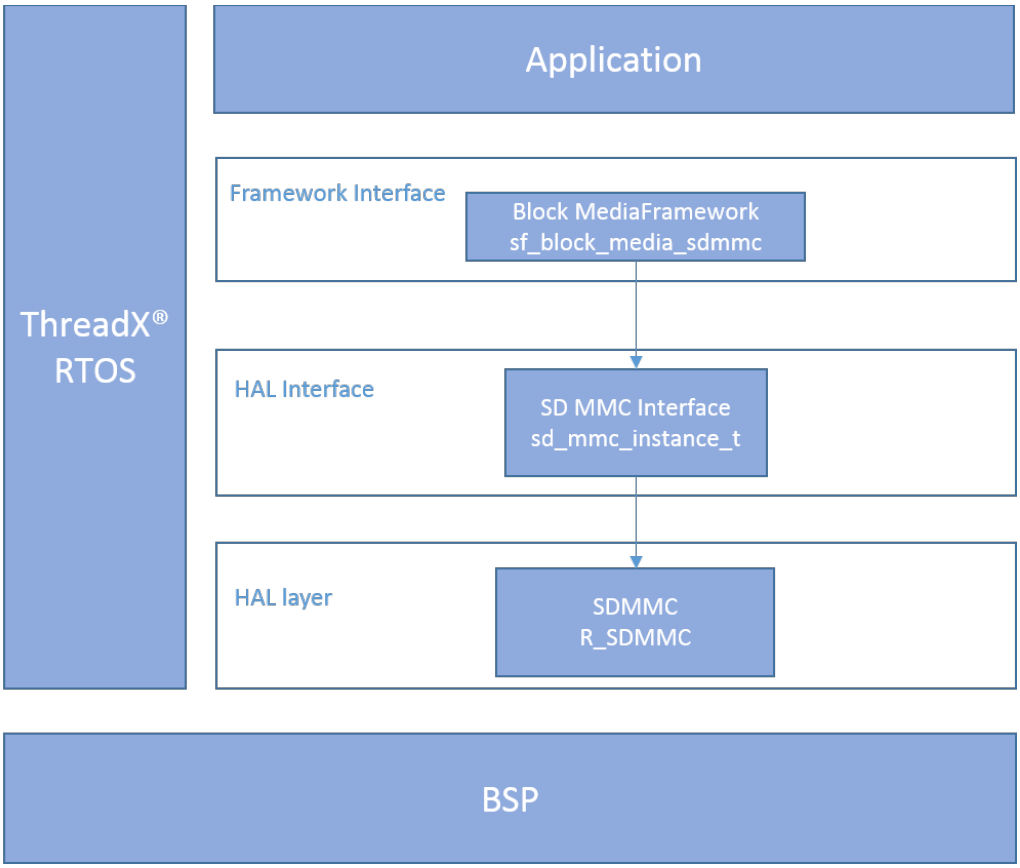


Figure 10.22 Block Media Interface for SD Multi Media Card

10.24.2 Estimated Memory Requirements

10.45 Memory Usage for Block Media Interface for SD Multi Media Card – GCC Compiler

sf_block_media_sdmmc	Flash (Bytes)
S3A7 MCU Group	399
S5D9 MCU Group	399
S7G2 MCU Group	399

Table 10.46 Memory Usage for Block Media Interface for SD Multi Media Card – IAR Compiler

sf_block_media_sdmmc	Flash (Bytes)
S3A7 MCU Group	380
S5D9 MCU Group	380
S7G2 MCU Group	380

11. Crypto Library

11.1 Component Introduction

The Secure Crypto Engine (SCE7) is the security and encryption block on Synergy S7G2 group MCU. It features many security features and National Institute of Standards and Technology (NIST)-compliant, primitive cryptographic algorithms for various applications. These features and algorithms can perform authentication and secure communication between the microcontroller and an external communication device or network, and can encrypt confidential and sensitive data and program for storage in the microcontroller. The security and encryption block also features high-throughput and low-power hardware accelerators to enable authentication and to meet secure communication requirements for various applications. The SSP Cryptographic library provides a simple C-callable API interface for these functions and capabilities available in the SCE7.

The SCE7 incorporates a high-throughput, 128-bit true random number generator (TRNG) that can generate random numbers with high entropy for use as seeds to deterministic random number generators (such as NIST SP800-90A DRBG). The TRNG generates cryptographically secure random numbers. Synergy MCUs also support several cryptographic hashing functions SHA1/SHA224/SHA256/MD5/GHASH.

Additionally, SCE7 supports several NIST-compliant symmetric encryption algorithms like Advanced Encryption Standard (AES 128/192/256-bit), Data Encryption Standard (3DES/DES) and Alleged RC4 (ARC4). These encryption algorithms, along with private keys, are used for secure data exchange and to securely store data and program in the MCU.

SCE7 also supports several NIST-compliant asymmetric algorithms for data exchange. The data transmitter and receiver uses shared keys.

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).
- Used for public-key cryptography.
- Generates two keys: public and private.
- Transmitter encrypts using the public key.
- Receiver decrypts using the private key.
- Supports up to 2048-bit key length.
- Used in digital verification for authentication, signature generation and verification, encryption/decryption for key exchange and wrapping, and other security functions.

Table 11.1 Cryptographic Library Functions supported on Synergy MCUs

	S7G2, S5D9 Groups	S3A7 Group	S124 Group	Notes
TRNG - Functions supported	Generate and read random number	Generate and read random number	Generate and read random number	Random number generation
AES - Functions supported	Encryption, decryption	Encryption, decryption	Encryption, decryption	Symmetric Key Encryption based on AES standard
AES - Key Size	128-bit, 192-bit, 256-bit	128-bit, 256-bit	128-bit, 256-bit	
AES - Chaining modes	ECB, CBC, CTR, GCM, XTS	ECB, CBC, CTR, GCM	ECB, CBC, CTR	
RSA - Functions supported	Signature Generation, Signature Verification, Public-key Encryption, Private-key Decryption	N/A	N/A	Supports CRT keys and standard keys for private key operations
RSA - Key Sizes supported	1,024-bit, 2048-bit	N/A	N/A	
DSA - Functions supported	Signature Generation, Signature Verification	N/A	N/A	
DSA - Key Sizes supported	(1,024, 128)-bit, (2,048, 224)-bit, (2,048, 256)-bit	N/A	N/A	
HASH - Methods supported	SHA1, SHA224, SHA256	N/A	N/A	Message digest algorithms

11.2 Estimated Memory Requirements

ARM Cortex M4 with GCC S7G2 Group MCU Crypto AES Encrypted key		ARM Cortex M4 with IAR S7G2 Group MCU Crypto AES Encrypted key	
Name	**Flash**	**Name**	**Flash**
aes	212	aes	182
aes128	18,700	aes128	16,614
aes192	10,577	aes192	9,770
aes256	15,856	aes256	14,142
arc4	2,152	arc4	1,772
dsa	28	dsa	24
dsa1024	176	dsa1024	166
dsa2048	360	dsa2048	332
hrk	2,756	hrk	2,382
r_sce	140	r_sce	124
rsa	108	rsa	100
RSA1024 enc with Pub-key	1,852	rsa1024	1,508
RSA1024 dec with CRT (caluclation of modular expornentiation exponent is up to 1024 bit)	1,492	rsa	1,148
RSA1024 dec with CRT	9,704	rsa	8,836
RSA2048 enc with Pub-key	2,564	rsa2048	2,036
RSA2048 dec with Prv-key	2,372	rsa2048	1,756
RSA2048 dec with CRT	10,432	rsa2048	9,572
sha1	536	sha1	80
sha256	572	sha256	80
tdes	5,788	tdes	84
tdes192	5,400	tdes192	92
trng	572	trng	472

ARM Cortex M4 with GCC S7G2 Group MCU Crypto AES Plain-Text key		ARM Cortex M4 with IAR S7G2 Group MCU Crypto AES Plain-Text key	
Name	**Flash**	**Name**	**Flash**
aes	212	aes	182
aes128	14,688	aes128	12,390
aes192	11,112	aes192	9,674
aes256	15,980	aes256	13,714
arc4	2,152	arc4	1,772
dsa	28	dsa	24
dsa1024	10,664	dsa1024	9,110
dsa2048	23,944	dsa2048	20,080
hrk	2,756	hrk	2,382
r_sce	140	r_sce	124
rsa	108	rsa	100
RSA1024 enc with Pub-key	1,852	rsa1024	1,508
RSA1024 dec with CRT (caluclation of modular exponentiation. exponent is up to 1024 bit)	1,492	rsa	1,148
RSA1024 dec with CRT	9,704	rsa	8,836
RSA2048 enc with Pub-key	2,564	rsa2048	2,036
RSA2048 dec with Prv-key	2,372	rsa2048	1,756
RSA2048 dec with CRT	10,432	rsa2048	9,572
sha1	536	sha1	80
sha256	572	sha256	80
tdes	3,752	tdes	3,396
tdes192	3,752	tdes192	3,404
trng	572	trng	472

12. CMSIS DSP Library

12.1 Component Introduction

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:

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

12.2 Estimated Memory Requirements

Estimated memory requirements for the CMSIS-DSP library are as follows:

CMSIS DSP ARM Cortex M0+ with GCC		CMSIS DSP ARM Cortex M0+ with IAR		CMSIS DSP ARM Cortex M4 with GCC		CMSIS DSP ARM Cortex M4 with GCC	
Name	**Flash**	**Name**	**Flash**	**Name**	**Flash**	**Name**	**Flash**
abs	196	abs	188	abs	692	abs	476
add	280	add	224	add	512	add	376
biquad	4,928	biquad	4,342	biquad	6,340	biquad	4,996
bitreversal	552	bitreversal	392	bitreversal	500	bitreversal	436
bitreversal2	80	bitreversal2	80	bitreversal2	192	bitreversal2	192
cfft	22,648	cfft	18,384	cfft	14,844	cfft	12,738
cmplx	2,304	cmplx	1,822	cmplx	5,396	cmplx	4,064
common	216,778	common	216,780	common	216,778	common	216,780
conv	17,752	conv	13,866	conv	20,632	conv	14,768
copy	92	copy	72	copy	332	copy	224
correlate	8,928	correlate	6,880	correlate	9,916	correlate	6,964
cos	312	cos	284	cos	276	cos	268
dct4	327,532	dct4	327,360	dct4	328,216	dct4	327,896
dot	332	dot	188	dot	680	dot	442
fill	56	fill	62	fill	240	fill	194
fir	10,884	fir	9,004	fir	18,128	fir	13,876
float	248	float	202	float	624	float	580
iir	1168	iir	980	iir	1,940	iir	1,464
lms	3092	lms	2,520	lms	4,516	lms	3,428
mat	6660	mat	5,878	mat	7,256	mat	5,610
max	248	max	174	max	720	max	486
mean	196	mean	170	mean	384	mean	338

CMSIS DSP ARM Cortex M0+ with GCC		CMSIS DSP ARM Cortex M0+ with IAR		CMSIS DSP ARM Cortex M4 with GCC		CMSIS DSP ARM Cortex M4 with GCC	
Name	**Flash**	**Name**	**Flash**	**Name**	**Flash**	**Name**	**Flash**
min	248	min	176	min	720	min	486
mult	288	mult	236	mult	716	mult	518
negate	184	negate	172	negate	424	negate	340
offset	272	offset	210	offset	424	offset	326
pid	440	pid	374	pid	276	pid	210
power	212	power	172	power	488	power	340
q15	88	q15	84	q15	356	q15	260
q31	80	q31	78	q31	336	q31	270
q7	88	q7	84	q7	376	q7	270
rfft	167,904	rfft	167,424	rfft	166,896	rfft	166,492
rms	324	rms	262	rms	408	rms	324
scale	364	scale	288	scale	896	scale	680
shift	356	shift	286	shift	932	shift	662
sin	1,252	sin	1,232	sin	1,092	sin	1,072
sqrt	560	sqrt	564	sqrt	484	sqrt	464
std	548	std	436	std	620	std	514
sub	280	sub	222	sub	516	sub	374
var	492	var	372	var	556	var	474

13. Hardware Abstraction Layer (HAL) Modules

13.1 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 MCU's and the level of support for these peripherals and features in SSP HAL software.

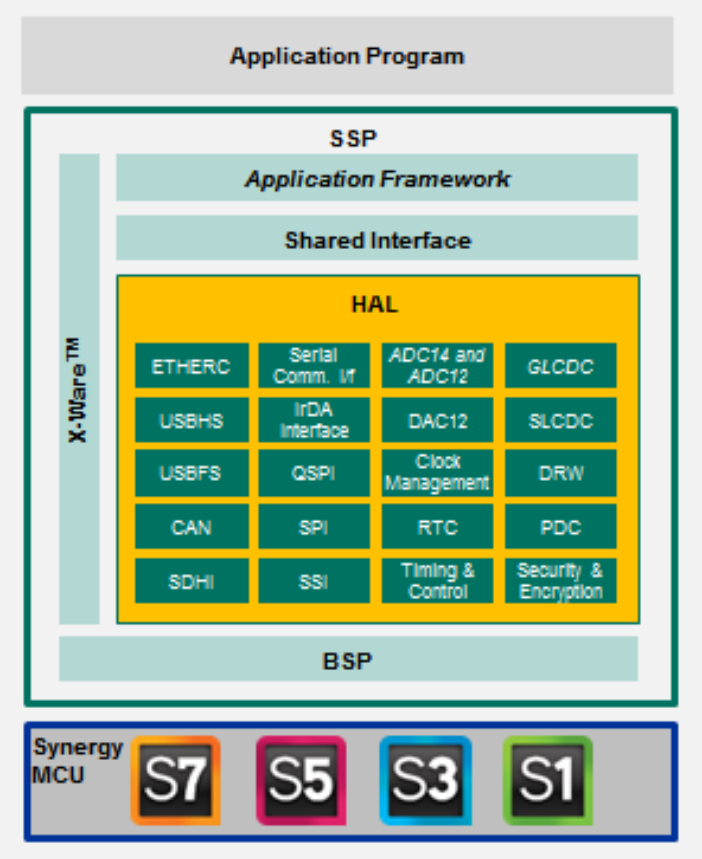


Figure 13.1 Hardware Abstraction Layer

13.2 SD Multi Media Card (SDMMC)

13.2.1 Component Introduction

The SDMMC driver module is a used access SD and an eMMC memory device on Synergy MCUs. The driver implements the SD/MMC bus protocol for read, write, and control of SD cards and eMMC embedded devices through the SDHI (SD Host Interface) peripheral. The SDMMC module can be used for standalone a SD card, eMMC, media driver, or it can be used with FileX®, or any other compatible file system. The SDMMC module can also be used as a standalone SDIO card driver.

Specifications and Features supported by module:

- Multiple channels of SD/MMC Host Interface SDHI.
- SDSC (SD Standard Capacity), SDHC (SD High Capacity) and eMMC (embedded) modes.
- 1, 4 or 8 bit (eMMC only) data bus.
- The Block Media Driver supports the SDMMC peripheral on the Synergy microcontroller hardware.

The SDMMC module implements the SDMMC interface in SSP.

13.2.2 Estimated Memory Requirements

Table 13.1 Memory Usage for SD Multi Media Card (SDMMC) – GCC Compiler

r_sdmmc	Flash (Bytes)
S3A7 MCU Group	2,915
S5D9 MCU Group	2,915
S7G2 MCU Group	2,915

Table 13.2 Memory Usage for SD Multi Media Card (SDMMC) – IAR Compiler

r_sdmmc	Flash (Bytes)
S3A7 MCU Group	2,556
S5D9 MCU Group	2556
S7G2 MCU Group	2,556

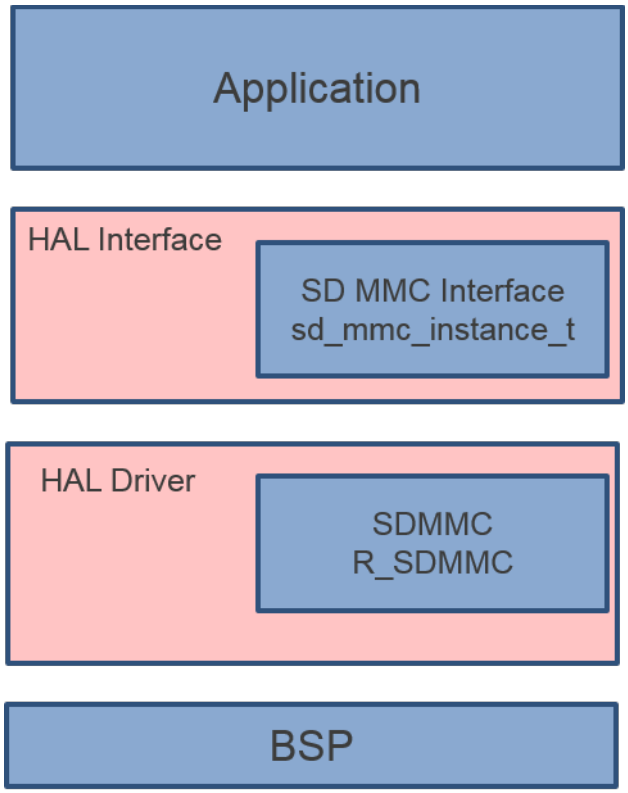


Figure 13.2 SD Multi Media Card

13.2.3 SSP Supported Hardware Features: SD/MMC (SDIO)

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
S3A7	✓	✓	✓	✓	✓	✓
S5D9 or S7G2	✓	✓	✓	✓	✓	✓

	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
S3A7	✓	☒	N/A	N/A	✓	☒
S5D9 or S7G2	✓	☒	Available on some S7G2 MCUs based on pin map	Available on some S7G2 MCUs based on pin map	✓	☒

	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
S3A7	✓	✓	☒	☒	✓	✓
S5D9 or S7G2	✓	✓	☒	☒	✓	✓

	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
S3A7	✓	☒	✓	☒	☒	☒
S5D9 or S7G2	✓	☒	✓	☒	☒	☒

	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
S3A7	N/A	N/A	N/A	N/A	N/A	N/A
S5D9 or S7G2	N/A	N/A	N/A	N/A	N/A	N/A

13.3 Clock Generation Circuit (CGC)

13.3.1 Component Introduction

The CGC driver supports the on–chip Clock Generation circuit available in Synergy MCUs, where the CGC peripheral is used to serve as the clock source for the MCU and its peripherals. The CGC driver provides the ability to configure and use all of the CGC module's capabilities. Among the capabilities is the selection of several clock sources to use as the system clock source. Additionally, the system clocks can be divided down to provide a wide range of frequencies for various system and peripheral needs, clock stability can be checked, and clocks may also be stopped to save power

when not needed. The driver can return the frequency of the system and system peripheral clocks at run time. The driver also can detect when the main oscillator has stopped with the option of calling a user provided callback function.

CGC driver can be used to configure the clocks on the Synergy microcontroller as follows:

- Configure any of the available clocks (HOCO, MOCO, LOCO, Main Clock, PLL, Sub-Oscillator) as the system clock source.
- Configure the internal clocks (ICLK, PCLK etc.).
- Switch the clocks on and off.
- Configure the output clocks.
- Set up the Oscillation Stop Detection feature.The CGC module implements the CGC interface in SSP:

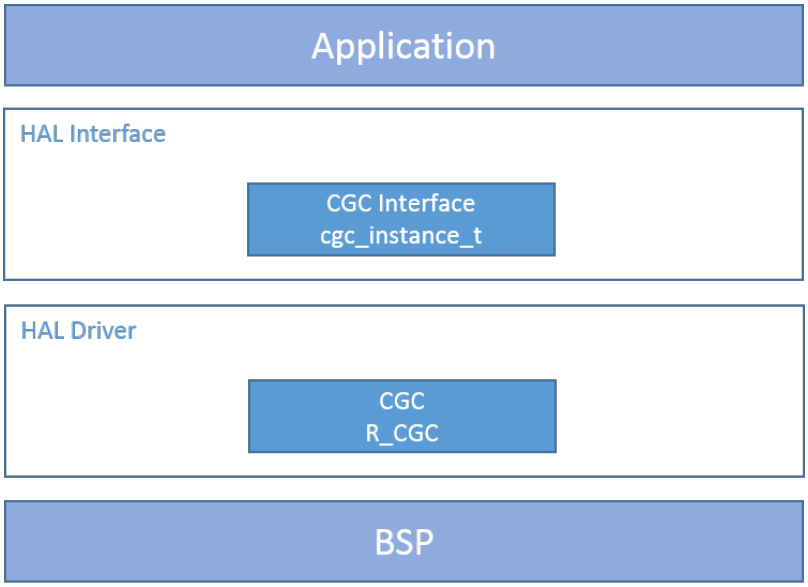


Figure 13.3 CGC module

13.3.2 Estimated Memory Requirements

Table 13.3 Memory Usage for Clock Generation Circuit (CGC) – GCC Compiler

r_cgc	Flash (Bytes)
S124 MCU Group	4,251
S3A7 MCU Group	4,127
S5D9 MCU Group	4,127
S7G2 MCU Group	4,127

Table 13.4 Memory Usage for Clock Generation Circuit (CGC) – IAR Compiler

r_cgc	Flash (Bytes)
S124 MCU Group	3,576
S3A7 MCU Group	3,508
S5D9 MCU Group	3,508
S7G2 MCU Group	3,508

13.3.3 SSP Supported Hardware Features: 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

	MOSC	SOSC	PLL Circuit	HOCO	MOCO	LOCO
S124	✓	✓	N/A	✓	✓	✓
S3A7	✓	✓	✓	✓	✓	✓
S5D9 or S7G2	✓	✓	✓	✓	✓	✓

	IWDTLOCO	JTAG External clock input	SWD External clock input
S124	✓	N/A	N/A
S3A7	✓	✓	✓
S5D9 or S7G2	✓	✓	✓

13.4 Capacitive Touch Sensing Unit (CTSU)

13.4.1 Component Introduction

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 CTSU HAL driver is used to initialize 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 like buttons, wheels, and sliders. 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 **Capacitive Touch Workbench for Renesas 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:

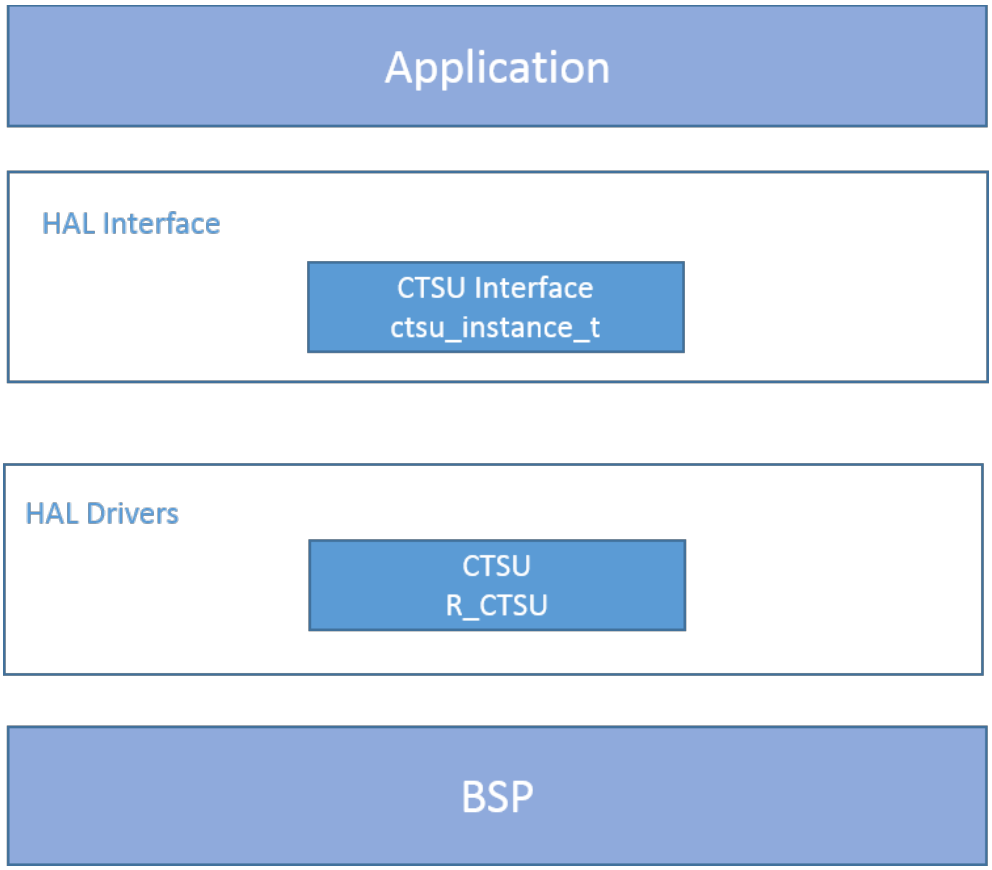


Figure 13.4 Capacitive Touch Sensing Unit

13.4.2 Estimated Memory Requirements

Table 13.5 Memory Usage for Capacitive Touch Sensing Unit (CTSU) – GCC Compiler

r_ctsu	Flash (Bytes)
S124 MCU Group	15,021
S3A7 MCU Group	15,257
S5D9 MCU Group	15,257
S7G2 MCU Group	15,257

Table 13.6 Memory Usage for Capacitive Touch Sensing Unit (CTSU) – IAR Compiler

r_ctsu	Flash (Bytes)
S124 MCU Group	12,532
S3A7 MCU Group	11,944
S5D9 MCU Group	11,944
S7G2 MCU Group	11,944

13.4.3 SSP Supported Hardware Features: Capacitive Touch Sensing Unit (CTSU)

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

	Self-Cap Single Scan Mode	Self-Cap Multi Scan Mode	Mutual-Cap Full Scan Mode	Sensor Stabilization Wait Time and Measurement Time	CTSU Interrupts
S124	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓
S5D9 or S7G2	✓	✓	✓	✓	✓

13.5 Digital to Analog convertor (DAC)

13.5.1 Component Introduction

The DAC module in SSP supports the 12-bit D/A converter in Synergy MCUs. This driver configures the dual-channel 12-bit D/A Converter (DAC12) to output one of 4096 voltage levels between the positive and negative reference voltages.

Key features supported by module include:

- Set left-justified or right-justified 12-bit value format for 16-bit input data registers.
- Enable or disable output amplifiers.
- Select external or internal reference voltages.
- Operate in synchronous anti-interference mode with Analog-to-Digital Converter (ADC) Module.

The DAC driver implements the DAC interface in SSP:

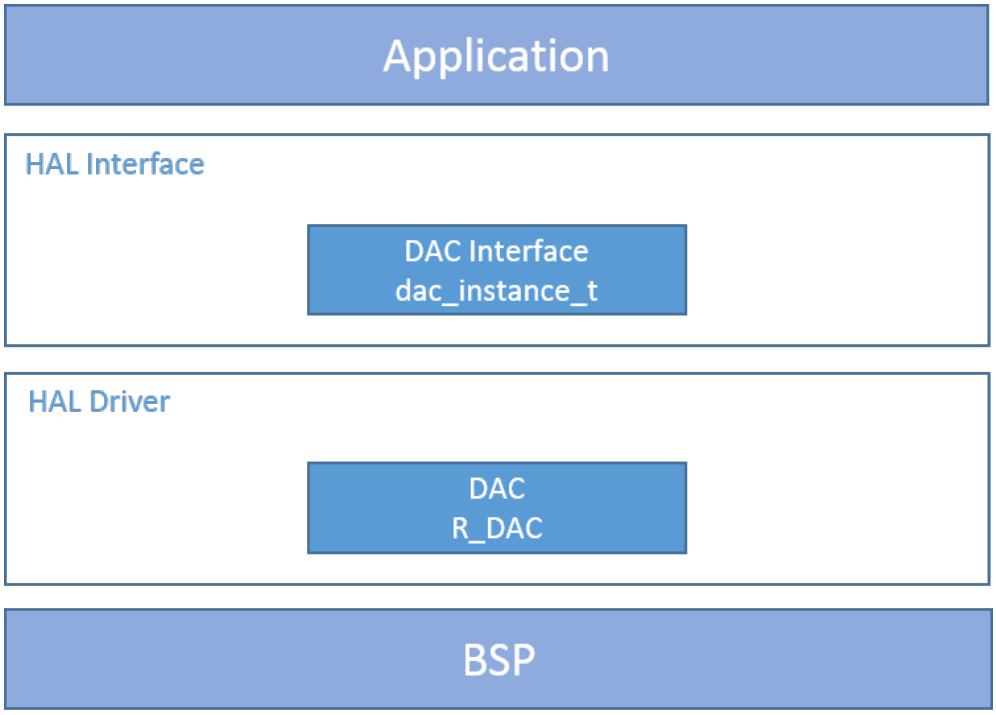


Figure 13.5 Digital to Analog convertor

13.5.2 Estimated Memory Requirements

Table 13.7 Memory Usage for Digital to Analog convertor (DAC) – GCC Compiler

r_dac	Flash (Bytes)
S124 MCU Group	877
S3A7 MCU Group	861
S5D9 MCU Group	861
S7G2 MCU Group	861

Table 13.8 Memory Usage for Digital to Analog convertor (DAC) – IAR Compiler

r_dac	Flash (Bytes)
S124 MCU Group	728
S3A7 MCU Group	712
S5D9 MCU Group	712
S7G2 MCU Group	712

13.5.3 SSP Supported Hardware Features: Digital to Analog convertor (DAC)

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

	12-Bit	2 Channels Output	Module Step Function	Event Link Function
S124	✓	N/A	✓	☒
S3A7	✓	✓	✓	☒
S5D9 or S7G2	✓	✓	✓	☒

13.6 Asynchronous General Purpose Timer (AGT)

13.6.1 Component Introduction

The AGT module supports the AGT peripheral in Synergy MCUs and can be used for accessing and configuring AGT timer modes. AGT is a 16-bit timer that can be configured to a user-specified period. When the period elapses, any of the following events can occur:

- Interrupt the CPU, which will call a user callback function if provided.
- Toggle a port pin.
- Transfer data using DMAC/DTC when configured with Transfer Interface.
- Start another peripheral when configured with ELC Interface.

The AGT supports runtime calculation of the period in standard units such as milliseconds and Hertz to ensure the period calculation is accurate and based on the current clock speed. The AGT can be used to wake the MCU from certain low power modes.

The AGT timer functions are used by the Timer Interface in SSP to provide timer services:

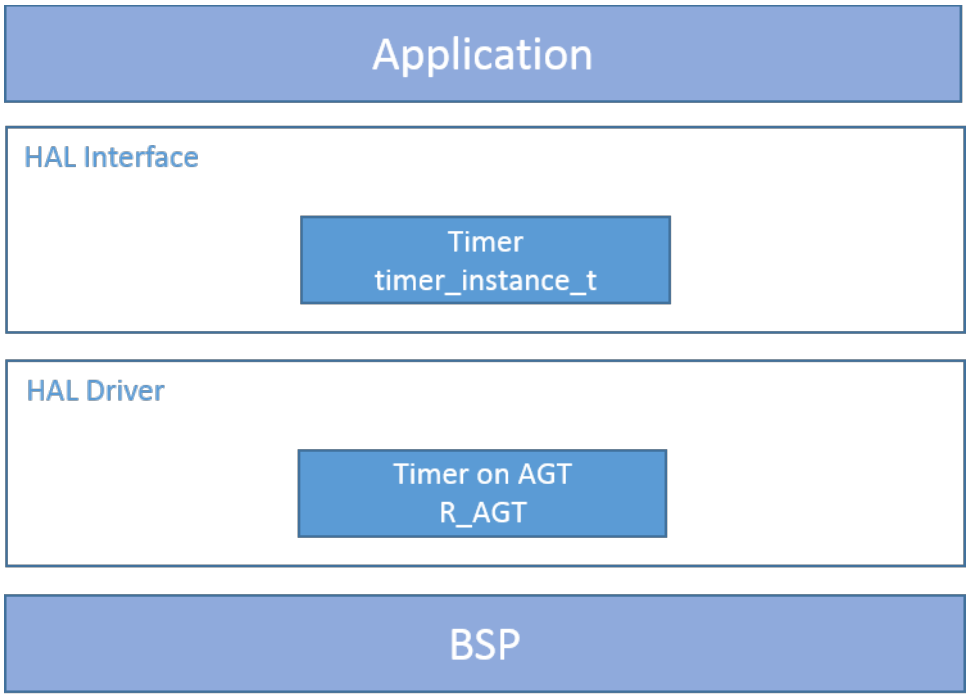


Figure 13.6 Asynchronous General Purpose Timer

13.6.2 Estimated Memory Requirements

Table 13.9 Memory Usage for Asynchronous General Purpose Timer (AGT) – GCC Compiler

r_agt	Flash (Bytes)
S124 MCU Group	3,015
S3A7 MCU Group	2,727
S5D9 MCU Group	2,727
S7G2 MCU Group	2727

Table 13.10 Memory Usage for Asynchronous General Purpose Timer (AGT) – IAR Compiler

r_agt	Flash (Bytes)
S124 MCU Group	2,494
S3A7 MCU Group	2,350
S5D9 MCU Group	2,350
S7G2 MCU Group	2,350

13.6.3 SSP Supported Hardware Features: Asynchronous General Purpose Timer (AGT)

The following hardware features are, or are not, supported by 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	✓	✓	✓	✓
S3A7	✓	✓	✓	✓
S5D9 or S7G2	✓	✓	✓	✓

MCU Group	Pulse period measurement mode	Interrupt/Event link function (Output)	Compare/Match Function
S124	✓	✓	✓
S3A7	✓	✓	✓
S5D9 or S7G2	✓	✓	✓

13.7 Cyclic Redundancy Check calculator (CRC)

13.7.1 Component Introduction

The CRC HAL driver supports the CRC hardware block in Synergy MCUs that can be 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 port using various types of industry standard polynomials. CRC calculations can be performed by sending data to the block using the CPU or by snooping on read or write activity on one of 10 SCI channels.

The CRC module supports the following functions:

- Calculate a CRC on a block of data in memory.
- Calculate a CRC on a stream of data being transmitted or received over a serial port.
- Specify the number of the serial port and the direction of data to perform the calculation on.

The CRC driver implements the CRC interface in SSP:

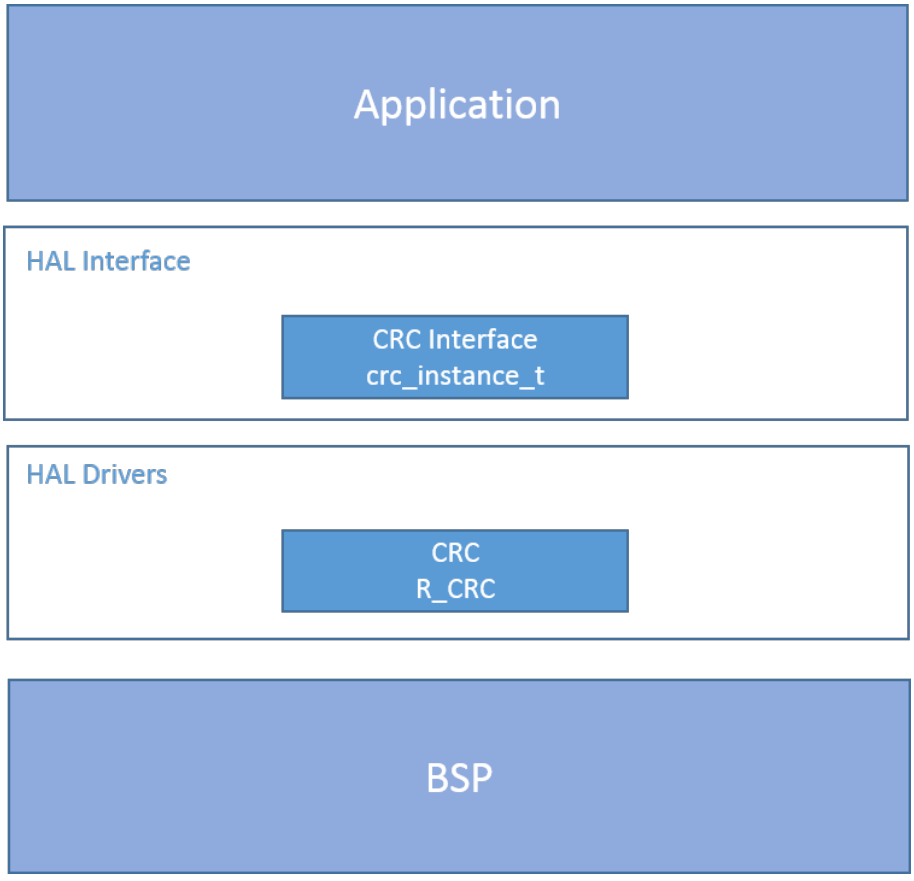


Figure 13.7 Cyclic Redundancy Check calculator

13.7.2 Estimated Memory Requirements

Table 13.11 Memory Usage for Cyclic Redundancy Check calculator (CRC) – GCC Compiler

r_crc	Flash (Bytes)
S124 MCU Group	973
S3A7 MCU Group	905
S5D9 MCU Group	905
S7G2 MCU Group	905

Table 13.12 Memory Usage for Cyclic Redundancy Check calculator (CRC) – IAR Compiler

r_crc	Flash (Bytes)
S124 MCU Group	772
S3A7 MCU Group	696
S5D9 MCU Group	696
S7G2 MCU Group	696

13.7.3 SSP Supported Hardware Features: Cyclic Redundancy Check calculator (CRC)

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 Step Function	CRC Snoop
S124	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓
S5D9 or S7G2	✓	✓	✓	✓	✓

13.8 Clock Frequency Accuracy Measurement (CAC)

13.8.1 Component Introduction

The CAC (Clock Accuracy Circuit) driver supports the 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 edges.

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)

A completed measurement may be identified by making API calls to poll the driver, or by establishing callback functions which are capable of triggering on any of the following conditions:

- Clock Measurement complete (MENDF)
- Clock Frequency error (FERRF)
- Clock counter overflow (OVFF)

The CAC driver implements the CAC interface in SSP:

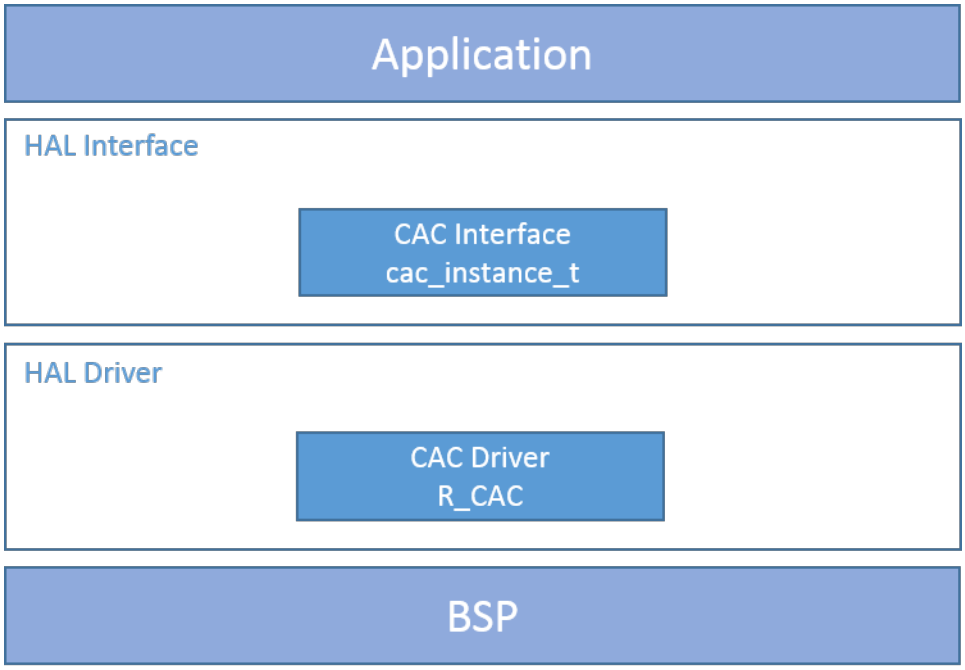


Figure 13.8 Clock Frequency Accuracy Measurement

13.8.2 Estimated Memory Requirements

Table 13.13 Memory Usage for Clock Frequency Accuracy Measurement (CAC) – GCC Compiler

r_cac	Flash (Bytes)
S124 MCU Group	2,789
S3 MCU Group	2,297
S5D9 MCU Group	2,297
S7G2 MCU Group	2,297

Table 13.14 Memory Usage for Clock Frequency Accuracy Measurement (CAC) – IAR Compiler

r_cac	Flash (Bytes)
S124 MCU Group	2,148
S3A7 MCU Group	2,180
S5D9 MCU Group	2,180
S7G2 MCU Group	2,180

13.8.3 SSP Supported Hardware Features: Clock Frequency Accuracy Measurement (CAC)

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	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓
S5D9 or 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	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓
S5D9 or S7G2	✓	✓	✓	✓	✓

	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	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓
S5D9 or S7G2	✓	✓	✓	✓	✓

	Selectable Function: Digital Filter	Interrupt Sources: Measurement End	Interrupt Sources: Frequency Error	Interrupt Sources: Overflow	Module-stop function to reduce power consumption
S124	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓
S5D9 or S7G2	✓	✓	✓	✓	✓

13.9 RIIC (I^2C Full Featured)

13.9.1 Component Introduction

SSP provides two RIIC drivers which implements two separate HAL interfaces for I^2C peripheral.

i) RIIC HAL module implements the I^2C master interface which configures and operates I^2C peripheral in master mode.

ii) 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 I²C peripheral can only operate either in master or slave mode (i.e. In a multi-master scenario, I2C peripheral cannot dynamically switch 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 I²C Normal-mode with bitrates up to 100 Kbps
- Supports I²C 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
- Receive complete

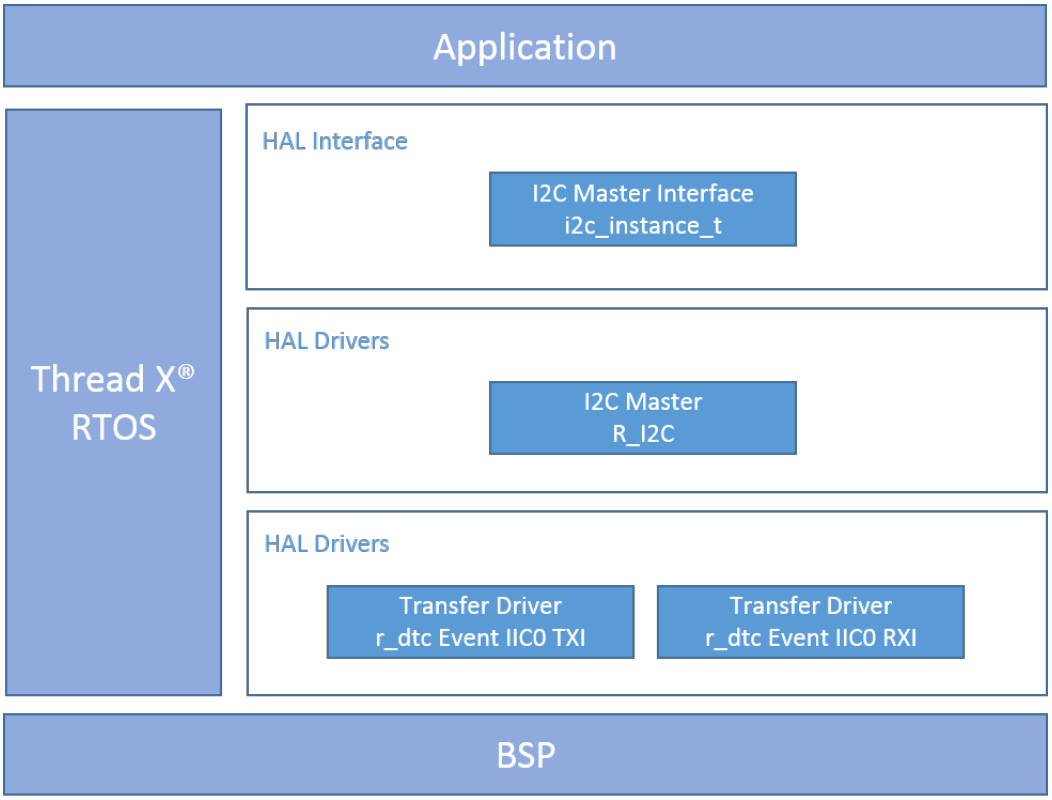


Figure 13.9 RIIC

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:

- Configure I²C peripheral with a 7 or 10 bit slave address
- Read from an I²C master device
- Write to an I²C master device

13.9.2 Estimated Memory Requirements

Table 13.15 Memory Usage for I2C (RIIC) – GCC Compiler

r_riic	Flash (Bytes)
S124 MCU Group	5,472
S3A7 MCU Group	5,664
S5D9 MCU Group	5,660
S7G2 MCU Group	5,660

Table 13.16 Memory Usage for I2C (RIIC) – IAR Compiler

r_riic	Flash (Bytes)
S124 MCU Group	5,220
S3A7 MCU Group	5,280
S5D9 MCU Group	5,280
S7G2 MCU Group	5,280

13.9.3 SSP Supported Hardware Features: RIIC

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	I^2C Format	SMBus Format	Master Mode	Slave Mode	Fast Mode Plus	Selectable duty cycle
S124	✓	☒	✓	✓	N/A	☒
S3A7	✓	☒	✓	✓	N/A	☒
S5D9 or 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	☒	7 and 10 bit	☒	✓	☒	☒
S3A7	☒	7 and 10 bit	☒	✓	☒	☒
S5D9 or 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	✓	☒	✓
S3A7	Master, NACK arbitrations	✓	☒	✓
S5D9 or S7G2	Master, NACK arbitrations	✓	☒	✓

RIIC Slave driver supported features:

MCU Group	I²C Format	SMBus Format	Master Mode	Slave Mode	Fast Mode Plus	Selectable duty cycle
S124	✓	☒	☒	✓	N/A	☒
S3A7	✓	☒	☒	✓	N/A	☒
S5D9 or 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	☒	✓	☒	☒
S3A7	One slave address	7 and 10 bit	☒	✓	☒	☒
S5D9 or 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
S124	Slave, NACK arbitration	✓	☒	TEI not supported
S3A7	Slave, NACK arbitration	✓	☒	TEI not supported
S5D9 or S7G2	Slave and NACK arbitration	✓	☒	TEI not supported

13.10 Serial Peripheral Interface (RSPI)

13.10.1 Component Introduction

The SPI HAL driver supports the SPI interface in Synergy MCUs and implements the SPI protocol.

The SPI driver configures SPI peripheral in master mode and supports the following functions:

- Initialize the driver.
- Serial Communication through SPI operation.
- Supports 8, 16 and 32 bit data transfers.
- Supports GPIO pins configured as chip selects.
- Communication with other devices as slave

The Interface also provides support for callbacks. The callback functions are called with the following events:

- Transfer aborted
- Transfer complete
- Mode fault
- Error events

The SPI HAL Interface is implemented by the SCI_SPI and SPI HAL driver modules in SSP:

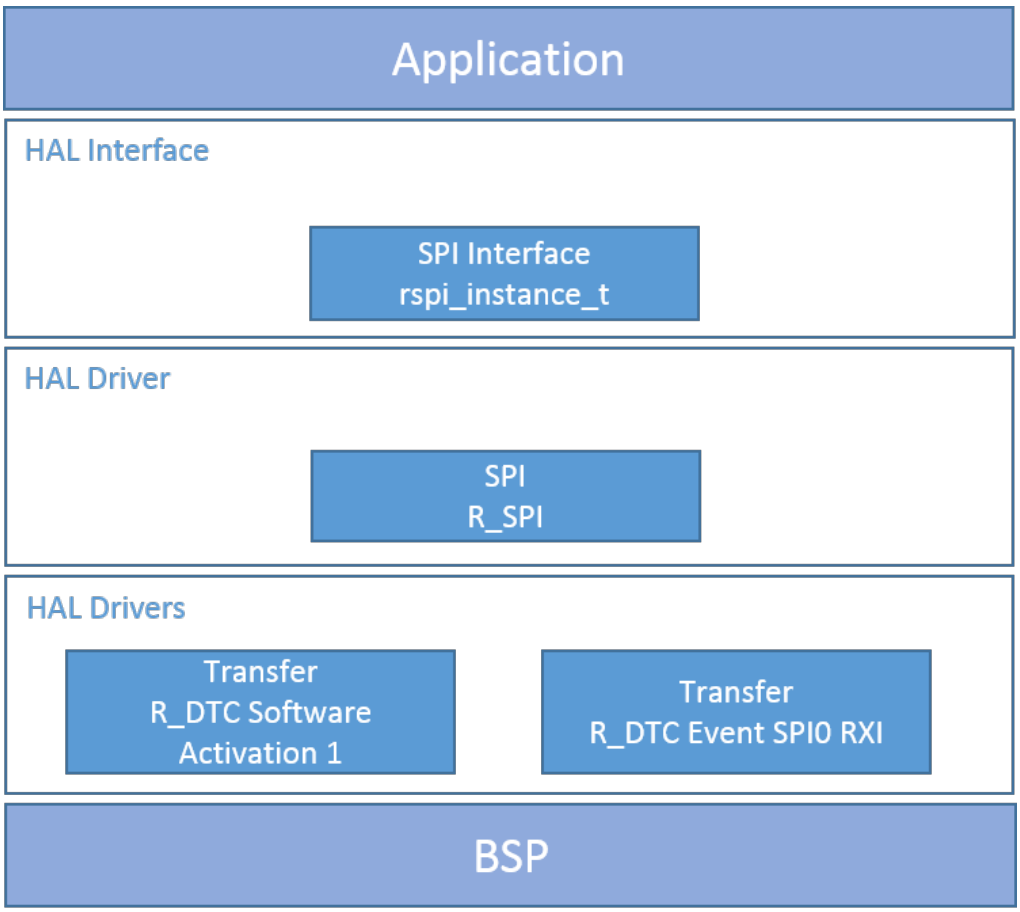


Figure 13.10 Serial Peripheral Interface

13.10.2 Estimated Memory Requirements

Table 13.17 Memory Usage for Serial Peripheral Interface (RSPI) – GCC Compiler

r_rspi	Flash (Bytes)
S124 MCU Group	3,895
S3A7 MCU Group	3,467
S5D9 MCU Group	3,467
S7G2 MCU Group	3,467

Table 13.18 Memory Usage for Serial Peripheral Interface (RSPI) – IAR Compiler

r_rspi	Flash (Bytes)
S124 MCU Group	3,068
S3A7 MCU Group	3,008
S5D9 MCU Group	3,008
S7G2 MCU Group	3,008

13.10.3 SSP Supported Hardware Features: RSPI

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	✓	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓	✓
S5D9 or 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	8/16	☒	✓	☒	✓	✓
S3A7	✓	☒	✓	☒	✓	✓
S5D9 or S7G2	✓	☒	✓	☒	✓	✓

MCU Group	Transfer of up to eight commands	All Interrupt sources	DTC Support	Event link function (output)	Function for switching between CMOS output and open-drain output	Loopback mode
S124	☒	✓	✓16-bit transfer	☒	✓	☒
S3A7	☒	✓	✓(32-bit transfer)	☒	✓	☒
S5D9 or S7G2	☒	✓	✓(32-bit transfer)	☒	✓	☒

MCU Group	Module Stop Function	Multi-master mode
S124	☒	☒
S3A7	☒	☒
S5D9 or S7G2	☒	☒

13.11 Quad SPI (QSPI)

13.11.1 Component Introduction

The QSPI HAL driver supports the Quad-SPI (QSPI) peripheral in Synergy microcontroller, which functions as a memory controller for connecting a serial ROM (non-volatile memory such as a serial flash memory, serial EEPROM, or serial FeRAM) with an SPI-compatible interface. The driver is used for erasing and programming the contents of a QSPI flash device connected to the microcontroller over the Quad SPI interface.

The QSPI driver supports the following functions:

- Access to 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.
- Erase specified blocks of QSPI flash device

- Select a bank to control access to a QSPI flash device.
- Query and receive information about the QSPI flash device such as size of Flash, flash program size and supported erase sizes. etc.

The QSPI driver implements the QSPI interface in SSP:

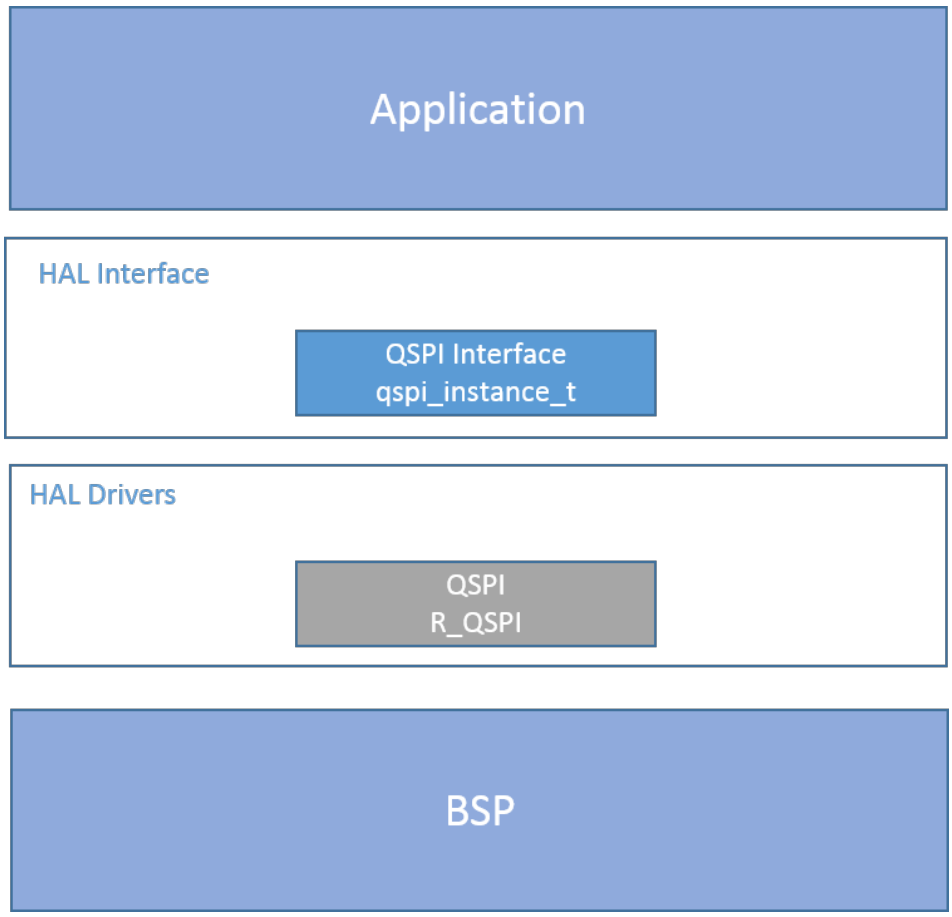


Figure 13.11 Quad SPI

13.11.2 Estimated Memory Requirements

Table 13.19 Memory Usage for Quad SPI (QSPI) – GCC Compiler

r_qspi	Flash (Bytes)
S3A7 MCU Group	1,296
S5D9 MCU Group	1,296
S7G2 MCU Group	1,296

Table 13.20 Memory Usage for Quad SPI (QSPI) – IAR Compiler

r_qspi	Flash (Bytes)
S3A7 MCU Group	1,112
S5D9 MCU Group	1,112
S7G2 MCU Group	1,112

13.11.3 SSP Supported Hardware Features: QSPI

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
S3A7	✓	☒	☒	☒	3 byte and 4 byte addresses	☒
S5D9 or 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
S3A7	✓	☒	☒	☒	☒	☒
S5D9 or 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
S3A7	N/A	☒	☒	N/A	N/A	N/A
S5D9 or S7G2	N/A	☒	☒	N/A	N/A	N/A

MCU Group	Interrupt source	Module-stop function	XIP
S124	N/A	N/A	N/A
S3A7	N/A	N/A	✓
S5D9 or S7G2	N/A	N/A	✓

13.12 Realtime Clock (RTC)

13.12.1 Component Introduction

The RTC HAL driver controls the Realtime Clock. The driver supports the RTC peripheral available on the Synergy microcontroller hardware.

The RTC driver supports the following functions of the Realtime Clock:

- RTC peripheral configuration
- Clock and calendar functions
- Alarm, periodic, and carry interrupts
- Time capture function
- Event linkage to other peripherals

The RTC driver implements the RTC interface in SSP:

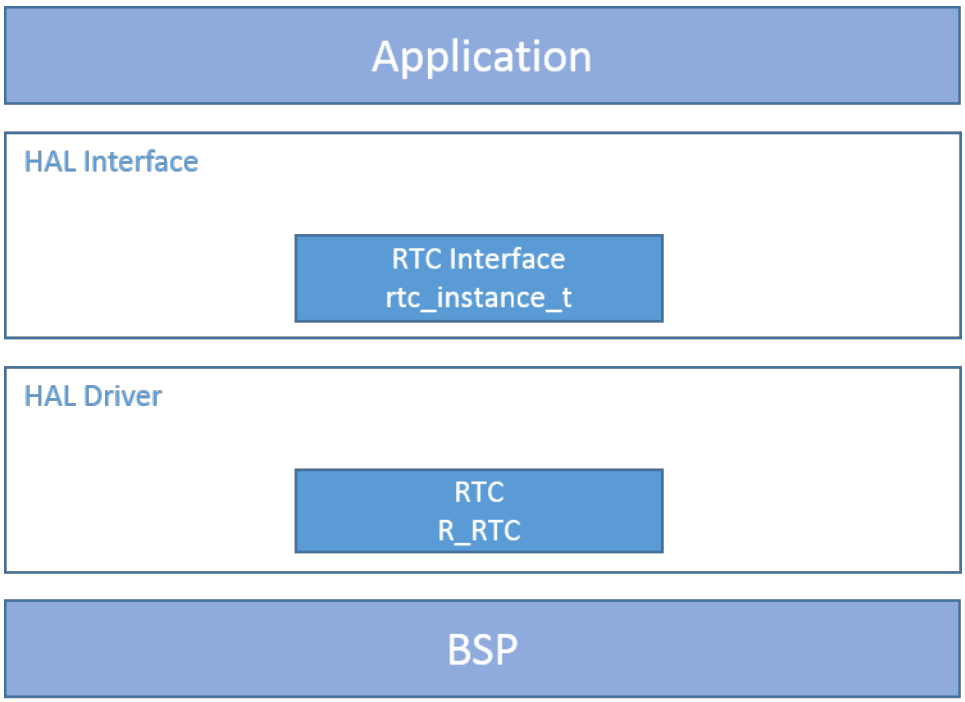


Figure 13.12 Realtime Clock

13.12.2 Estimated Memory Requirements

Legend:

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

Table 13.21 Memory Usage for Realtime clock (RTC) – GCC Compiler

r_rtc	Flash (Bytes)
S124 MCU Group	4,689
S3A7 MCU Group	4,185
S5D9 MCU Group	4,185
S7G2 MCU Group	4,185

Table 13.22 Memory Usage for Realtime clock (RTC) – IAR Compiler

r_rtc	Flash (Bytes)
S124 MCU Group	3,762
S3A7 MCU Group	3,064
S5D9 MCU Group	3,064
S7G2 MCU Group	3,064

13.12.3 SSP Supported Hardware Features: Real Time Clock (RTC)

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 or S128	✓	☒	✓	✓	24 Hours	✓
S3A7	✓	☒	✓	✓	24 Hours	✓
S5D9 or S7G2	✓	☒	✓	✓	24 Hours	✓

	Periodic interrupt (RTC_PRD)	Carry interrupt (RTC_CUP)	Time capture function	Event link function	Start/stop function	Clock error correction function
S124 or S128	✓	✓	☒	☒	✓	✓
S3A7	✓	✓	☒	☒	✓	✓
S5D9 or S7G2	✓	✓	☒	☒	✓	✓

13.13 Segment LCD (SLCDC)

13.13.1 Component Introduction

The Segment LCD HAL driver controls the Segment LCD Display. The driver supports the SLCD peripheral available on the Synergy microcontroller hardware. Segment LCD is a feature of the S3 MCU Group (the S5 and S7 MCU Groups have Graphics LCD drivers and hardware).

The driver 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, 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.

Module supports selecting the following features:

- 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).

The SLCDC driver implements the SLCDC interface in SSP:

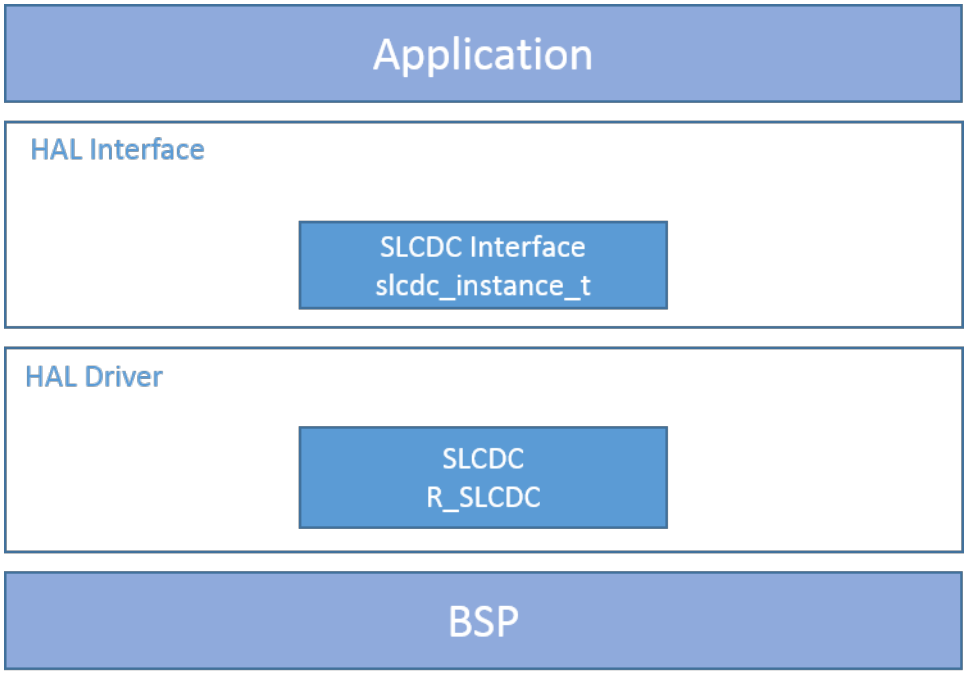


Figure 13.13 Segment LCD

13.13.2 Estimated Memory Requirements

Table 13.23 Memory Usage for Segment LCD (SLCDC) – GCC Compiler

r_slcdc	Flash (Bytes)
S3A7 MCU Group	1,679

Table 13.24 Memory Usage for Segment LCD (SLCDC) – IAR Compiler

r_slcdc	Flash (Bytes)
S3A7 MCU Group	1,564

13.13.3 SSP Supported Hardware Features: Segment LCD (SLCD)

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

	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
S3A7	✓	✓	☒	✓	☒	☒
S5D9 or S7G2	N/A	N/A	N/A	N/A	N/A	N/A

	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
S3A7	✓	✓	✓	☒	✓	✓
S5D9 or S7G2	N/A	N/A	N/A	N/A	N/A	N/A

13.14 Serial Communication Interface UART (SCI_UART)

13.14.1 Component Introduction

The SCI_UART driver enables serial communication using the UART protocol over the SCI peripheral in Synergy MCUs.

The UART Interface supports the generic UART protocol.

The UART Interface used with the SCI peripheral in UART mode (UART on SCI) supports multiple features in addition to the standard UART protocol.

Specifications and Features:

- 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
- Includes bit-rate modulation function on all SCI modules
- 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

The SCI_UART driver implements the UART Interface in SSP:

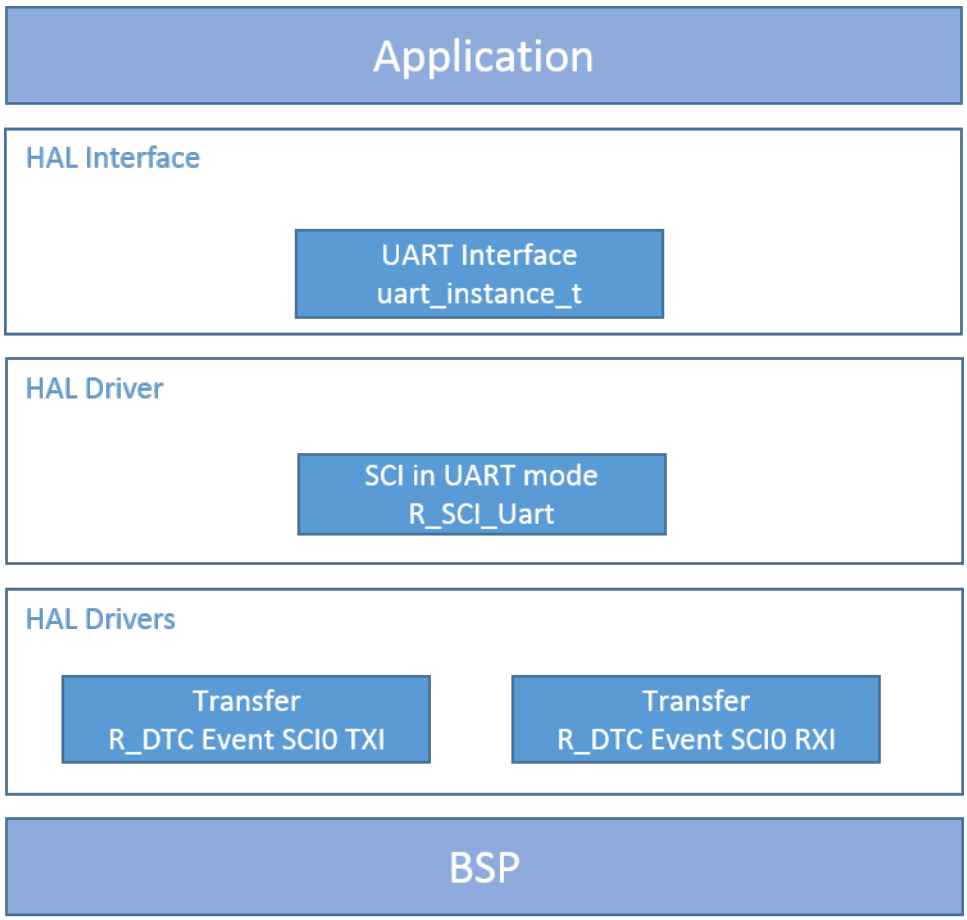


Figure 13.14 Serial Communication Interface UART

13.14.2 Estimated Memory Requirements

Table 13.25 Memory Usage for Serial Communication Interface UART (SCI_UART) – GCC Compiler

r_sci_uart	Flash (Bytes)
S124 MCU Group	4,874
S3A7 MCU Group	4,650
S5D9 MCU Group	4,650
S7G2 MCU Group	4,650

Table 13.26 Memory Usage for Serial Communication Interface UART (SCI_UART) – IAR Compiler

r_sci_uart	Flash (Bytes)
S124 MCU Group	4,380
S3A7 MCU Group	4,136
S5D9 MCU Group	4,136
S7G2 MCU Group	4,136

13.14.3 SSP Supported Hardware Features: UART SCI

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

	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	✓	☒	☒	✓	✓
S3A7	✓	☒	☒	✓	✓
S5D9 or S7G2	✓	☒	☒	✓	✓

	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	✓	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓	✓
S5D9 or S7G2	✓	✓	✓	✓	✓	✓

	Hardware flow control	Transmission and reception	Address match	Address un-match	Start-bit detection	Break detection
S124	✓	Register	☒	☒	✓	☒
S3A7	✓	FIFO	☒	☒	✓	☒
S5D9 or S7G2	✓	FIFO	☒	☒	✓	☒

	Clock source Internal/ external	Double-speed mode	Multi-processor communications	Noise cancellation	Bit rate modulation function	iRDA support
S124	✓	☒	☒	✓	✓	☒
S3A7	✓	☒	☒	✓	✓	☒
S5D9 or S7G2	✓	☒	☒	✓	✓	☒

13.15 Serial Communication Interface I²C over SCI (SCI_I2C)

13.15.1 Component Introduction

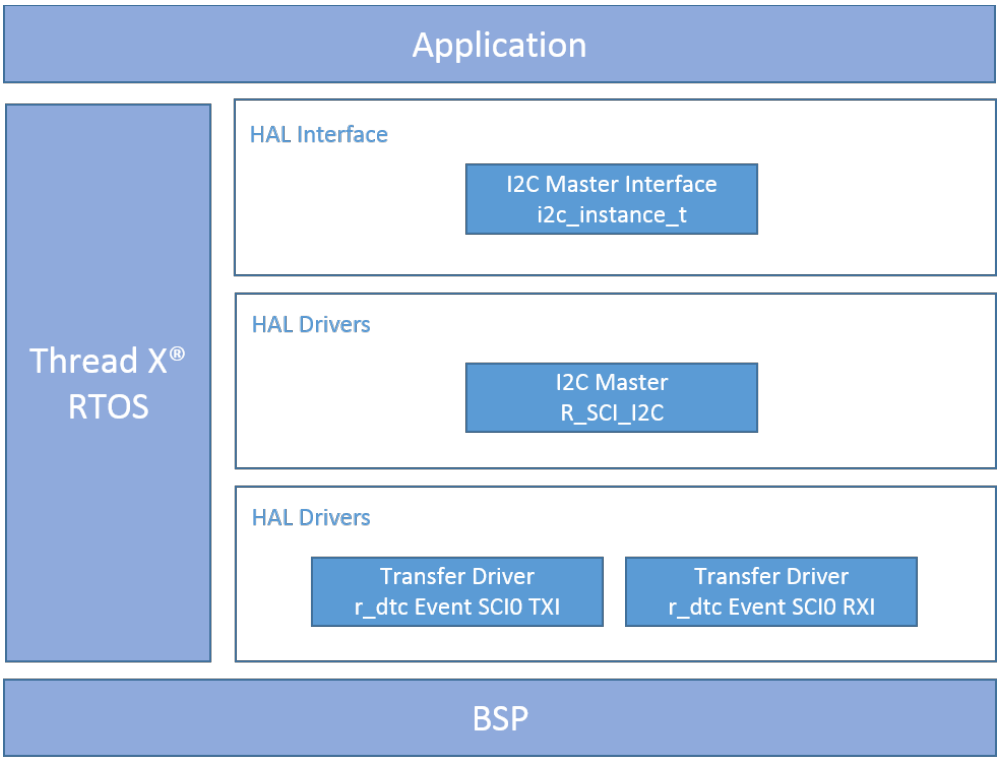
The SCI_I2C driver supports the SCI peripheral in Synergy MCUs. The driver supports the I²C protocol for communicating in **master mode only** and provides following capabilities:

- 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.



13.15.2 Estimated Memory Requirements

13.27 Memory Usage for Serial Communication Interface I2C (SCI_I2C) – GCC Compiler

r_sci_i2c	Flash (Bytes)
S124 MCU Group	5,170
S3A7 MCU Group	5,070
S5D9 MCU Group	5,102
S7G2 MCU Group	5,102

Table 13.28 Memory Usage for Serial Communication Interface I2C (SCI_I2C) – IAR Compiler

r_sci_i2c	Flash (Bytes)
S124 MCU Group	4,712
S3A7 MCU Group	4,676
S5D9 MCU Group	4,676
S7G2 MCU Group	4,676

13.15.3 SSP Supported Hardware Features: I²C over SPI

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

	Master Mode	Slave mode	Support all Interrupt Sources	Digital noise filter	Bit rate modulation	SDA delay
S124	✓	N/A	ERI not supported	☒	✓	✓
S3A7	✓	N/A	ERI not supported	☒	✓	✓
S5D9 or S7G2	✓	N/A	ERI not supported	☒	✓	✓

	Timeout on bus lockout
S124	N/A
S3A7	N/A
S5D9 or S7G2	N/A

13.16 Serial Communication Interface SPI (SCI_SPI)

13.16.1 Component Introduction

SCI_SPI HAL driver module supports SPI serial communication using the microcontroller's SCI peripheral. The module implements the SPI Interface. The SPI Interface configures SPI communication in master mode. The Interface allows:

- Initializing the driver
- Serial Communication through SPI operation
- Includes bit-rate modulation function on all SCI modules

The Interface also provides support for callbacks. The callback functions are called with the following events:

- Transfer aborted
- Transfer complete
- Mode fault
- Error events
- Selecting a SPI Module

SCI_SPI module support 8-bit data transfer and GPIO pins configured as chip selects. The module uses DTC for data transfer.

The SPI HAL Interface is implemented by the SCI_SPI HAL driver modules in SSP:

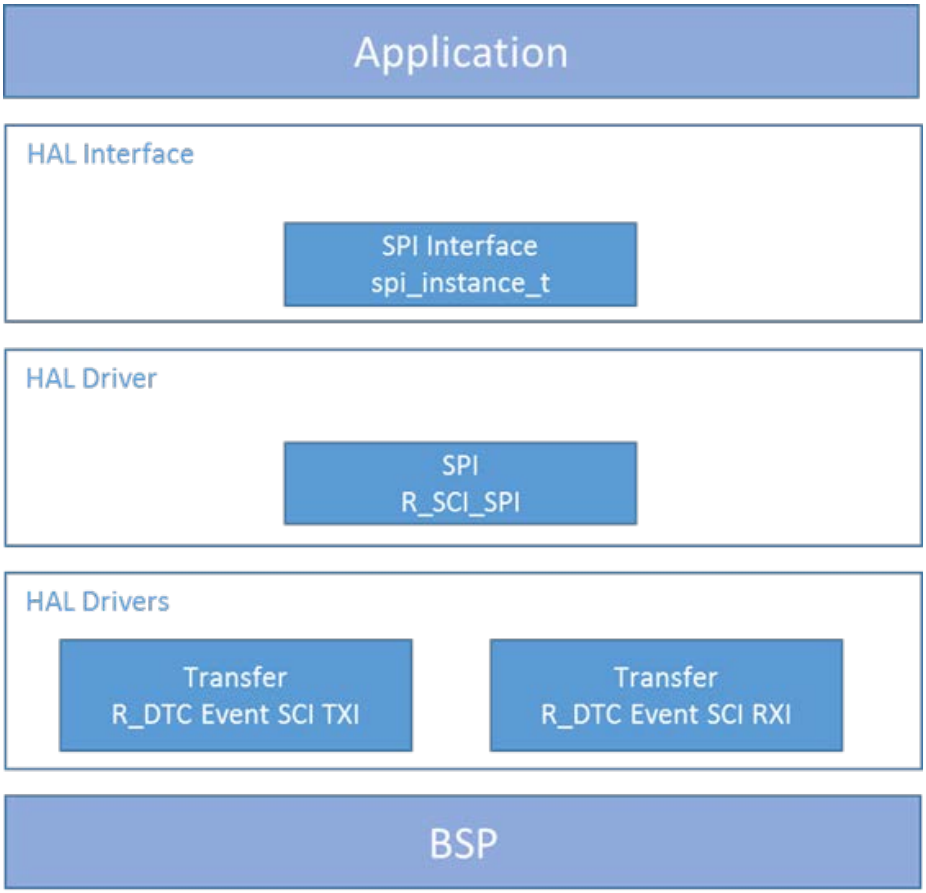


Figure 13.15 Serial Communication Interface SPI

13.16.2 Estimated Memory Requirements

Table 13.29 Memory Usage for Serial Communication Interface SPI (SCI_SPI) – GCC Compiler

r_sci_spi	Flash (Bytes)
S124 MCU Group	5,049
S3A7 MCU Group	5,070
S5D9 MCU Group	5,102
S7G2 MCU Group	5,102

Table 13.30 Memory Usage for Serial Communication Interface SPI (SCI_SPI) – IAR Compiler

r_sci_spi	Flash (Bytes)
S124 MCU Group	4,712
S3A7 MCU Group	4,676
S5D9 MCU Group	4,676
S7G2 MCU Group	4,676

13.16.3 SSP Supported Hardware Features: SCI_SPI

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

	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
S3A7	✓	✓	GPIO	N/A	✓	N/A
S5D9 or S7G2	✓	✓	GPIO	N/A	✓	N/A

	Overrun error detection	Data format transfer bit length	Master and Slave Mode selectable	Clock source internal/external	Configurable phase and polarity	Interrupt sources support
S124	✓	8-bit	Master	internal	✓	✓
S3A7	✓	8-bit	Master	internal	✓	✓
S5D9 or S7G2	✓	8-bit	Master	internal	✓	✓

13.17 JPEG Codec (JPEG Codec)

13.17.1 Component Introduction

The on-chip JPEG engine performs high-speed image data compression and decoding of JPEG image data. The JPEG Decoder conforms to the JPEG baseline decompression standard, JPEG Part 2, ISO-IEC10918-2. The JPEG Codec HAL driver module supports the JPEG hardware peripheral in Synergy MCUs.

Specifications and Features:

- Supports JPEG decompression for applications to convert a JPEG image into bitmap data suitable for display frame buffer.
- Supports polling mode that allows an application to wait for JPEG decoder to complete.
- Supports Interrupt mode with user-supplied callback functions.
- Provides interfaces for applications to specify parameters such as:
  - Horizontal and vertical subsample values
  - Horizontal stride
  - Decoded pixel format
  - Input and output data format
  - Color space.
- Obtains the size of image prior to the decoding step.
- Supports putting encoded data in an input buffer and an output buffer to store the decoded image frame.
- Streams encoded data input into JPEG Decoder module. This feature allows an application to read encoded JPEG image from a file or from network while decoding it, without the necessity to buffer the entire image.
- 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 formats YCbCr444, YCbCr422, YCbCr420, YCbCr411.
- Supports the output decoded formats ARGB8888 and RGB565.
- Returns error when JPEG image's size height and width do not match the specified input values.

The JPEG Codec HAL module implements the JPEG Decode interface in SSP:

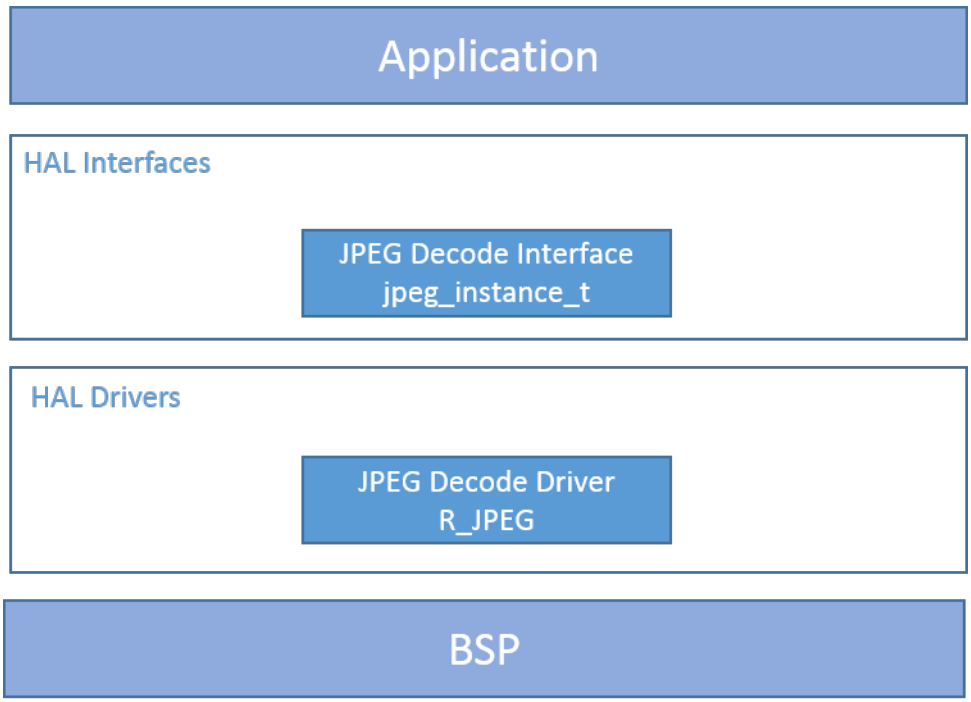


Figure 13.16 JPEG Codec

Estimated Memory Requirements:

Table 13.31 Memory Usage for JPEG Codec (JPEG Codec) – GCC Compiler

r_jpeg_decode	Flash (Bytes)
S5D9 MCU Group	2,213
S7G2 MCU Group	3,198

Table 13.32 Memory Usage for JPEG Codec (JPEG Codec) – IAR Compiler

r_jpeg_decode	Flash (Bytes)
S5D9 MCU Group	1,980
S7G2 MCU Group	2,684

13.17.2 SSP Supported Hardware Features: JPEG

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

	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
S3A7	N/A	N/A	N/A	N/A	N/A	N/A
S5D9 or S7G2	✓	✓	✓	✓	✓	✓

13.18 Flash Memory-High Performance (FLASH_HP)

13.18.1 Component Introduction

The Flash memory module supports the following features:

- The S7G2 Group microcontrollers support up to 4 MB high-speed Code flash for user applications and 64 KB of high-speed Data Flash for storing data. The Flash Memory-High Performance HAL driver supports the High Performance Flash memory block on S7G2 Group MCU and enables 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 functionality available through the module is listed below: Blocking erasing, reading, writing and blank checking of ROM flash.
- Non-Blocking erasing, reading, writing and blank checking of Data flash.
- Blocking erasing, reading, writing and blank checking of Data and 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.
- Swap area for boot block swapping which allows safe re-writing of the startup program without first erasing it.
- Capability to query the flash memory device to identify capabilities such as number of flash regions, minimum erase and write sizes for code and data flash

The driver makes the process of programming and erasing on-chip flash areas easy. The module can be used to perform blocking erase and program operations for both code and data flash, with BGO operation available for data flash operations only. When a code flash operation is on-going, you cannot access that code flash area. If there's an attempt to access the code flash area while a code Flash operation is in progress, the flash control unit will transition into an error state.

The FLASH_HP Modules implements the Flash Interface in SSP.

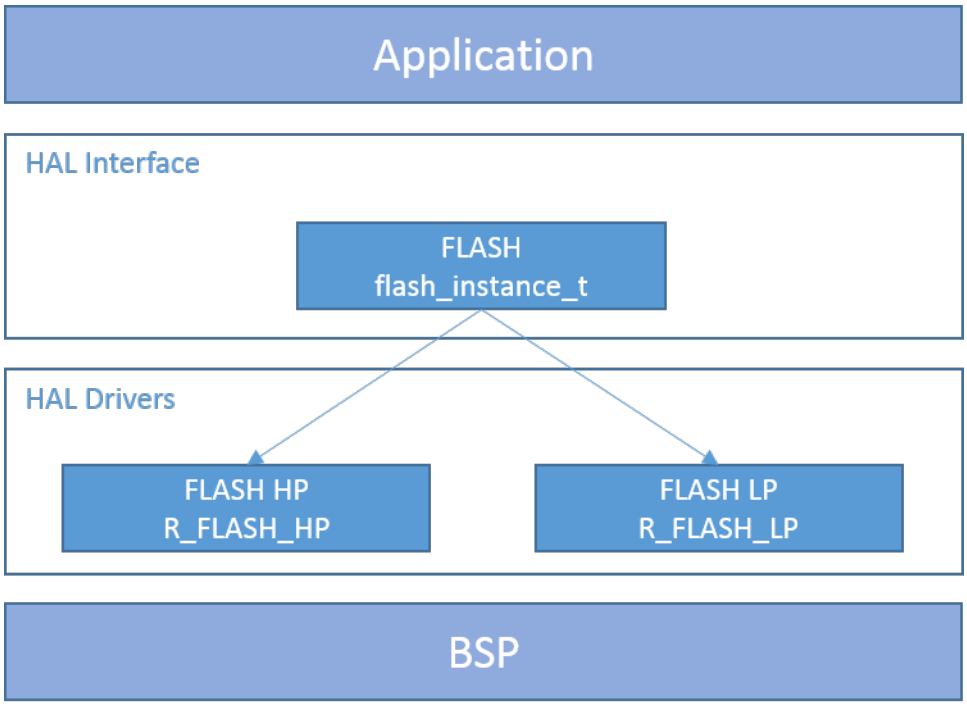


Figure 13.17 Flash Memory-High Performance

13.18.2 Estimated Memory Requirements

Table 13.33 Memory Usage for Flash Memory-High Performance (FLASH_HP) – GCC Compiler

r_flash_hp	Flash (Bytes)
S5D9 MCU Group	3,660
S7G2 MCU Group	3,620

Table 13.34 Memory Usage for Flash Memory-High Performance (FLASH_HP) – IAR Compiler

r_flash_hp	Flash (Bytes)
S5D9 MCU Group	3,340
S7G2 MCU Group	3,320

13.18.3 SSP Supported Hardware Features: Flash Memory-High Performance (FLASH_HP)

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

	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
S3A7	N/A	N/A	N/A
S5D9 or S7G2	✓	✓	✓

13.19 Flash Memory-Low Power (FLASH_LP)

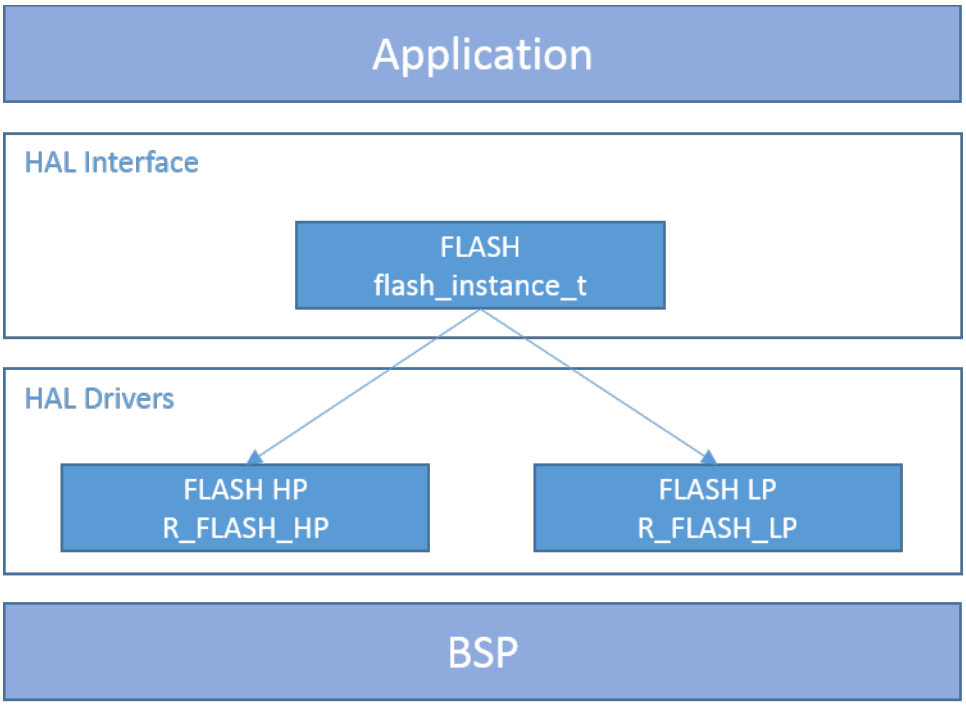
13.19.1 Component Introduction

The S3A7 Group microcontrollers support up to 1 MB low-power Code flash memory for user applications and 16 KB of low power Data Flash memory for storing data. The Flash Memory-Low Power HAL driver supports the Low Power Flash memory block on S3A7 MCU and enables 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 functionality available through the module is listed below:

- Non-Blocking erasing, reading, writing and blank checking of Data flash.
- Blocking erasing, reading, writing and blank checking of Data and 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.
- Swap area for boot block swapping which allows safe re-writing of the startup program without first erasing it.
- Capability to query the flash memory device to identify capabilities such as number of flash regions, minimum erase and write sizes for code and data

The driver makes the process of programming and erasing on-chip flash areas easy. The module can be used to perform blocking erase and program operations for both code and data flash, with BGO (Background Operation) available for data flash operations only. When a code flash operation is on-going, you cannot access that code flash area. If there's an attempt to access the code flash area while a code Flash operation is in progress, the flash control unit will transition into an error state.

The FLASH_LP Modules implements the Flash Interface in SSP.



13.19.2 Estimated Memory Requirements

Table 13.35 Memory Usage for Flash Memory-Low Power (FLASH_LP) – GCC Compiler

r_flash_lp	Flash (Bytes)
S124 MCU Group	3,860
S3A7 MCU Group	3,604

Table 13.36 Memory Usage for Flash Memory-Low Power (FLASH_LP) – IAR Compiler

r_flash_lp	Flash (Bytes)
S124 MCU Group	3,380
S3A7 MCU Group	3,272

13.19.3 SSP Supported Hardware Features: Flash Memory-Low Power (FLASH_LP)

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

	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	✓	✓	✓
S3A7	✓	✓	✓
S5D9 or S7G2	N/A	N/A	N/A

13.20 Data Transfer Controller (DTC)

13.20.1 Component Introduction

Data Transfer Controller (DTC) driver supports the DTC peripheral that is used to transfer data between memory and peripherals, or between two peripherals without CPU intervention (background data transfer). The DTC driver moves

data from a user specified source to a user specified destination when an interrupt or event occurs. The DTC module uses a RAM based vector table, with slots for every interrupt in the system. When the DTC transfer completes, the activation source interrupt is called. The activation source interrupt must be enabled to use the DTC. The activation source interrupt is generally muted by the DTC until the transfer completes, unless TRANSFER_IRQ_EACH is specified in the configuration. The DTC also allows chained transfers, meaning that more than one transfer can occur after a single activation source interrupt. This feature is supported by the driver but must be configured outside the e^2 studio IDE.

The Data Transfer Controller allows data transfers to occur in place of or in addition to any interrupt. It does not support data transfers using software start.

The DTC module supports following transfer modes:

- Normal Mode - A single transfer is triggered each time an activation source event occurs. A single transfer is 1 byte, 2 bytes, or 4 bytes depending on the selected settings. Total length (size) of data to be transferred is configurable.
- Repeat Mode – In addition to the length (size) of data block that needs to be transferred, Repeat Mode provides additional provision for specifying number of time transfer should be repeated with the same length of data. In this mode if the repeat area is set to source, the same source location is used for each transfer iteration. Alternatively, if the repeat area is set to destination, the same destination location is used for each transfer iteration.
- Block Mode – The block mode transfer operates similar to the Repeat mode, but is triggered by a source event, the entire transfer length is transferred each time an activation source event occurs. For example, if a transfer is configured in block mode with timer as the activation source, a 2 byte size, and a 12 byte length, 24 bytes are transferred each time the activation source event occurs. Similar to Block Mode if the repeat area is set to source, the source register is reloaded with its initial value when the transfer restarts.
- Address Mode – The Address mode transfer operates similar to the Normal Mode, but after each transfer the source pointer and destination pointer is incremented by the length of transfer
- Chained Transfer Mode - Chained transfers are only supported by DTC, in this mode successive transfers are linked by creating an array of Transfer Info structures and setting the mode TRANSFER_CHAIN_MODE_ENABLED for all transfers except the last transfer. The module is configured to point to the base of the first structure in the array to indicate the first transfer source and destination location.

The HAL Transfer Interface is a generic interface for Transfer applications and is implemented by two data transfer modules in SSP, DMAC and DTC.

13.20.2 Estimated Memory Requirements

Table 13.37 Memory Usage for Data Transfer Controller (DTC) – GCC Compiler

r_dtc	Flash (Bytes)
S124 MCU Group	2,542
S3A7 MCU Group	2,426
S5D9 MCU Group	2,426
S7G2 MCU Group	2,426

Table 13.38 Memory Usage for Data Transfer Controller (DTC) – IAR Compiler

r_dtc	Flash (Bytes)
S124 MCU Group	2,380
S3A7 MCU Group	2,404
S5D9 MCU Group	2,552
S7G2 MCU Group	2,552

13.20.3 SSP Supported Hardware Features: DTC

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

	Normal Transfer Mode	Repeat Transfer Mode	Block Transfer Mode	Selectable Data Transfer Units {8 bits, 16 bits, 32 bits}	Event link function
S124	✓	✓	✓	✓	☒
S3A7	✓	✓	✓	✓	☒
S5D9 or S7G2	✓	✓	✓	✓	☒

13.21 Data Operation Circuit (DOC)

13.21.1 Component Introduction

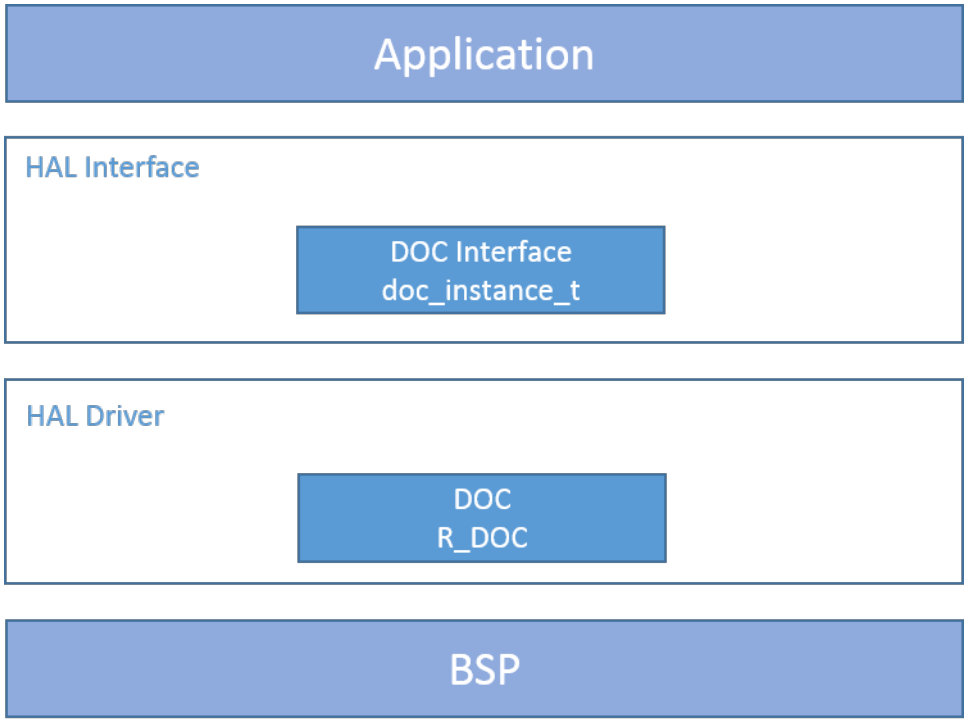
The Data Operation Circuit (DOC) peripheral performs a 16-bit addition, subtraction, and comparison without CPU intervention. The DOC driver provided with SSP supports the DOC peripheral available on the Synergy microcontroller hardware and controls the peripheral according to user configuration.

The driver can detect the following events:

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

When the configured event occurs and a callback is available (with interrupts enabled), the driver invokes the callback with the supplied arguments, which in turn indicates the occurrence of the event to application. If interrupts are not enabled, the API supports checking the DOC status to poll the status of the comparison, addition or subtraction operation.

The DOC driver implements the DOC interface in SSP.



13.21.2 Estimated Memory Requirements

Table 13.39 Memory Usage for Data Operation Circuit (DOC) – GCC Compiler

r_doc	Flash (Bytes)
S124 MCU Group	1,229
S3A7 MCU Group	1,097
S5D9 MCU Group	1,093
S7G2 MCU Group	1,093

Table 13.40 Memory Usage for Data Operation Circuit (DOC) – IAR Compiler

r_doc	Flash (Bytes)
S124 MCU Group	1,000
S3A7 MCU Group	936
S5D9 MCU Group	936
S7G2 MCU Group	936

13.21.3 SSP Supported Hardware Features: DOC

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

	Module-stop function	Output to the Event Link Controller
S124	✓	☒
S3A7	✓	☒
S5D9 or S7G2	✓	☒

13.22 Direct Memory Access Controller (DMAC)

13.22.1 Component Introduction

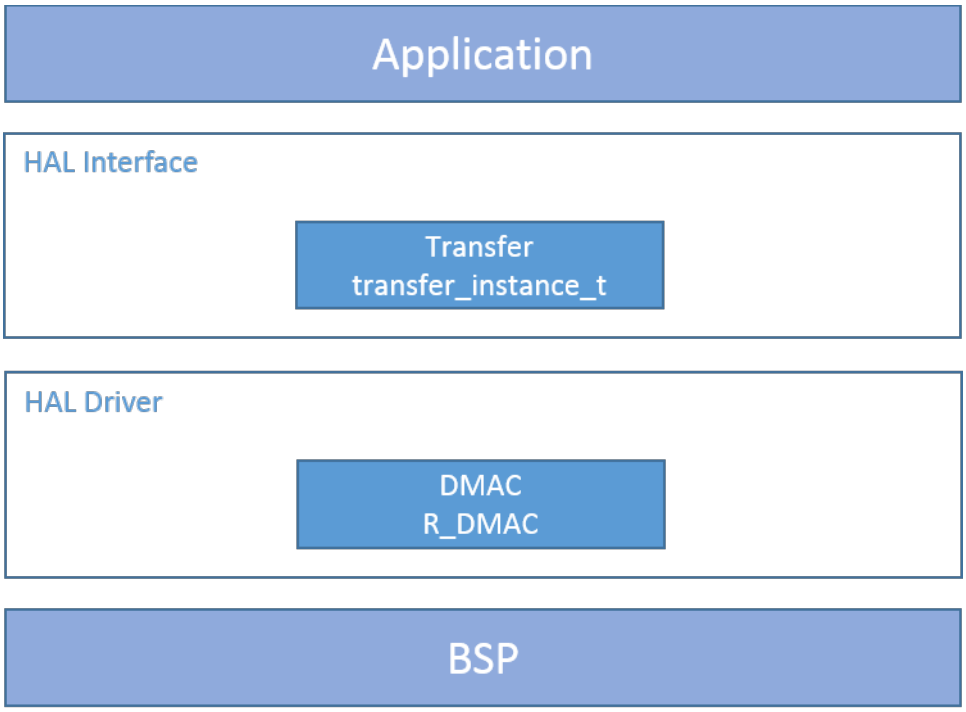
The Direct Memory Access Controller (DMAC) driver supports the DMAC peripheral that is used to transfer data between memory and peripherals, or between two peripherals without CPU intervention (background data transfer). The DMAC driver moves data from a user specified source to a user specified destination when an interrupt or event occurs. The DMAC module uses DMAC peripheral registers, so the number of transfers in the system is limited to number of available DMAC channels on the device. The activation source does not have to be enabled to use the DMAC. When the DMAC transfer completes, a DMAC interrupt is called. If the activation source interrupt is enabled, it fires at the same time the transfer is triggered. If the DMAC interrupt is enabled, it fires after all transfers are complete. The DMAC does not support chained transfers.

The DMAC module supports following transfer modes:

- Normal Mode – A single transfer is triggered each time an activation source event occurs. A single transfer is 1 byte, 2 bytes, or 4 bytes depending on the selected settings. Total length (size) of data to be transferred is configurable.
- Repeat Mode – In addition to the length (size) of data block that needs to be transferred, Repeat Mode provides additional provision for specifying number of time transfer should be repeated with the same length of data. In this mode if the repeat area is set to source, the same source location is used for each transfer iteration. Alternatively, if the repeat area is set to destination, the same destination location is used for each transfer iteration.
- Block Mode – The block mode transfer operates similar to the Repeat mode, but is triggered by a source event, the entire transfer length is transferred each time an activation source event occurs. For example, if a transfer is configured in block mode with timer as the activation source, a 2 byte size, and a 12 byte length, 24 bytes are transferred each time the activation source event occurs. Similar to Block Mode if the repeat area is set to source, the source register is reloaded with its initial value when the transfer restarts.

- Address Mode – The Address mode transfer operates similar to the Normal Mode, but after each transfer the source pointer and destination pointer is incremented by the length of transfer.

The HAL Transfer Interface is a generic interface for Transfer applications and is implemented by two data transfer modules in SSP, DMAC and DTC.



13.22.2 Estimated Memory Requirements

Table 13.41 Memory Usage for Direct Memory Access Controller (DMAC) – GCC Compiler

r_dmac	Flash (Bytes)
S3A7 MCU Group	2,644
S5D9 MCU Group	2,644
S7G2 MCU Group	2,556

Table 13.42 Memory Usage for Direct Memory Access Controller (DMAC) – IAR Compiler

r_dmac	Flash (Bytes)
S3A7 MCU Group	2124
S5D9 MCU Group	2,124
S7G2 MCU Group	2,124

13.22.3 SSP Supported Hardware Features: DMAC

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

	Normal Transfer Mode	Repeat Transfer Mode	Block Transfer Mode	Extended repeat area function	Event link function
S124	✓	✓	✓	✓	☒
S3A7	✓	✓	✓	✓	☒
S5D9 or S7G2	✓	✓	✓	✓	☒

13.23 Interrupt Controller Unit (ICU)

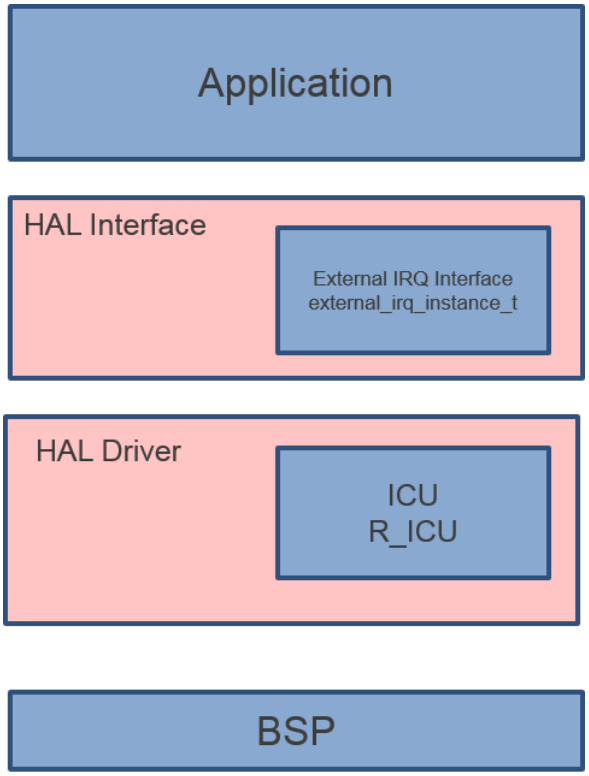
13.23.1 Component Introduction

The External IRQ HAL driver supports the Interrupt Controller Unit (ICU) on Synergy Microcontroller hardware for external pin interrupts used by push-button devices and other applications using external interrupts.

The external IRQ HAL driver supports external inputs, for example, input from pins or capacitive touch buttons. When an input trigger is detected, a user provided callback function will be called. The driver configures the external IRQ inputs in the ICU (Interrupt Controller Unit). The driver supports the following features of the external IRQ inputs:

- Enabling and disabling generation of an interrupt
- Calling of a callback when the IRQ event occurs
- Enabling and disabling IRQ noise filter

The ICU driver module implements the External IRQ interface in SSP.



13.23.2 Estimated Memory Requirements

Table 13.43 Memory Usage for Interrupt Controller Unit (ICU) – GCC Compiler

r_icu	Flash (Bytes)
S124 MCU Group	1,281
S3A7 MCU Group	1,209
S5D9 MCU Group	1,209
S7G2 MCU Group	1,209

Table 13.44 Memory Usage for Interrupt Controller Unit (ICU) – IAR Compiler

r_icu	Flash (Bytes)
S124 MCU Group	1,104
S3A7 MCU Group	1,040
S5D9 MCU Group	1,040
S7G2 MCU Group	1,040

13.23.3 SSP Supported Hardware Features: ICU

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

	Peripheral function interrupts	External pin interrupts	DTC and DMAC control	Interrupt sources for NVIC	Non-maskable interrupts	Return from low-power mode
S124	✓	✓	✓	✓	✓ (See Note below)	✓ (See Note below)
S3A7	✓	✓	✓	✓	✓ (See Note below)	✓ (See Note below)
S5D9 or S7G2	✓	✓	✓	✓	✓ (See Note below)	✓ (See Note below)

Notes:

- The ICU module in SSP (r_icu) only handles External pin interrupts but not for other features above.
- Peripheral function intrrupts 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-maskarable 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) recpectively.
- Wake Up Interrupt Enable setting is supported by LVD module (r_lvd). In terms of the low power mode, refer to the LPM section for more detail.

13.24 Event Link Controller (ELC)

13.24.1 Component Introduction

The ELC Controller uses event requests to link different peripherals together without CPU intervention.

The ELC module supports the following functions:

- Create an event link between two blocks.
- Break that event link between two blocks.
- Generate one of two software events which interrupt the CPU.

ELC driver module implements the ELC interface in SSP:

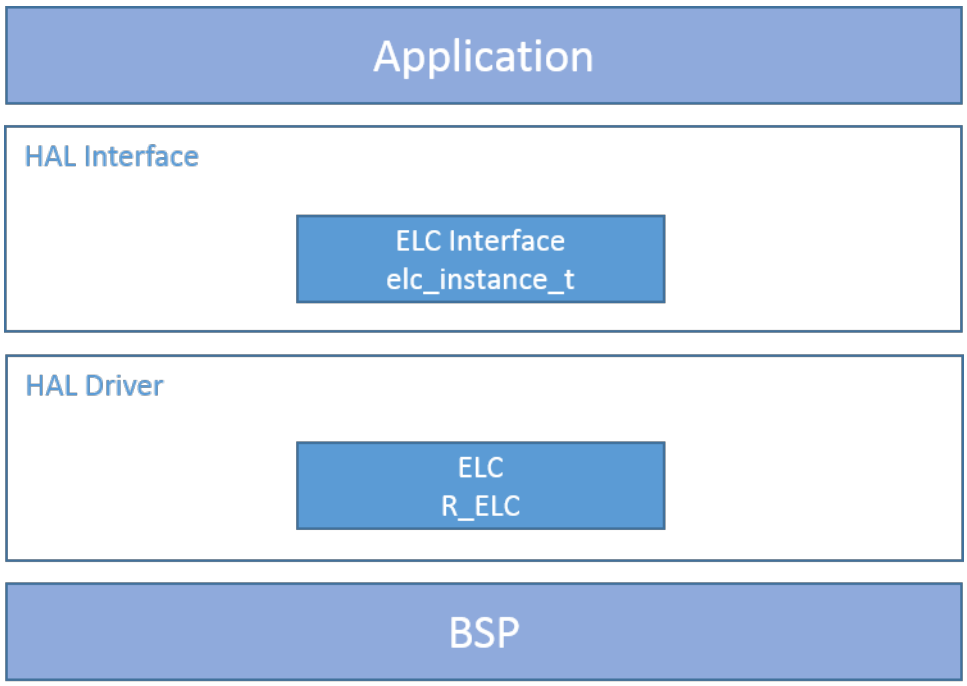


Figure 13.18 Event Link Controller

13.24.2 Estimated Memory Requirements

Table 13.45 Memory Usage for Event Link Controller (ELC) – GCC Compiler

r_elc	Flash (Bytes)
S124 MCU Group	429
S3A7 MCU Group	405
S5D9 MCU Group	405
S7G2 MCU Group	405

Table 13.46 Memory Usage for Event Link Controller (ELC) – IAR Compiler

r_elc	Flash (Bytes)
S124 MCU Group	372
S3A7 MCU Group	392
S5D9 MCU Group	392
S7G2 MCU Group	392

13.24.3 SSP Supported Hardware Features: 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

	Create	Break	Generate
S124	✓	✓	✓
S3A7	✓	✓	✓
S5D9 or S7G2	✓	✓	✓

13.25 General Purpose Timer (GPT)

13.25.1 Component Introduction

The GPT driver module supports the GPT peripheral in Synergy Microcontroller. The driver configures a 32 bit timer to a user specified period, when the period elapses, the GPT module can call a user callback and toggle a port pin.

The GPT driver provides standard timer functionality including periodic mode, one-shot mode, and free-running timer mode. After each timer cycle (overflow or underflow), an interrupt can be triggered. The driver configures a timer to a user specified period. When the period elapses, any of the following user configured events can be triggered:

- Interrupt the CPU, which will call 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 ELC Interface.

The driver also provides an output compare extension to output the timer signal to the GTIOC pin.

The HAL Timer Interface in SSP is a generic interface for timer applications and is implemented by the AGT and GPT driver modules.

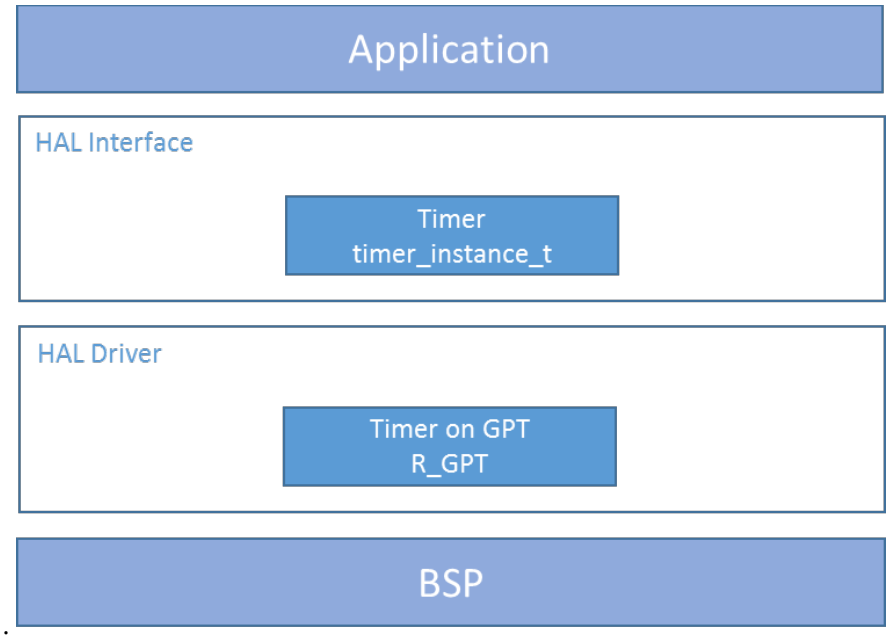


Figure 13.19 General Purpose Timer

13.25.2 Estimated Memory Requirements

Table 13.47 Memory Usage for General Purpose Timer (GPT) – GCC Compiler

r_gpt	Flash (Bytes)
S124 MCU Group	3,265
S3A7 MCU Group	2,977
S5D9 MCU Group	3,037
S7G2 MCU Group	3037

Table 13.48 Memory Usage for General Purpose Timer (GPT) – IAR Compiler

r_gpt	Flash (Bytes)
S124 MCU Group	2,892
S3A7 MCU Group	2,660
S5D9 MCU Group	2,704
S7G2 MCU Group	2,704

13.25.3 SSP Supported Hardware Features: GPT

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

	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	☒	✓	☒	✓	☒	☒
S3A7	☒	✓	☒	✓	☒	☒
S5D9 or S7G2	☒	✓	☒	✓	☒	☒

	PWM Mode	Phase Count Function	One-Shot Operation	Event linking (ELC) function	Noise filtering function
S124	✓	☒	✓	☒	☒
S3A7	✓	☒	✓	☒	☒
S5D9 or S7G2	✓	☒	✓	☒	☒

13.26 General Purpose I/O Port (GPIO / IOPORT)

13.26.1 Component Introduction

The IOPORT module supports the I/O Ports peripheral available on the Synergy microcontroller hardware. The driver configures one or more I/O pins. The direction of the pin or pins can be configured along with following options:

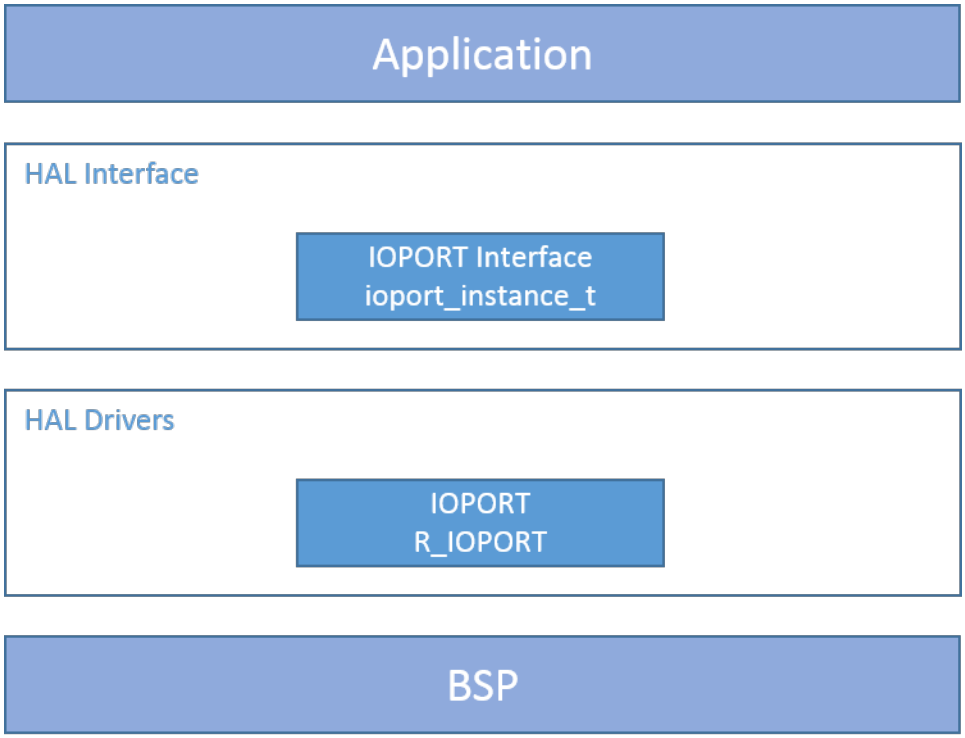
- Pull-up
- NMOS/PMOS
- Drive strength
- Event edge trigger (falling, rising or both)
- Whether the pin is to be used as an IRQ pin
- Whether the pin is to be used as an analog pin
- Whether the pin is to be used as a peripheral pin and which peripheral

The driver supports the following features:

- Change direction of one or more pins on a port.
- Write to one or more pins on a port.
- Read from one or more pins on a port.
- Set event output data.
- Read event input data.

The IOPORT HAL drivers provide ability to access the I/O Ports of a device at both bit and port level. Port and pin direction can be changed. In addition a number of configuration APIs are provided to change the functionality of individual pins.

The IOPRT driver implements the IOPORT interface in SSP.



13.26.2 Estimated Memory Requirements

Table 13.49 Memory Usage for General Purpose I/O Port (GPIO / IOPORT) – GCC Compiler

r_ioport	Flash (Bytes)
S124 MCU Group	2,136
S3A7 MCU Group	2,230
S5D9 MCU Group	2,230
S7G2 MCU Group	2,230

Table 13.50 Memory Usage for General Purpose I/O Port (GPIO / IOPORT) – IAR Compiler

r_ioport	Flash (Bytes)
S124 MCU Group	1,554
S3A7 MCU Group	1,686
S5D9 MCU Group	1,686
S7G2 MCU Group	1,686

13.26.3 SSP Supported Hardware Features: GPIO

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

	Port Direction Setting	Intput Data Read function	Output Port Write function	Pin Mode Control	Ethernet Mode Configration	ELC_PORTn Event Intput Read function
S124	✓	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓	✓
S5D9 or S7G2	✓	✓	✓	✓	✓	✓

	ELC_PORTn Event Output Setting
S124	✓
S3A7	✓
S5D9 or S7G2	✓

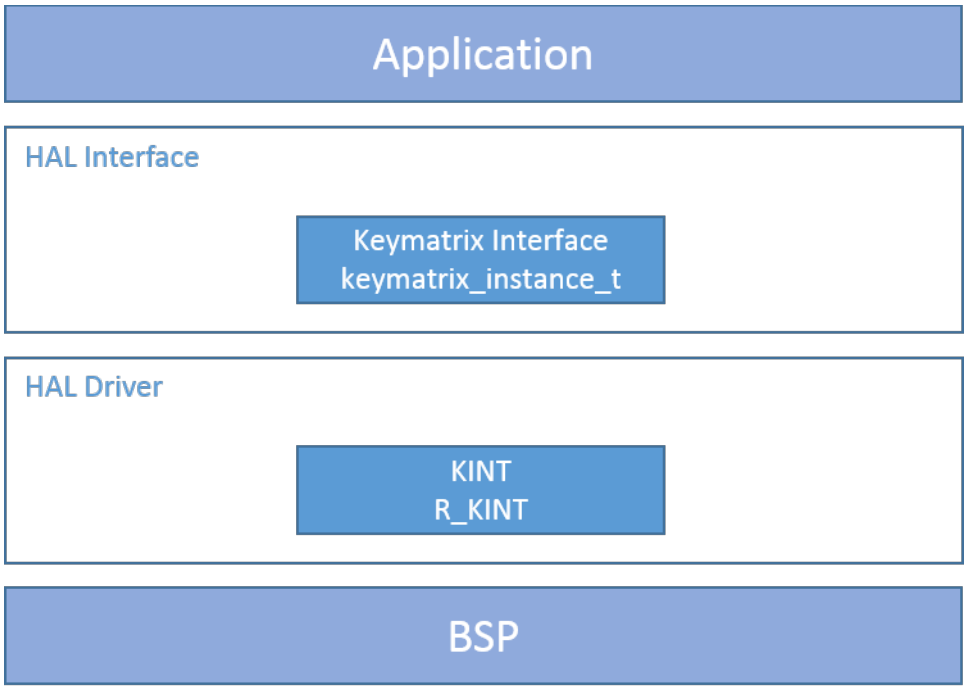
13.27 Keyboard Interrupt Interface (KINT)

13.27.1 Component Introduction

The Keyboard interrupt interface supports the Key Interrupt Function peripheral available on the Synergy microcontroller hardware. The Key input driver can be used for one to eight channels or in a matrix format. This module implements the Key Matrix Interface in SSP.

The Key Interrupt (KINT) driver detects rising or falling edges on any of the KINT channels. When such an event is detected on any of the configured pins, the module generates an interrupt. The interrupt then calls the user callback that specifies the channel(s) on which the edge was detected via a bitmask. Even though detection of an edge on any one channel generates the interrupt, the callback returns a bit-mask of all the pins that were triggered at that time if any other pins also detected an edge. Thus an interrupt is not necessarily generated for edge detection on each pin if an edge was detected on another pin also before the callback was called. If a new edge is detected after the callback was called, then the interrupt is triggered again resulting in a new callback.

This module can be used to implement a matrix keypad with edges on any two channels indicating the actual key that was pressed. Alternatively, the module can be used as a single input to detect an edge on an input pin.



13.27.2 Estimated Memory Requirements

Table 13.51 Memory Usage for Keyboard Interrupt Interface (KINT) – GCC Compiler

r_kint	Flash (Bytes)
S124 MCU Group	1,285
S3A7 MCU Group	1,145
S5D9 MCU Group	1,145
S7G2 MCU Group	1,145

Table 13.52 Memory Usage for Keyboard Interrupt Interface (KINT) – IAR Compiler

r_kint	Flash (Bytes)
S124 MCU Group	1,020
S3A7 MCU Group	1,020
S5D9 MCU Group	1,020
S7G2 MCU Group	1,020

13.27.3 SSP Supported Hardware Features: KINT

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

	Vary Input from KR00 to KR07
S124	✓
S3A7	✓
S5D9 or S7G2	✓

13.28 Graphics LCD Controller (GLCD)

13.28.1 Component Introduction

The microcontroller features a highly configurable, integrated Graphics LCD Controller that can be used to drive a variety of color TFT LCD screens. The GLCD controller reads image data from system memory, displays it on an LCD panel connected to GLCD interface, and frees up the CPU for other processing tasks.

GLCD controller provides standard display functionality:

- Signal timing configuration for LCD panels with RGB interface.
- Dot clock source selection (internal or external) and frequency divider.
- Blending of multiple graphics layers on the background screen.
- Color correction (brightness/configuration/gamma correction).
- Interrupts and callback function.

GLCD controller supports following features:

- Supports LCD panels with RGB interface (up to 24bits) and sync signals (HSYNC, VSYNC and Data Enable)
- Supports various color formats for input graphics planes (RGB888, ARGB888, RGB565, ARGB1555, ARGB4444, CLUT8, CLUT4, CLUT1).
- Supports CLUT (Color Look-Up Table) usage for input graphics planes with 512 words (32bits/word).
- Supports various color formats for output (RGB888, RGB666, RGB565, Serial RGB).
- 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).
- Supports brightness adjustment, contrast adjustment and gamma correction.
- Supports GLCDC interrupts to handle frame buffer switching or underflow detection.

The figure below shows an overview of the graphics data flow using the GLCDC driver module. The driver supports reading graphics frame image data from memory (up to two frames) and blending those images on top of the monochrome background screen. The driver supports CLUT memory and specifies the graphic frame format for the CLUT.

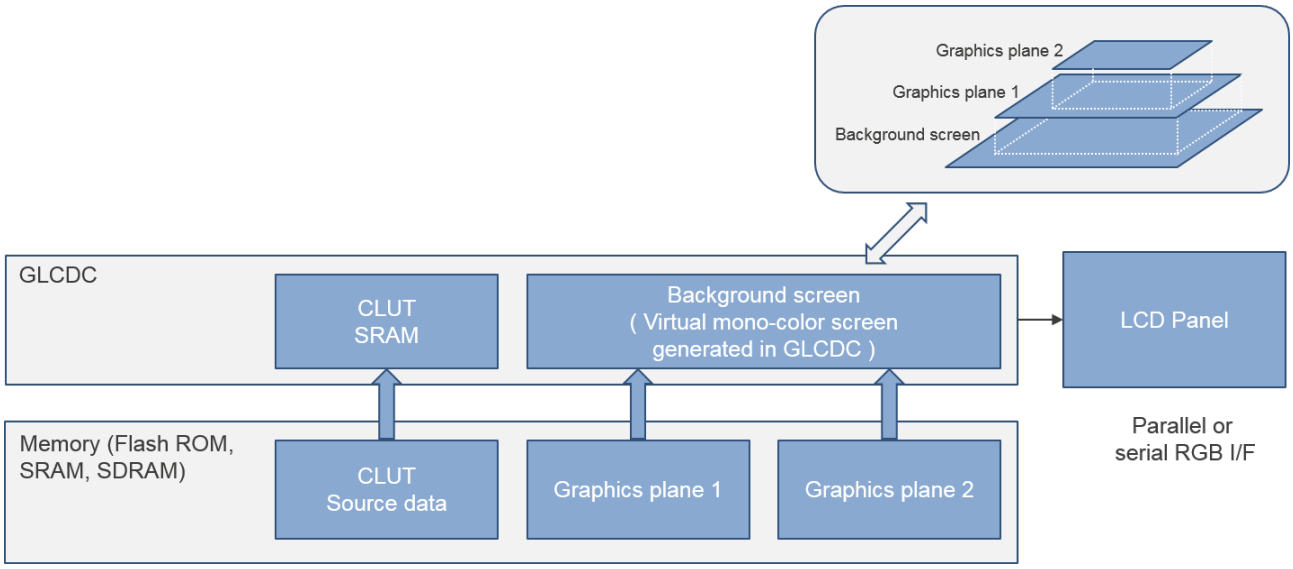
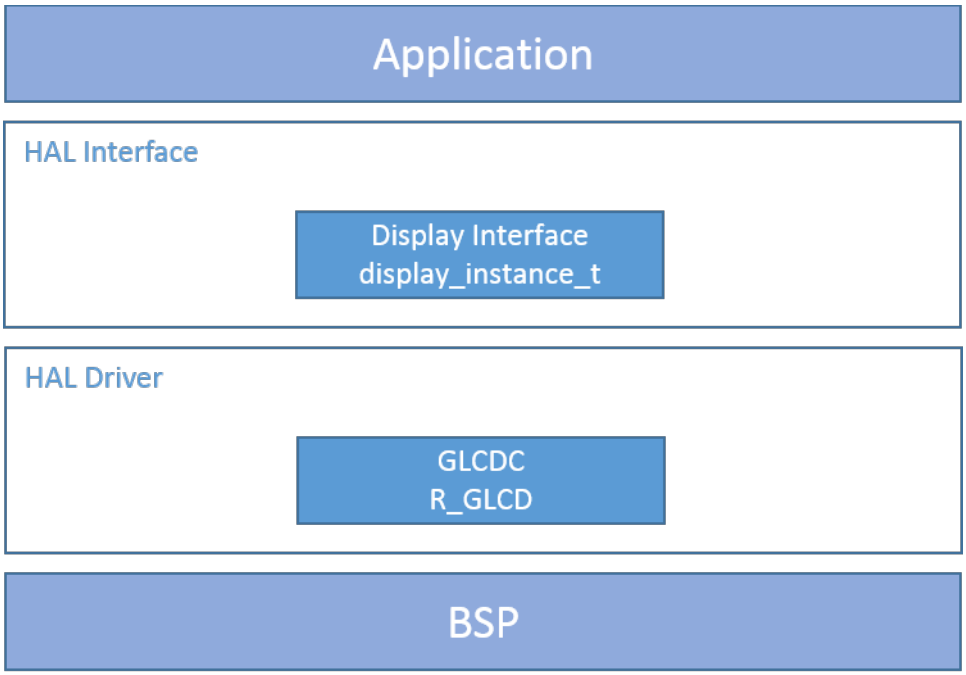


Figure 13.20 GCLD Controller Data Flow

The GLCD Controller drivers implements the HAL Display interface in SSP.



13.28.2 Estimated Memory Requirements

Table 13.53 Memory Usage for Graphics LCD Controller (GLCD) – GCC Compiler

r_glcd	Flash (Bytes)
S5D9 MCU Group	7,723
S7G2 MCU Group	7,723

Table 13.54 Memory Usage for Graphics LCD Controller (GLCD) – GCC Compiler

r_glcd	Flash (Bytes)
S5D9 MCU Group	5,724
S7G2 MCU Group	5,724

13.28.3 SSP Supported Hardware Features: GLCD

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

	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
S3A7	N/A	N/A	N/A	N/A	N/A	N/A
S5D9 or S7G2	✓	✓	✓	✓	✓	✓

	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
S3A7	N/A	N/A	N/A	N/A	N/A	N/A
S5D9 or S7G2	✓	✓	✓	✓	✓	✓

	Supports All Dithering Modes	Support output of VSYNC, HSYNC and Horizontal Data Enable	Supports Brightness and Contrast	Supports Gamma Correction
S124 or S128	N/A	N/A	N/A	N/A
S3A7	N/A	N/A	N/A	N/A
S5D9 or S7G2	✓	✓	✓	✓

13.29 Watchdog Timer (WDT)

13.29.1 Component Introduction

WDT driver in SSP supports the WDT peripheral on Synergy MCUs. This driver configures the Watchdog Timer (WDT) Interface and when the WDT underflows or is refreshed outside of the permitted refresh window, one of the following events will occur based on the configuration:

- Resetting of the device
- Generation of an NMI

WDT driver provides the ability to configure the operation of WDT (when used in register start mode), refresh the watchdog, read the timer value and read and clear status flags.

The WDT and IWDT drivers implements the WDT HAL interface in SSP:

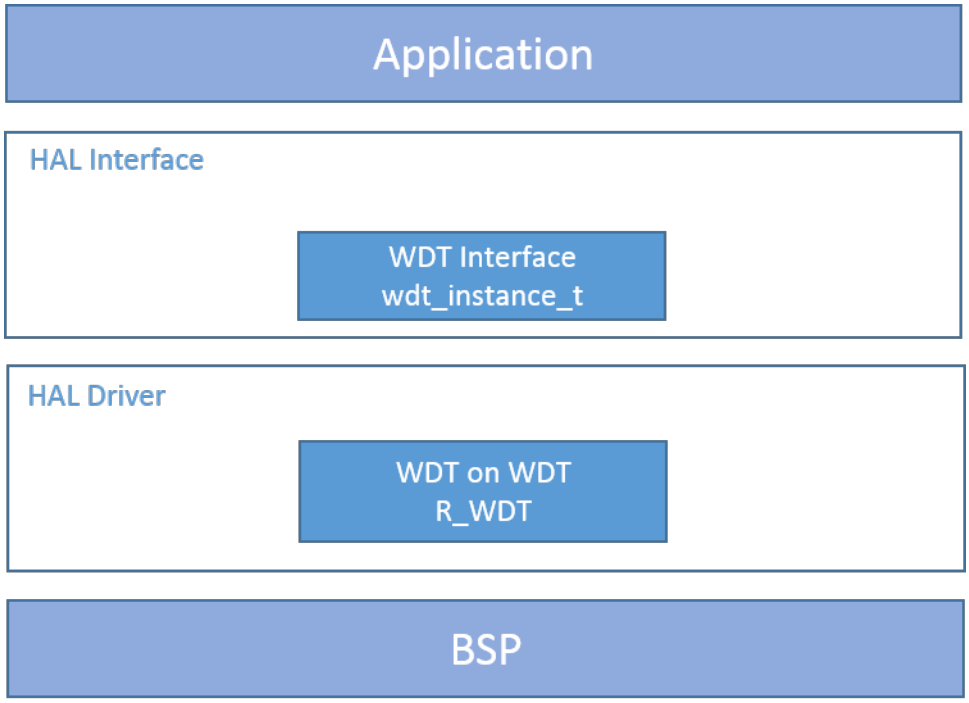


Figure 13.21 Watchdog Timer

13.29.2 Estimated Memory Requirements

Table 13.55 Memory Usage for Watchdog Timer (WDT) – GCC Compiler

r_wdt	Flash (Bytes)
S124 MCU Group	1,148
S3A7 MCU Group	1,112
S5D9 MCU Group	1,112
S7G2 MCU Group	1,112

Table 13.56 Memory Usage for Watchdog Timer (WDT) – IAR Compiler

r_wdt	Flash (Bytes)
S124 MCU Group	1,020
S3A7 MCU Group	1,012
S5D9 MCU Group	1,012
S7G2 MCU Group	1,012

13.29.3 SSP Supported Hardware Features: WDT

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

	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	✓	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓	✓
S5D9 or S7G2	✓	✓	✓	✓	✓	✓

	Sleep mode count stop control output	Event link function	Window function	Conditions for stopping the Counter – reset/underflow-refresh error	Refresh error and under flow error detect	Reading the counter value
S124	☒	☒	✓	✓	✓	✓
S3A7	☒	☒	✓	✓	✓	✓
S5D9 or S7G2	☒	☒	✓	✓	✓	✓

13.30 Independent Watchdog Timer (IWDT)

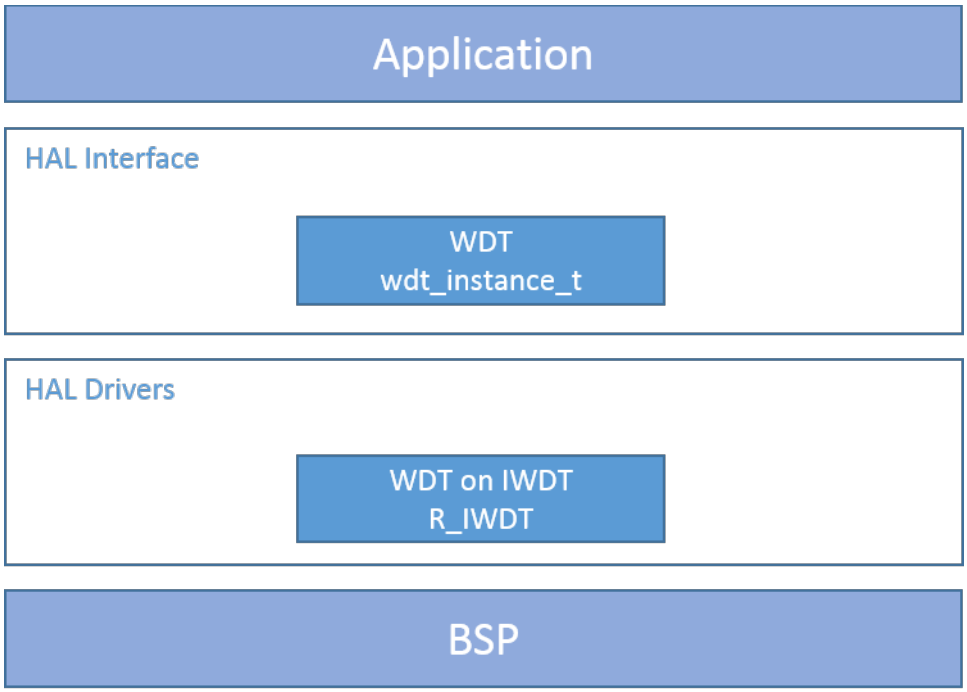
13.30.1 Component Introduction

The Independent Watchdog Timer (IWDT) peripheral in Synergy MCUs consists of a 14-bit down counter that must be serviced periodically to prevent counter underflow. The IWDT driver supports the IWDT peripheral on Synergy MCUs. This driver configures the Watchdog Timer (WDT) Interface and when the WDT underflows or is refreshed outside of the permitted refresh window, one of the following events will occur based on the configuration:

- Resetting of the device
- Generation of an NMI

The IWDT driver provides ability to refresh the independent watchdog, read the timer value and read and clear status flags. When used in NMI output mode the callback to be called by the NMI ISR can be registered.

The WDT and IWDT drivers implements the WDT HAL interface in SSP.



13.30.2 Estimated Memory Requirements

Table 13.57 Memory Usage for Independent Watchdog Timer (IWDT) – GCC Compiler

r_iwdt	Flash (Bytes)
S124 MCU Group	907
S3A7 MCU Group	887
S5D9 MCU Group	887
S7G2 MCU Group	887

Table 13.58 Memory Usage for Independent Watchdog Timer (IWDT) – IAR Compiler

r_iwdt	Flash (Bytes)
S124 MCU Group	732
S3A7 MCU Group	724
S5D9 MCU Group	724
S7G2 MCU Group	724

13.30.3 SSP Supported Hardware Features: 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

	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	✓	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓	✓
S5D9 or S7G2	✓	✓	✓	✓	✓	✓

	Sleep mode count stop control output	Event link function	Window function	Conditions for stopping the Counter – reset/underflow-refresh error	Refresh error and under flow error detect	Reading the counter value
S124	☒	☒	✓	✓	✓	✓
S3A7	☒	☒	✓	✓	✓	✓
S5D9 or S7G2	☒	☒	✓	✓	✓	✓

	Selecting the clock frequency division ratio after a reset	Selecting the timeout period of the independent watchdog timer
S124	☒	☒

13.31 Analog to Digital Converter (ADC)

13.31.1 Component Introduction

The HAL driver supports ADC12 and ADC14 peripherals available on the Synergy microcontroller hardware.

The ADC driver controls the ADC on a Synergy microcontroller according to the user configuration, it can access both ADC units on the MCU and configure them for single scan, continuous scan, and group scan modes. When a scan is complete and a callback is available (with interrupts enabled), the driver invokes the callback function.

If interrupts are not enabled, the driver checks the scan status to poll if the scan is complete and provides a function to read the converted ADC result.

The ADC driver also supports reading the data from the on-chip temperature sensor and using it for comparison.

Group Mode Operation: The driver also supports group mode operation. In this mode, channels can be assigned to one of two groups: group-A or group-B. A trigger is assigned for each group to start the scan. In group mode, only hardware triggers can be used, as opposed to normal mode, where software triggers or an external trigger can be used.

The ADC Driver implements the ADC HAL interface in SSP:

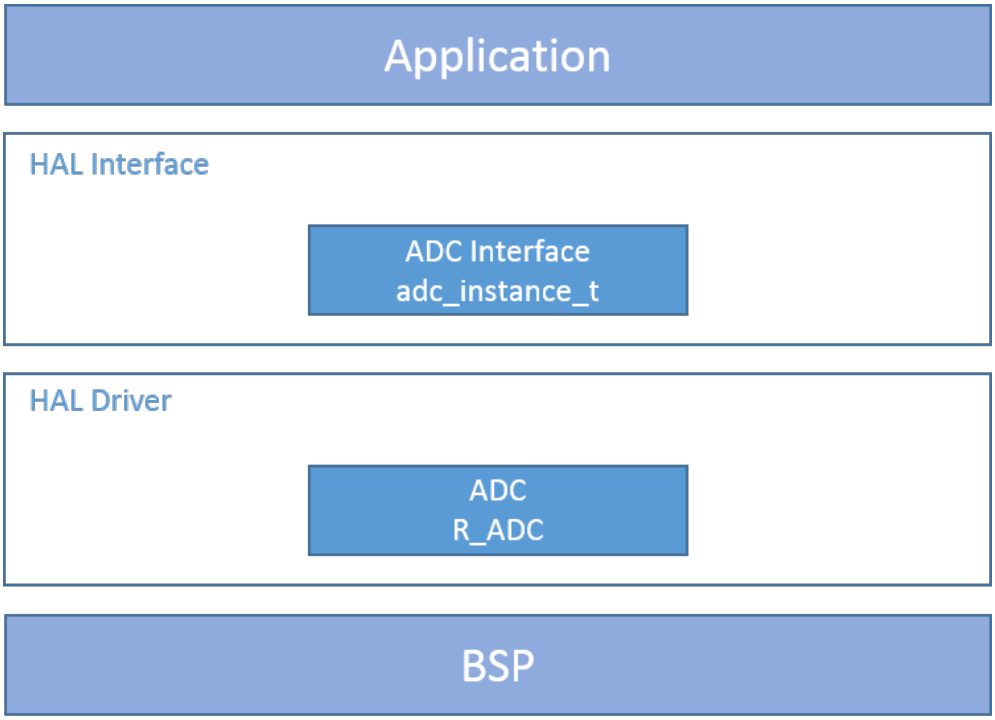


Figure 13.22 Analog to Digital Converter

13.31.2 Estimated Memory Requirements

Table 13.59 Memory Usage for Analog to Digital Converter (ADC) – GCC Compiler

r_adc	Flash (Bytes)
S124 MCU Group	3,709
S3A7 MCU Group	3,361
S5D9 MCU Group	3,353
S7G2 MCU Group	3,353

Table 13.60 Memory Usage for Analog to Digital Converter (ADC) – IAR Compiler

r_adc	Flash (Bytes)
S124 MCU Group	3,353
S3A7 MCU Group	3,392
S5D9 MCU Group	3,392
S7G2 MCU Group	3,392

13.31.3 SSP Supported Hardware Features: ADC

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

Legend:

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

	Support for all Analog Channels (Unit 0 and Unit 1)	8-Bit	10-bit	12-Bit	14-bit	Single-scan Mode
S124	✓	N/A	N/A	✓	✓	✓
S3A7	✓	N/A	N/A	✓	✓	✓
S5D9 or S7G2	✓	✓	✓	✓	N/A	✓

	Continuous Scan Mode	Group Scan Mode	Programmable Gain Amplifier
S124	✓	✓	☒
S3A7	✓	✓	☒
S5D9 or S7G2	✓	✓	☒

13.32 Factory Microcontroller Information (FMI)

13.32.1 Component Introduction

FMI module implements the HAL Driver for FMI peripheral in Synergy MCUs. The driver provides a generic API for reading records from the Factory MCU Information Flash Table. The FMI driver reads the FMIFRT (Factory MCU Information Flash Root Table) on a Synergy microcontroller for the address of the start of the table in flash. It sets the caller's pointer the Product Information record from the table. The FMI driver is used to query the factory flash to read MCU specific features and capabilities programmed in the factory flash. This capability is used to configure all SSP modules that need to be configured with MCU specific information at the time of initialization.

Note: FMI takes up an extra 1K of Flash memory for MCU data used by the FMI module.

The FMI Driver implements the FMI HAL interface in SSP.

13.32.2 Estimated Memory Requirements

Table 13.61 Memory Usage for Factory Microcontroller Information (FMI) – GCC Compiler

r_fmi	Flash (Bytes)
S124 MCU Group	1,685
S3A7 MCU Group	1,605
S5D9 MCU Group	1,589
S7G2 MCU Group	1,589

Table 13.62 Memory Usage for Factory Microcontroller Information (FMI) – IAR Compiler

r_fmi	Flash (Bytes)
S124 MCU Group	1,852
S3A7 MCU Group	1,792
S5D9 MCU Group	1,776
S7G2 MCU Group	1,776

13.33 Low Power Mode Version 2 (LPMV2)

13.33.1 Component Introduction

The LPMV2 driver provides access and configuration of the MCU operating power control modes using the Low Power Mode hardware peripheral. This HAL driver replaces the previous LPM driver; although the original LPM driver is still available for backward compatibility purposes, Renesas encourages the use of the new LPMV2 driver instead.

Note: The Low Power Modes Version 2 Driver will no longer handle operating power control modes of the MCU. The operating power control modes of the MCU are now handled by CGC Driver.

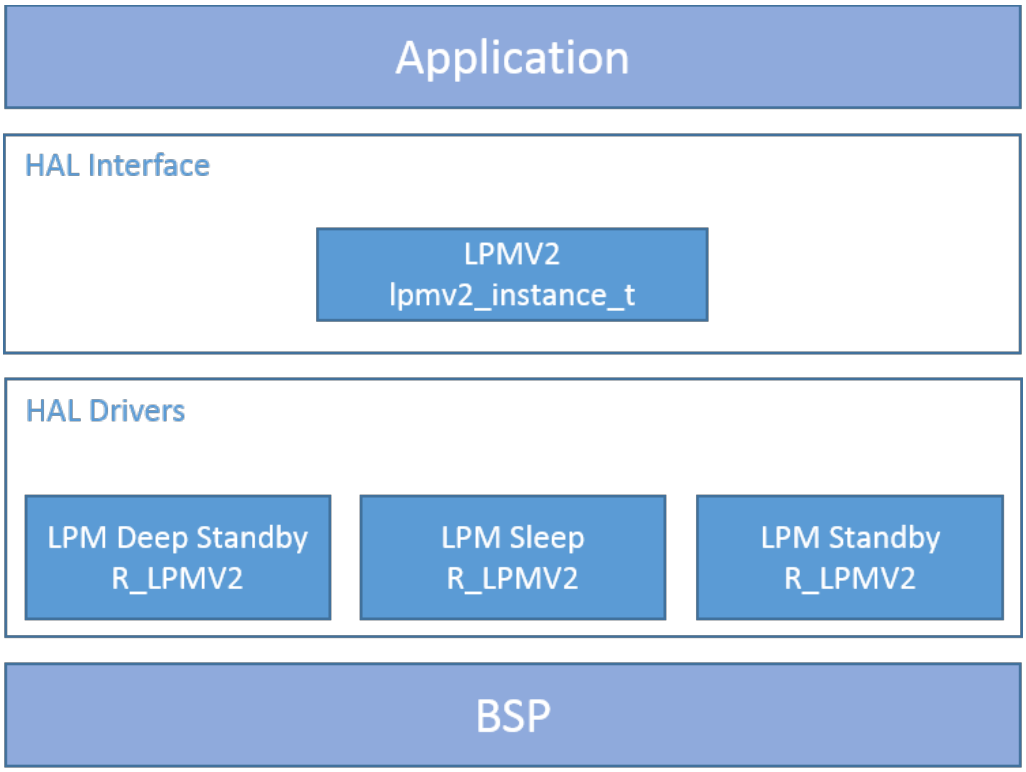
The LPM driver supports the following low power modes:

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

The LPMV2 driver supports reducing power consumption when in deep stand-by mode via internal power supply control and resetting the states of IO ports. The LPM driver supports disabling and enabling of the MCUs other hardware peripherals.

Additional functionality supported: Enable/disable of hardware peripheral for additional power reduction.

The LPMV2 Driver implements the LPMV2 HAL interface in SSP.



13.33.2 Estimated Memory Requirements

Table 13.63 Memory Usage for Low Power Mode – GCC Compiler

r_lpm	Flash (Bytes)
S124 MCU Group	1,121
S3A7 MCU Group	1,121
S5D9 MCU Group	1,645
S7G2 MCU Group	1,645

Table 13.64 Memory Usage for Low Power Mode – IAR Compiler

r_lpm	Flash (Bytes)
S124 MCU Group	832
S3A7 MCU Group	844
S5D9 MCU Group	1,192
S7G2 MCU Group	1,192

13.33.3 SSP Supported Hardware Features: LPMV2

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

Legend:

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

	Module-stop/start bits access	High Speed Operating Power Control Mode	Middle Speed Operating Power Control Mode	Low Speed Operating Power Control Mode	Low Voltage Operating Power Control Mode	Subosc Speed Operating Power Control Mode
S124	✓	✓	✓	✓	☒	✓
S3A7	✓	✓	✓	✓	☒	✓
S5D9 or S7G2	✓	✓	N/A	✓	N/A	✓

	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	✓	✓	☒	✓	☒	✓
S3A7	✓	✓	☒	✓	☒	✓
S5D9 or S7G2	✓	✓	✓	✓	☒	✓

	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	☒	✓	✓	☒
S3A7	✓	✓	✓	☒
S5D9 or S7G2	✓	✓	✓	✓

13.34 Controller Area Network (CAN)

13.34.1 Component Introduction

The CAN driver supports the CAN peripherals available on the Synergy microcontroller hardware. The API provides open, close, read, write, control and information functions. The driver allows for bit timing configuration as defined in the CAN specification and can be configured for 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 can be defined, causing the driver to invoke the callback when transmit, receive or error interrupts are received. The callback provides the argument to indicate the channel, mailbox and the event.

The CAN Driver implements the CAN HAL interface in SSP:

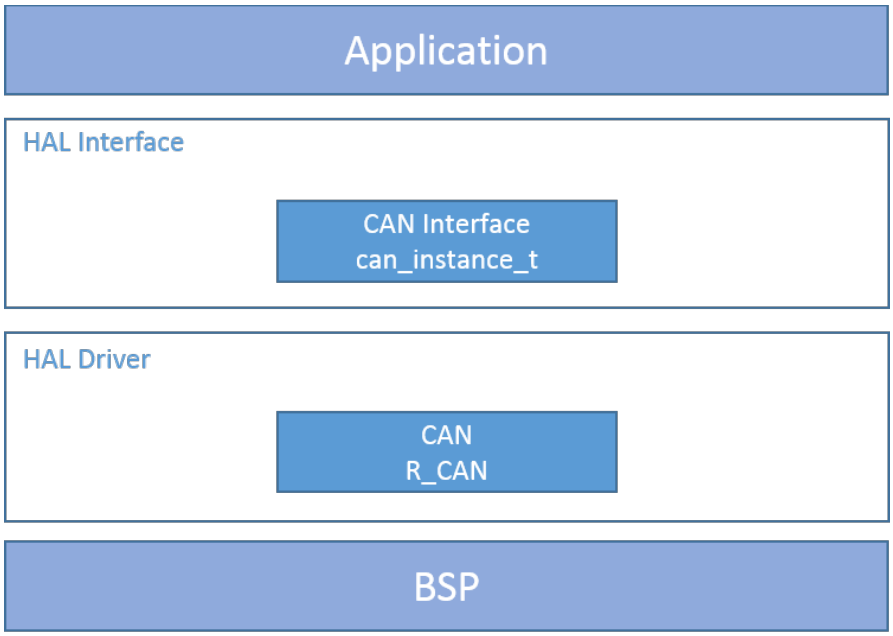


Figure 13.23 Controller Area Network

13.34.2 Estimated Memory Requirements

Table 13.65 Memory Usage for Controller Area Network – GCC Compiler

r_can	Flash (Bytes)
S124 MCU Group	2,357
S3A7 MCU Group	1,985
S5D9 MCU Group	1,981
S7G2 MCU Group	1,981

Table 13.66 Memory Usage for Controller Area Network – IAR Compiler

r_can	Flash (Bytes)
S7G2 MCU Group	2,064
S3A7 MCU Group	2,028
S124 MCU Group	2028
S5D9 MCU Group	2,028

13.34.3 SSP Supported Hardware Features: CAN

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

	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	✓	✓	✓	☒	✓	✓
S3A7	✓	✓	✓	☒	✓	✓
S5D9 or S7G2	✓	✓	✓	☒	✓	✓

	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	☒	✓	✓	✓	✓
S3A7	☒	✓	✓	✓	✓
S5D9 or S7G2	☒	✓	✓	✓	✓

	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	☒	✓	☒	☒	☒	☒
S3A7	☒	✓	☒	☒	☒	☒
S5D9 or S7G2	☒	✓	☒	☒	☒	☒

	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	✓	✓ Except error warning	☒	✓	☒	☒
S3A7	✓	✓ Except error warning	☒ Only 8-bit	✓	☒	☒
S5D9 or S7G2	✓	✓ Except error warning	☒ Only 8-bit	✓	☒	☒

	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 (J1939, 39 bit)
S124	N/A	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓	✓
S5D9 or S7G2	✓	✓	✓	✓	✓	✓

13.35 Serial Sound Interface (SSI)

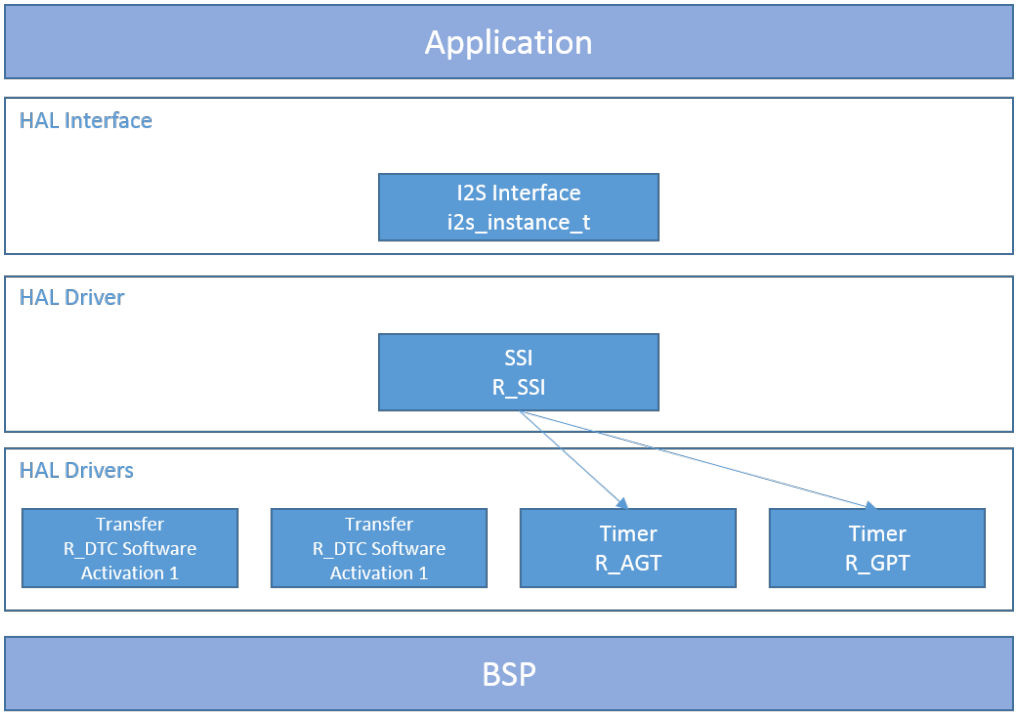
13.35.1 Component Introduction

The SSI driver supports the SSI peripheral in I^2S master mode on the Synergy microcontroller hardware. The driver provides a generic API for serial audio communication using the I^2S serial communication protocol. The driver is typically used to send and receive uncompressed audio in master mode.

The I^2S Driver used with the SSI peripheral in I^2S master mode supports the following features in addition to the standard I^2S protocol:

- Full-duplex I^2S communication (SSI channel 0 only)
- Interrupt driven data transmission and reception
- Integration with the DTC transfer module

The SSI driver implements the I^2S HAL interface in SSP.



13.35.2 Estimated Memory Requirements

Table 13.67 Memory Usage for Serial Sound Interface – GCC Compiler

r_ssi	Flash (Bytes)
S3A7 MCU Group	5,339
S5D9 MCU Group	5,339
S7G2 MCU Group	5,339

Table 13.68 Memory Usage for Serial Sound Interface – IAR Compiler

r_ssi	Flash (Bytes)
S3A7 MCU Group	4,808
S5D9 MCU Group	4,808
S7G2 MCU Group	4,808

13.35.3 SSP Supported Hardware Features: 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

	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	☒	☒	☒	☒	☒	☒
S3A7	☒	☒	☒	✓	✓	☒
S5D9 or S7G2	✓	☒	☒	✓	✓	☒

	Ability to select Stop Word SSIWS	Accepts Interrupts from Communication Errors	Accepts Interrupts from Receive Data Full	Accepts Interrupts from Transmit Data Empty
S124	☒	☒	☒	☒
S3A7	✓	✓	✓	✓
S5D5	☒	✓	✓	✓
S7G2	✓	✓	✓	✓

13.36 Parallel Data Capture Unit (PDC)

13.36.1 Component Introduction

The PDC (Parallel Data Capture Unit) driver can be used for capturing image data from a camera or image sensor module. The PDC driver supports the PDC peripheral available on the Synergy microcontroller hardware.

PDC starts a capture from a connected and configured camera. When a capture is complete and a callback is available (with interrupts enabled), the driver invokes the callback with the argument which provides a pointer to the buffer where the captured image is stored. The event causing the callback is also provided.

The PDC driver implements the PDC interface in SSP:

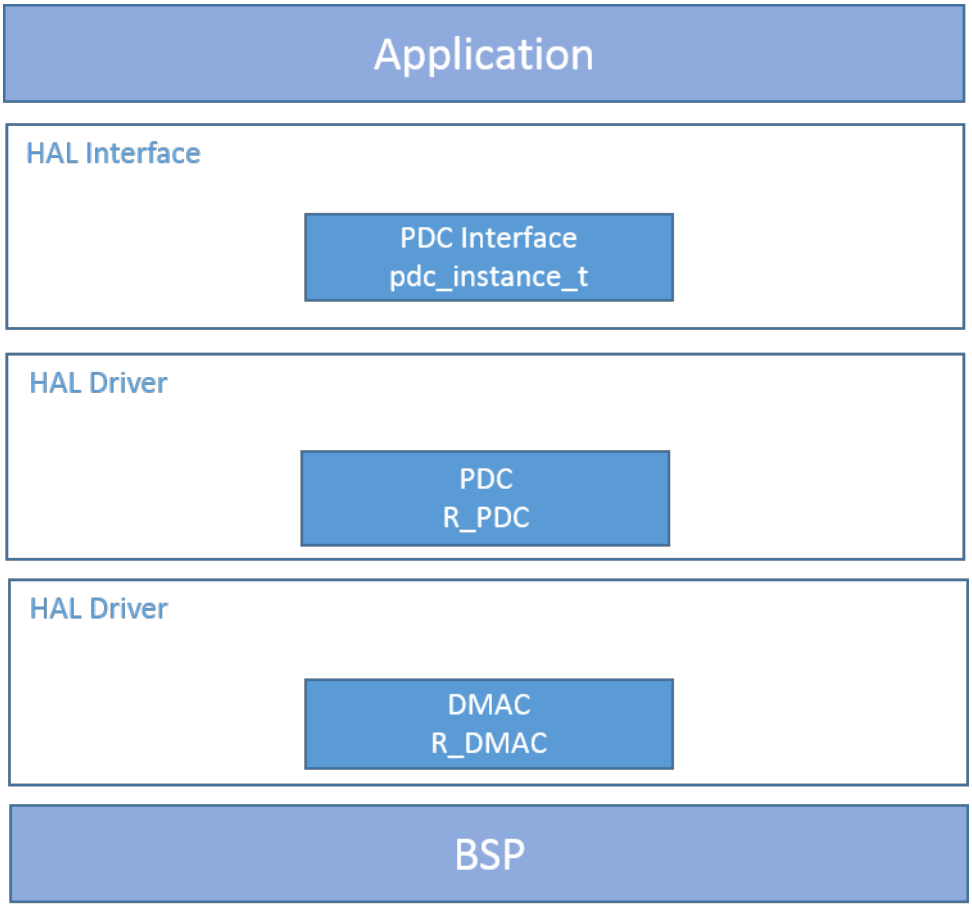


Figure 13.24 Parallel Data Capture Unit

13.36.2 Estimated Memory Requirements

Table 13.69 Memory Usage for Parallel Data Capture – GCC Compiler

r_pdc	Flash (Bytes)
S5D9 MCU Group	2,237
S7G2 MCU Group	2,237

Table 13.70 Memory Usage for Parallel Data Capture – IAR Compiler

r_pdc	Flash (Bytes)
S5D9 MCU Group	2,020
S7G2 MCU Group	2,020

13.36.3 SSP Supported Hardware Features: PDC

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

	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
S3A7	N/A	N/A	N/A	N/A	N/A	N/A
S5D9 or S7G2	✓	✓	✓	✓	✓	✓

	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
S3A7	N/A	N/A	N/A	N/A	N/A	N/A
S5D9 or S7G2	✓	✓	☒	☒	☒	☒

	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
S3A7	N/A	N/A	N/A
S5D9 or S7G2	✓	✓	✓

13.37 Low Voltage Detection (LVD)

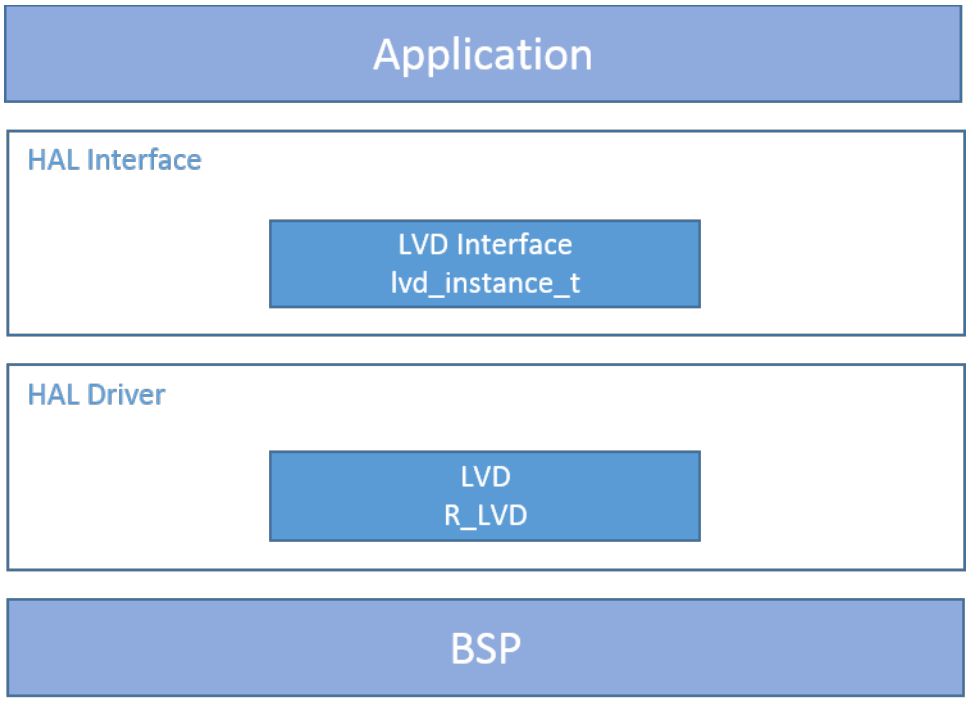
13.37.1 Component Introduction

The LVD (Low Voltage Detection) driver provides access to the configuration of the Low Power Modes hardware peripheral in Synergy Microcontrollers.

The LVD driver supports configuration of the LVD monitors of the Synergy MCUs. The LVD driver provides configuration structures that provide all the information needed to fully configure a single LVD monitor. One instance of the LVD driver is needed per instance of LVD monitor with the exception of the LVD0 monitor. The LVD0 monitor is not configurable at runtime and must be configured at compile time via the OFS1 register.

The LVD1 and LVD2 monitors are both configurable at runtime and are configured by this driver. The open function allows the user to configure and enable an LVD monitor with a single function call. The close function disables the LVD monitor. The statusGet function returns the current status of the LVD monitor. The statusGet function should be used if the driver is in polling mode, without the LVD monitor interrupt enabled. The monitor status consists of two flags, the first flag is a latched flag called crossing_detected, which indicates whether or not the voltage being monitored has crossed the voltage threshold. In polling mode, this flag must be cleared via a call to statusClear. The flag does not need to be cleared explicitly if the LVD interrupt is in use, it will be cleared in the LVD interrupt by the driver code after the user callback function is called. The other flag, current_state, is the instantaneous status of the monitored voltage with respect to the voltage threshold. This flag is not latched and will change as the monitored voltage changes.

The LVD driver implements the Low Voltage Detection Interface in SSP.



13.37.2 Estimated Memory Requirements

Table 13.71 Memory Usage for Low Voltage Detection – GCC Compiler

r_lvd	Flash (Bytes)
S124 MCU Group	2,367
S3A7 MCU Group	2,107
S5D9 MCU Group	2,103
S7G2 MCU Group	2,103

Table 13.72 Memory Usage for Low Voltage Detection – IAR Compiler

r_lvd	Flash (Bytes)
S124 MCU Group	2,184
S3A7 MCU Group	2,108
S5D9 MCU Group	2,108
S7G2 MCU Group	2,108

13.37.3 SSP Supported Hardware Features: 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

	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	✓	✓	✓	✓	✓	✓
S3A7	✓	✓	✓	✓	✓	✓
S5D9 or S7G2	✓	✓	✓	✓	✓	✓

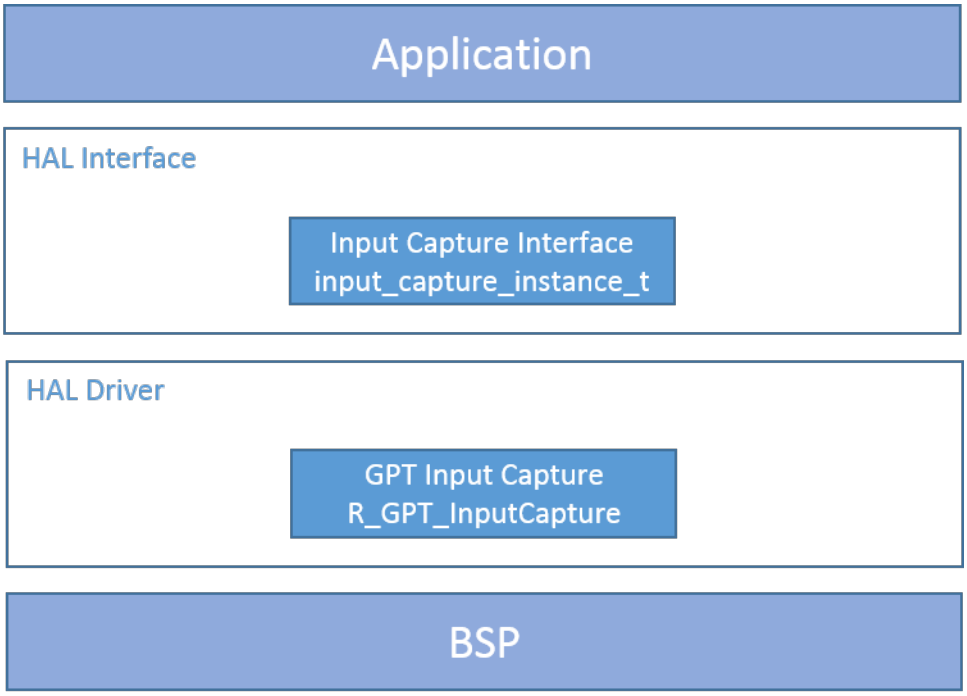
	Digital Filtering with adjustable filter time	Event linking
S124	N/A	☒
S3A7	N/A	☒
S5D9 or S7G2	✓	☒

13.38 General PWM Timer with Input Capture (GPT_INPUT_CAPTURE)

13.38.1 Component Introduction

The GPT Input Capture driver supports the in General PWM Timer (GPT) with Input Capture in Synergy Microcontrollers. The Input Capture Driver configures a timer to measure pulse widths. When a measurement is captured or the counter overflows, a callback is called from a CPU interrupt with the measurement data.

The GPT Input Capture driver implements the Input Capture Driver Interface in SSP.



13.38.2 Estimated Memory Requirements

Table 13.73 Memory Usage for GPT Input Capture – GCC Compiler

r_gpt_input_capture	Flash (Bytes)
S124 MCU Group	1,963
S3A7 MCU Group	1,779
S5D9 MCU Group	1,779
S7G2 MCU Group	1,779

Table 13.74 Memory Usage for GPT Input Capture – IAR Compiler

r_gpt_input_capture	Flash (Bytes)
S124 MCU Group	1,600
S3A7 MCU Group	1,636
S5D9 MCU Group	1,636
S7G2 MCU Group	1,636

13.38.3 SSP Supported Hardware Features: GPT_INPUT_CAPTURE

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

	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	☒	☒	☒	☒	✓	☒
S3A7	☒	☒	☒	☒	✓	☒
S5D9 or S7G2	☒	☒	☒	☒	✓	☒

	PWM Mode	Phase Count Function	Event linking (ELC) function	Noise filtering function
S124	☒	☒	☒	✓
S3A7	☒	☒	☒	✓
S5D9 or S7G2	☒	☒	☒	✓

14. Board Support Package (BSP)

14.1 Component Introduction

The SSP includes BSPs for DK-S7G2, PE-HMI1, DK-S3A7, DK-S124, PK-S5D9, and SK-S7G2 kits. 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 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

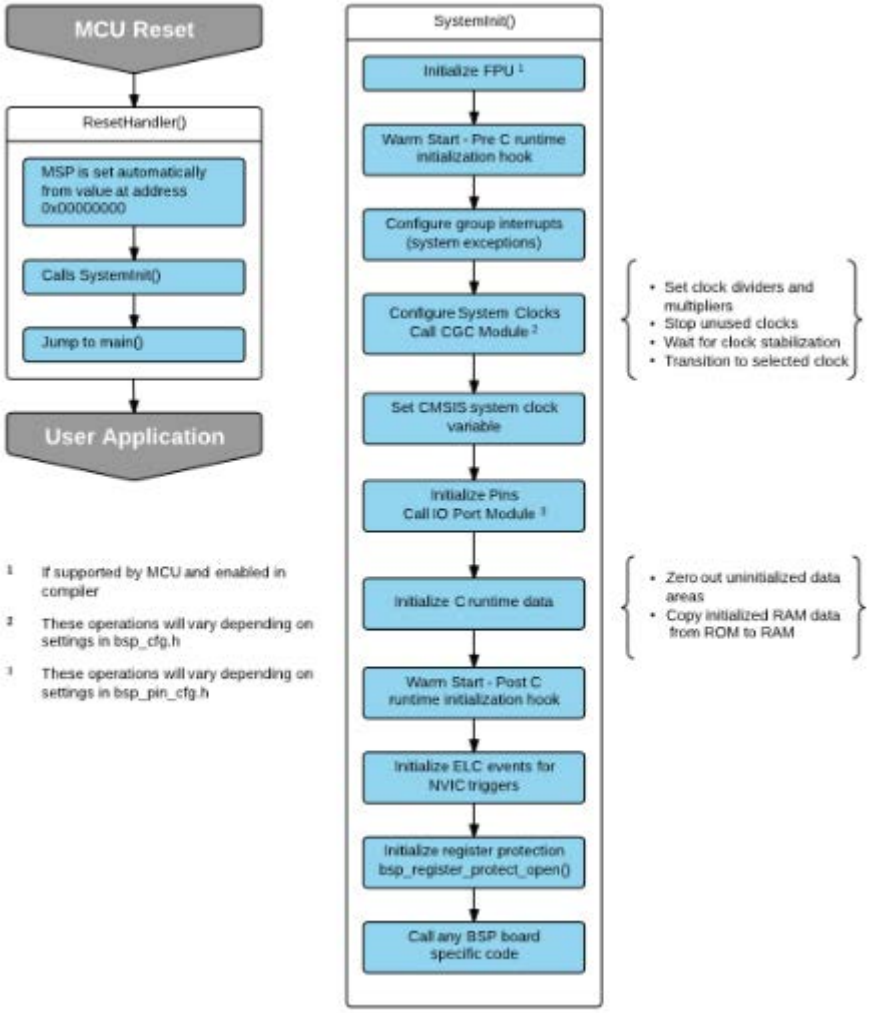


Figure 14.1 Board Support Package Flow of Events

14.1.1 Estimated Memory Requirements

Table 14.1 Memory Usage for Board Support Package – GCC Compiler

BSP	Flash (Bytes)
DK-S7G2	5,577
DK-S3A7	5,093
DK-S124	3,245
PK-S5D9	4,888

Table 14.2 Memory Usage for Board Support Package – IAR Compiler

BSP	Flash (Bytes)
DK-S7G2	5,340
DK-S3A7	4,870
DK-S124	3,110
PK-S5D9	4,779

15. SSP System Performance – Warranted

This section describes the SSP system performance measurements and the environment in which they were measured for the SSP running on specified Synergy hardware, typically a Synergy Development Kit. There are two portions to some of the stated benchmarking and performance test results sections:

- Measured results that are warranted by Renesas under the SSP warranty policy (Section 15, this Section)
- Measured results that are not covered by the SSP warranty policy but are supplemental reference data. (Section 16).

Individual benchmarking and performance characterization depends on the version and configuration of the test environment. The test environments are specified in tables on the next page.

For SSP warranty claims, the claimant must identify which particular test (by particular section number within this performance portion of the datasheet). The claimant must reproduce the suspected errant SSP behavior while operating within the specified environment that is described for the particular test. For example, you must document the compiler, the ISDE version, the version of SSP, your compiler option flags, and so on. If the errant SSP behavior is reproduced within the specified environment, a SSP warranty claim can be made by contacting your local Renesas sales representative or affiliate to process your claim. The claimant must hold a valid SSP Development/Production license to make a SSP warranty claim. Once the claim has been processed by Renesas, the claimant will be notified of the resolution status of the warranty claim within seven days of receipt.

NOTE: Renesas warranties these performance characteristics to within +/- 5% of reported scores or specifications, based on the same hardware configuration (using Renesas kits) specified in this SSP Software Datasheet.

The specified test environment, per individual test, will include but may be not limited to these elements:

- SSP release version upon which the test software was built
- Test software release version
- Compiler version and optimization settings used to build the test software
- Development environment tool version and configuration used to build the test software
- Hardware kit type – typically a Synergy Development Kit with exact kit version number, on which the test software was executed
    - The kit, by default, also indicates the Synergy MCU type with CPU core type and clock speed
- Other configurations that may vary from one test to another depending on the test. Two examples - memory access such as
    - CPU executes instructions from on-chip Flash memory or SRAM, or from external XIP Flash memory or SDRAM
    - Source of data - USB-flash drive for file system, SD card, QSPI Flash memory, etc.

Individual benchmark and performance tests selected for SSP datasheet testing include those that represent performance characterizations of specific MCU features as well as industry-standard benchmark tests. Some of the benchmark test software used in SSP testing is widely available as open source, or it is easily licensed.

Note: Benchmarking can be an inexact science, as there are literally dozens of factors that can affect performance, such as:

- Compiler options
- Compiler version
- Compiler optimization options
- RTOS version

RTOS “as built” configurations

- Libraries included
- MCU clock speed
- Bus clock speed

These performance numbers are therefore best used as guidance and to help select MCU’s for a particular application.

15.1 Test Environments

The following test environments were used for all of the benchmarking and performance characterizations, **unless otherwise noted in each individual measurement section**.

Table 15.1 Test Configuration for DK-S7G2

Test Environment		Part Number and Revision/Version
DK-S7G2 Development Kit		YSDKS7G2S30 kit, v3.1 PCB, ARM® Cortex®-M4 CPU operating frequency 240 MHz
Synergy Software Package		SSP v1.2.0
e² studio ISDE tool		e² studio v5.3.1.002
Compilers	GCC Compiler	4.9.3 20150529 (release)
	GCC Compiler Optimization	-O3, no debug
	IAR C/C++ Compiler	7.71.1
	IAR Compiler Optimization	High speed, no size constraints, no debug

Table 15.2 Test Configuration for PK-S5D9

Test Environment		Part Number and Revision/Version
PK-S5D9 Development Kit		YSPKS5D9E10 V2.0 PCB, ARM® Cortex®-M4 CPU operating frequency 120 MHz
Synergy Software Package		SSP v1.2.0
e² studio ISDE tool		e² studio v5.3.1.002
Compilers	GCC Compiler	4.9.3 20150529 (release)
	GCC Compiler Optimization	-O3, no debug
	IAR C/C++ Compiler	v7.71.1
	IAR Compiler Optimization	High speed, no size constraints, no debug

Table 15.3 Test Configuration for DK-S3A7

Test Environment		Part Number and Revision/Version
DK-S3A7 Development Kit		YSDKS3A7E20 V2.0 PCB, ARM® Cortex®-M4 CPU operating frequency 48 MHz
Synergy Software Package		SSP v1.2.0
e² studio ISDE tool		e² studio v5.3.1.002
Compilers	GCC Compiler	4.9.3 20150529 (release)
	GCC Compiler Optimization	-O3, no debug
	IAR C/C++ Compiler	v7.71.1
	IAR Compiler Optimization	High speed, no size constraints, no debug

Table 15.4 Test Configuration for DK-S124

Test Environment		Part Number and Revision/Version
DK-S124 Development Kit		YSDKS124S20 V3.0 PCB, ARM® Cortex®-M0+ CPU operating frequency 32 MHz
Synergy Software Package		SSP v1.2.0
e² studio ISDE tool		e² studio v5.3.1.002
Compilers	GCC Compiler	4.9.3 20150529 (release)
	GCC Compiler Optimization	-O3, no debug
	IAR C/C++ Compiler	v7.71.1
	IAR Compiler Optimization	High speed, no size constraints, no debug

15.2 MCU Measurements

15.2.1 EEMBC CoreMark®

A general purpose, run-time test of CPU core and memory sub-system performance that reflects typical embedded MCU application requirements, EEMBC CoreMark has replaced Dhrystone MIPS and is widely used in the embedded

MCU industry. Although EEMBC has a dozen benchmark suites (Autobench, Digital Entertainment/DENbench, Netbench, and so on), the benchmark most suited for embedded MCU's is CoreMark.

"Although it doesn't reflect how you would use a processor in a real application, sometimes it's important to isolate the CPU's core from the other elements of the processor and focus on one key element. For example, you might want to have the ability to ignore memory and I/O effects and focus primarily on the pipeline operation. This is CoreMark's domain. CoreMark is capable of testing a processor's basic pipeline structure, as well as the ability to test basic read/write operations, integer operations, and control operations." – EEMBC website

Table 15.5 Warranted Test Results for MCU-level Performance – Larger is better

Kits	Toolchain	Clock MHz	Iterations	Time(sec)	Iterations/Second
DK-S7G2	GCC	240	10,000	21	476
	IAR	240	10,000	14	714
PK-S5D9	GCC	120	10,000	38	263
	IAR	120	10,000	24	417
DK-S3A7	GCC	48	10,000	95	105
	IAR	48	10,000	63	159
DK-S124	GCC	32	10,000	199	50
	IAR	32	10,000	125	80

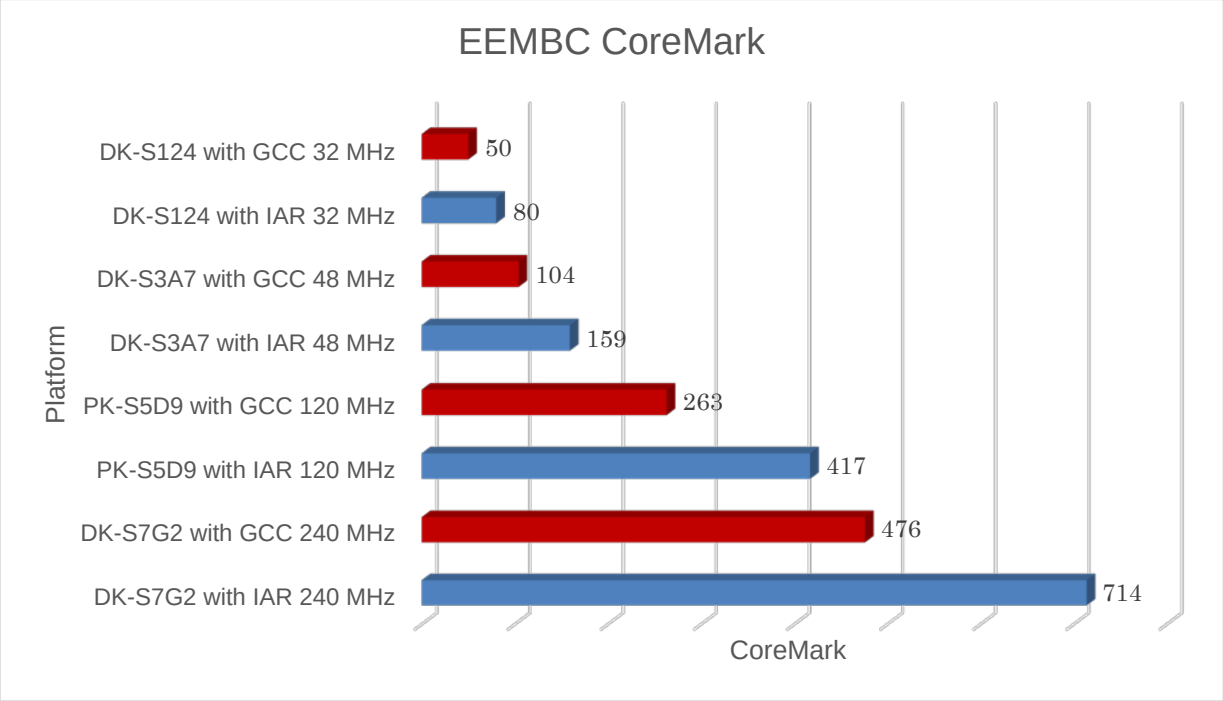


Figure 15.1 EEMBC CoreMark Across the Synergy Product Line

As the data shows in Figure 15.1, the Synergy Platform is a scalable platform with a variety of MCU CPU performance characteristics.

15.3 Operating System Performance Metrics.

15.3.1 Thread-Metric Benchmarks

Express Logic's Thread-Metric is a free-source benchmark suite for measuring RTOS performance, in particular the speed with which an RTOS completes services for an application. This test is applicable to real-word embedded MCU system applications using a Synergy family MCU with elements from Synergy Platform. Thread-Metric has a number of kernels that are used for testing, and has been ported to other RTOS systems such as FreeRTOS and others.

Thread Metric is available at no charge from Express Logic: http://rtos.com/DownloadCenter/Thread-MetricForm.php

The scores for each kernel is a composite value which indicate performance of the system with higher score the indicating better performance. The Thread-Metric test suite consists of 8 distinct RTOS tests that are designed to highlight commonly used aspects of an RTOS. The test measures the total number of RTOS events that can be

processed during a specific timer interval. A 30 second time interval is recommended by Express Logic and was used in our testing.

Basic Processing Test: This is the baseline test consisting of a single thread. This should execute the same on every operating system. Test values from testing with different RTOS products should be scaled relative to the difference between the values of this test.

Cooperative Scheduling Test: This test consists of 5 threads created at the same priority that voluntarily releases control to each other in a round-robin fashion. Each thread will increment its run counter and then relinquish to the next thread. At the end of the test the counters will be verified to make sure they are valid (should all be within 1 of the same value). If valid, the numbers will be summed and presented as the result of the cooperative scheduling test.

Preemptive Scheduling Performance: The Preemptive context switching test is much more complex, and reflects a real-time computing system that has to deal with different thread priorities. It measures the time the RTOS takes to change the execution context from one thread to a higher-priority thread, which preempts the executing thread. The steps are:

1. Five threads are created that each have a unique priority.
2. The threads run until preempted by a higher-priority thread.
3. All threads except the lowest-priority thread are in a suspended state.
4. The lowest-priority thread resumes the next highest-priority thread, and so on until the highest-priority thread executes.
5. Each thread increments its run count and then suspends.
6. Once processing returns to the lowest-priority thread, it increments its run counter and again resumes the next highest-priority thread, starting the process over again.

Interrupt Processing Test: This test consists of a single thread. The thread will cause an interrupt (typically implemented as a trap), which will result in a call to the interrupt handler. The interrupt handler will increment a counter and then post to a semaphore. After the interrupt handler completes, processing returns to the test thread that initiated the interrupt. The thread then retrieves the semaphore set by the interrupt handler, increments a counter and then generates another interrupt.

Interrupt Preemption Processing Test: This test is similar to the previous interrupt test. The big difference is the interrupt handler in this test resumes a higher priority thread, which causes thread preemption.

Message Processing: This test consists of a thread sending a 16 byte message to a queue and retrieving the same 16 byte message from the queue. After the send/receive sequence is complete, the thread increments its run counter.

Synchronization Processing Test: This test consists of a thread getting a semaphore and then immediately releasing it. After the get/put cycle completes, the thread will increment its run counter.

Memory Kernel Performance: The Thread-Metric memory kernel test is a memory allocation and deallocation test, and measures the time it takes the RTOS to allocate a fixed-size block of memory for a thread. This test is designed to test memory allocation of 128-byte blocks.

Table 15.6 Warranted Results for Operating System Performance using Thread Metrics

Unit	Iterations During a 30-second period. Higher value means better performance		
Test Name	**Kit**	**GCC**	**IAR**
Basic processing	DK-S7G2 240 MHz	439,241	638,711
Cooperative scheduling		35,430,188	32,665,109
Interrupt preemption processing		12,202,417	12,787,462
Interrupt processing		41,376,976	44,441,455
Memory allocation		47,997,134	50,346,392
Message processing		28,105,039	29,001,135
Preemptive scheduling		15,671,200	16,185,615
Synchronization processing		56,689,420	61,534,341
Basic processing	PK-S5D9 120 MHz	270,278	439,036
Cooperative scheduling		22,581,316	20,927,542
Interrupt preemption processing		7,910,623	7,945,972
Interrupt processing		26,467,357	27,065,063
Memory allocation		29,996,266	30,505,475
Message processing		17,389,148	17,901,297
Preemptive scheduling		10,260,765	10,196,950
Synchronization processing		33,329,149	34,611,908
Basic processing	DK-S3A7 48 MHz	108,102	175,589
Cooperative scheduling		8,349,957	8,033,222
Interrupt preemption processing		2,867,916	2,986,503
Interrupt processing		10,906,497	10,743,350
Memory allocation		11,800,437	12,410,376
Message processing		6,665,173	6,982,836
Preemptive scheduling		3,788,266	3,852,964
Synchronization processing		13,087,937	13,842,244
Basic processing	DK-S124 24 MHz	46,831	63,836
Cooperative scheduling		3,378,762	3,246,739
Interrupt preemption processing		1,282,557	1,259,889
Interrupt processing		4,642,996	4,442,138
Memory allocation		5,493,617	5,177,156
Message processing		2,834,254	2,787,441
Preemptive scheduling		1,676,195	1,634,857
Synchronization processing		6,150,971	5,898,545

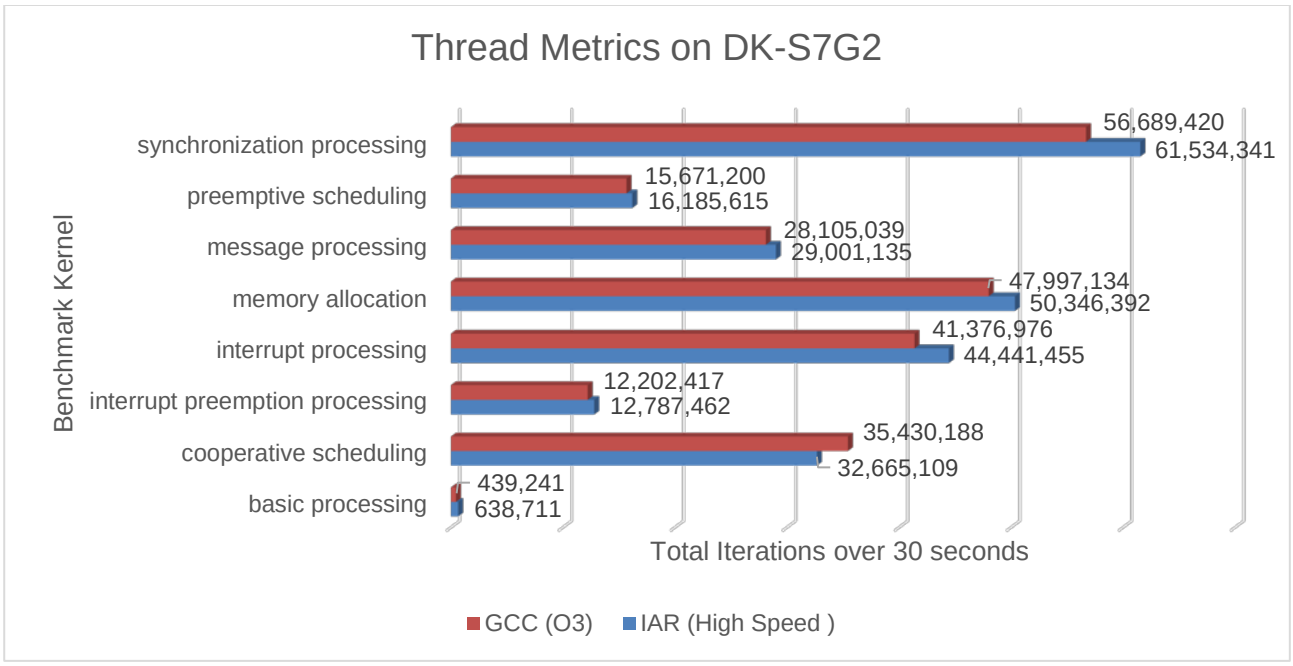


Figure 15.2 Thread Metric on DK-S7G2 kit

.

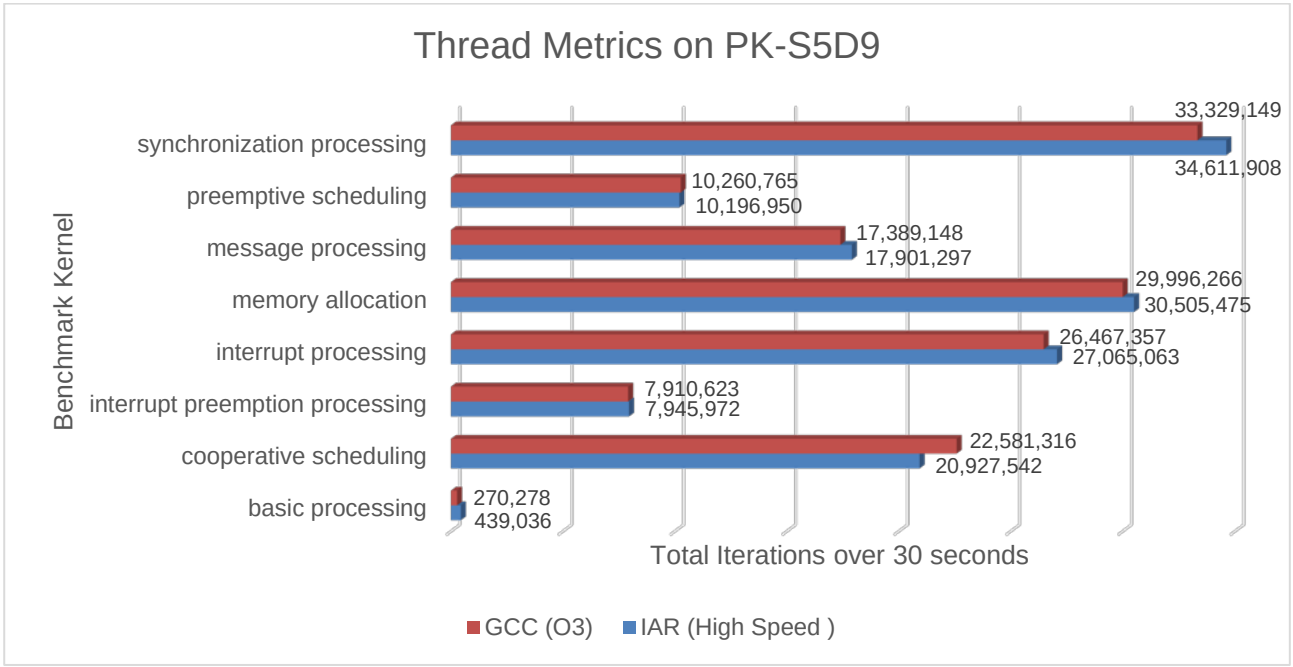


Figure 15.3 Thread Metric on PK-S5D9 kit

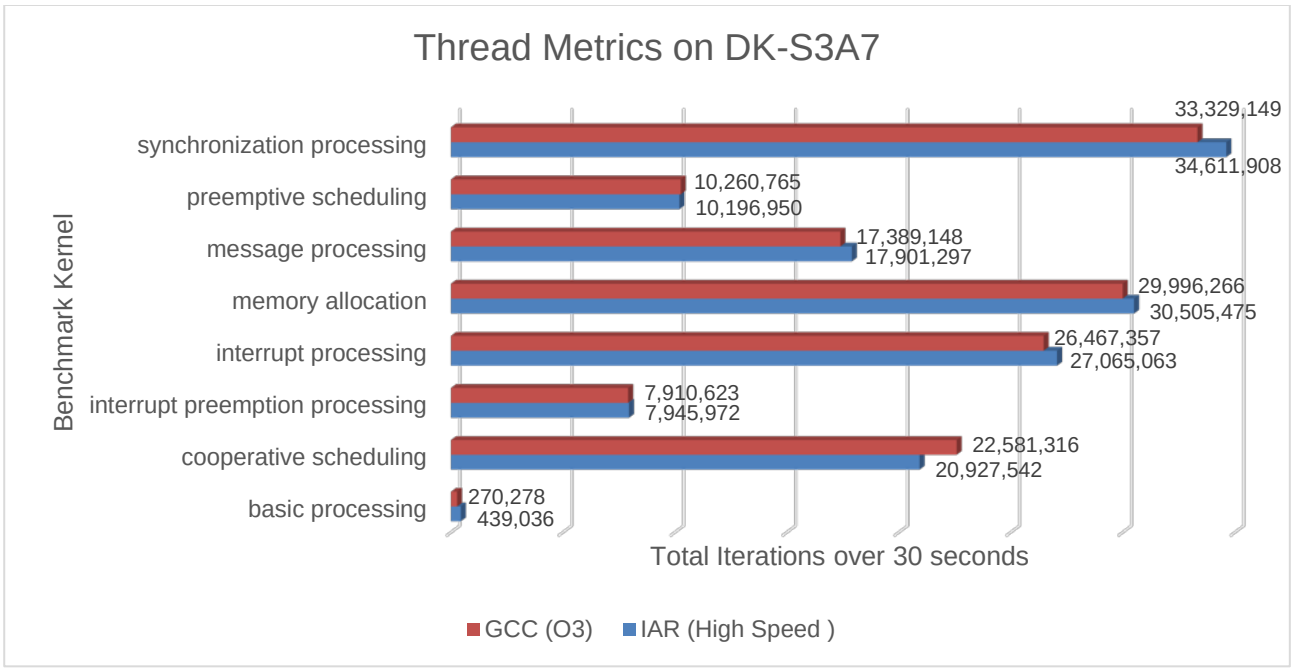


Figure 15.4 Thread Metric on DK-S3A7 kit

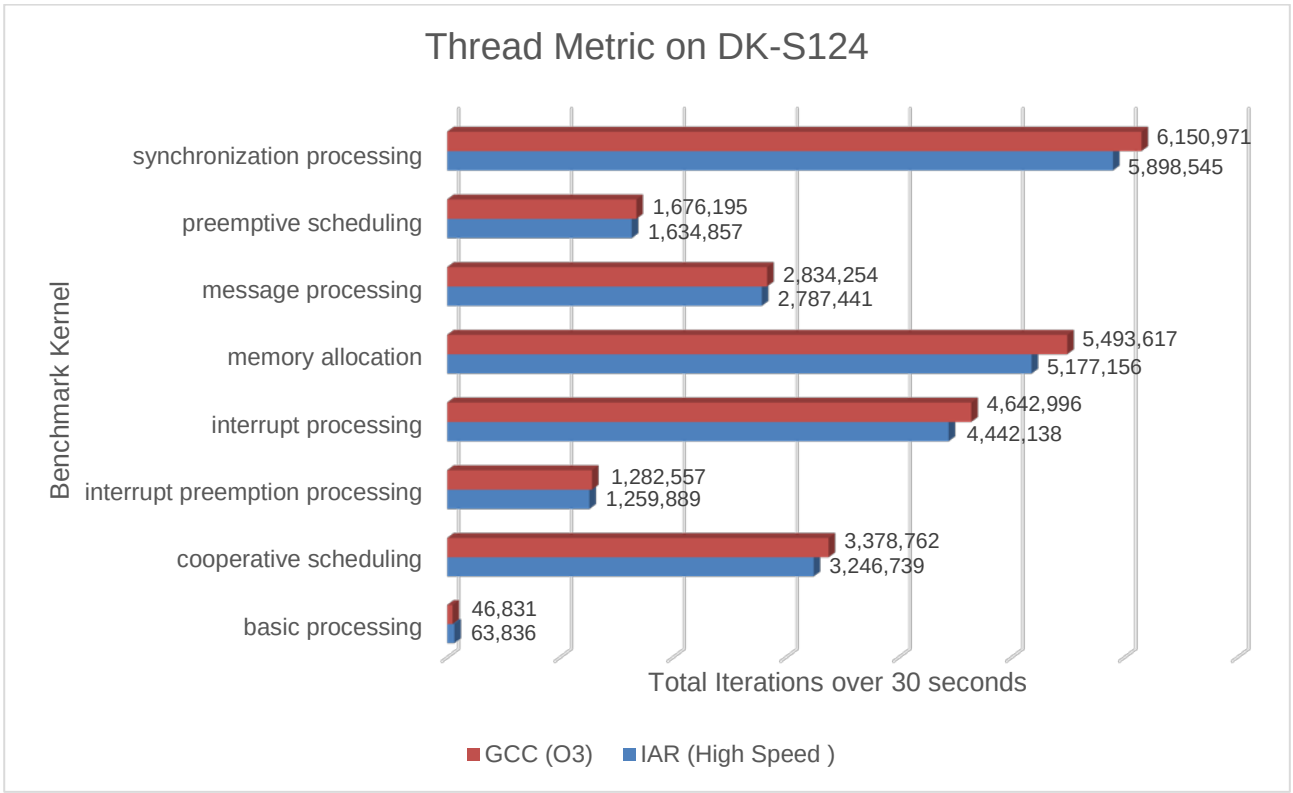


Figure 15.5 Thread Metric on DK-S124 Kit

Note that the MCU device on the DK-S124 kit is from the S124 MCU Group operating at 24 MHz using an ARM Cortex M0+ processor core, different from the ARM Cortex M4-based CPU cores in the previous kits.

15.4 Networking Performance Metrics

15.4.1 Metrics Covered by SSP Warranty

(1) iPerf Benchmark

The industry standard iPerf benchmark suite, an open-source benchmark suite that runs on a Linux or Windows host was used for network performance testing of the SSP's NetX TCP/IP networking stack. Specific focus is on TCP and UDP performance and measured network throughput as the key performance indicator. iPerf and instructions can be found here: http://sourceforge.net/projects/iperf/asdf. For this test, iPerf 2.0.2.7 was used.

Table 15.7 Warranted Results for Networking Performance – iPerf

	DK-S7G2 Throughput in Mbps		
	Test Case	GCC	IAR
NetX™	TCP Transmit Test	93	94
	TCP Receive Test	93	93
	UDP Transmit Test	94	94
	UDP Receive Test	93	92
NetX Duo™ IPv4	TCP Transmit Test	94	94
	TCP Receive Test	94	94
	UDP Transmit Test	94	94
	UDP Receive Test	92	90
NetX Duo™ IPv6	TCP Transmit Test	93	93
	TCP Receive Test	92	92
	UDP Transmit Test	92	92
	UDP Receive Test	84	86

15.5 File System Performance Metrics

15.5.1 Performance characterization on DK-S3A7 and DK-S7G2 kits

(1) PostMark Benchmark Characterization

PostMark is the benchmark used in the NetApp Technical Report TR-3022, *"PostMark: A New File System Benchmark"*. The paper fully explains how to use this tool. PostMark measures performance by simulating performance in the small-file regime used by Internet software, especially Electronic mail, Netnews, and web-based commerce doing file Create, Read, Write, Append, and Delete. At its essence, PostMark is a benchmark designed to simulate the behavior of mail servers, but in the embedded space can be generalized to read/write performance for 1 MB files of any type.

PostMark consists of three phases. In the first phase a pool of files are created. In the next phase four types of transactions are executed: files are created, deleted, read, and appended to. In the last phase, all files in the pool are deleted. Renesas selected a configuration that specified a maximum of 200 files and 500 transactions. Each transaction consists of either a read or append operation, and a create or delete operation, so the workload consisted of creating 200 files, executing a sequence of transactions, and deleting the files that remained after the transaction sequence. The exact sequence of operations is specified by a pseudo-random number generator.

The metric for this test is the number of seconds required to execute this workload. The split-out of the file Create, Read, Append, and then Delete functions, PostMark was taken “out of the box” and combined into a single workload.

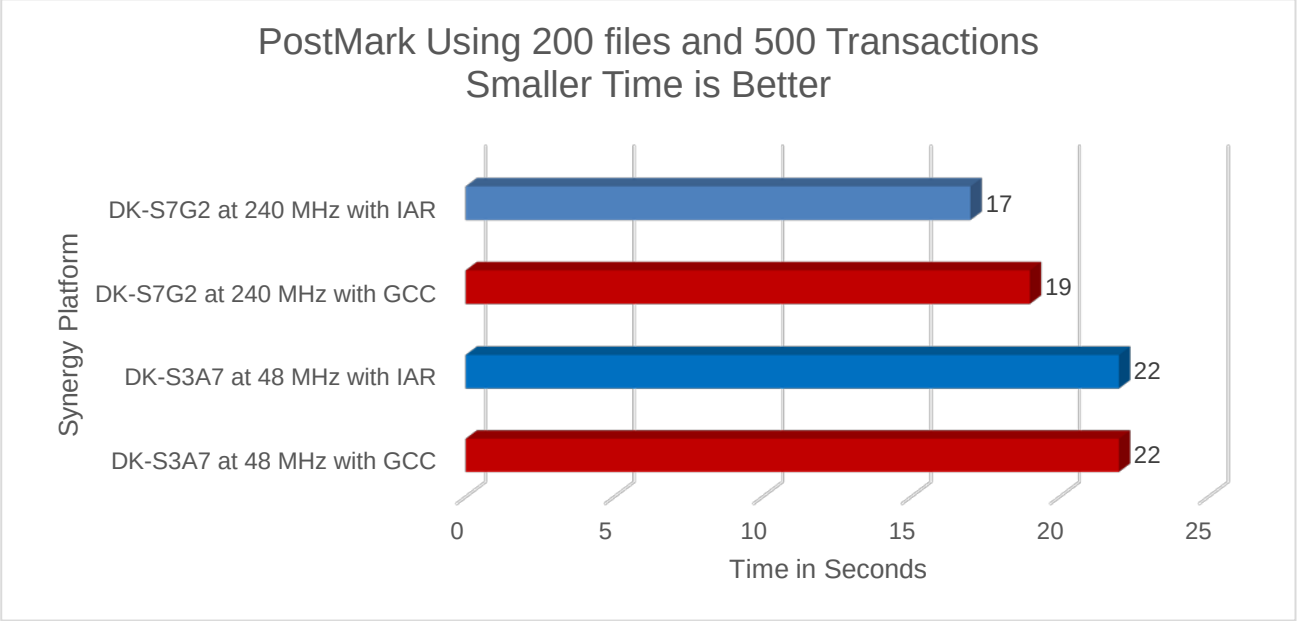


Figure 15.6 PostMark Benchmark Characterization

16. SSP System Performance – Supplemental Reference Data (not warranted)

Supplemental Reference Data are measured results that are not covered by the SSP warranty policy but are useful indicators of performance to help you plan your system design.

Unless otherwise noted, the Configurations used in Section 15 are the same for Section 16.

16.1 USB

16.1.1 USBX Low Level Measurements

Using a commonly-available USB device (Kingston DTSE9G2 64GB USB3.0), read and write times were measured through USBX compiled under IAR and GCC, for both burst mode and sustained mode, to and from the USB flash drive.

Configurations:

CPU Clock: 120 Mhz
GCC Ver: 4.9.3 20150529 (release) [ARM/embedded-4_9-branch revision 227977]
Optimize:-O2
ROM: 2 MB, SDRAM: 640 KB, FLASH: 64KB
PK-S5D9: V1.1

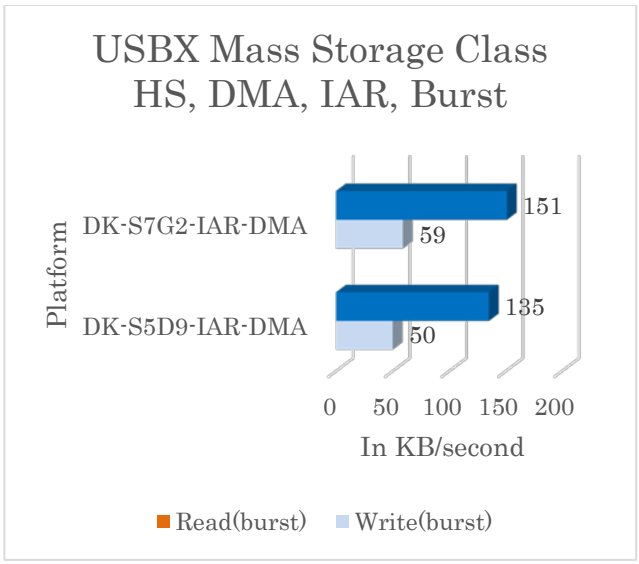
CPU Clock: 120 Mhz
IAR Version: 7071001
Optimize:-OHB
ROM: 2 MB, SDRAM: 640 KB, FLASH: 64KB
PK-S5D9: V1.1

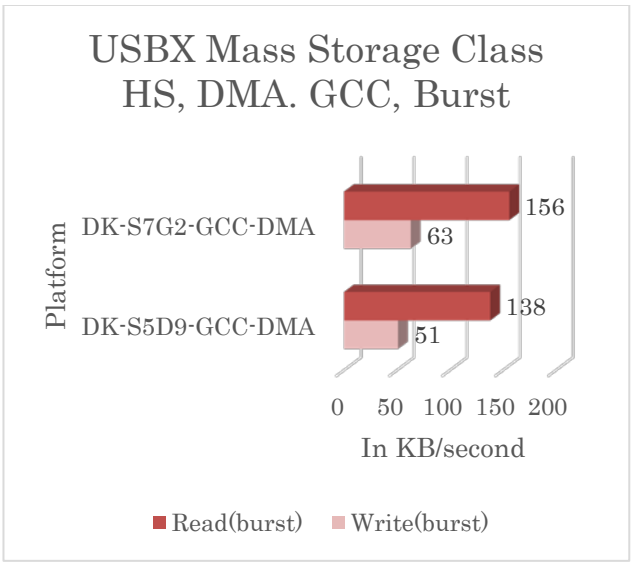
CPU Clock: 240 Mhz
GCC Ver: 4.9.3 20150529 (release) [ARM/embedded-4_9-branch revision 227977]
Optimize:-O2
ROM: 4 MB, SDRAM: 640 KB, FLASH: 64KB
DK-S7G2: V3.1

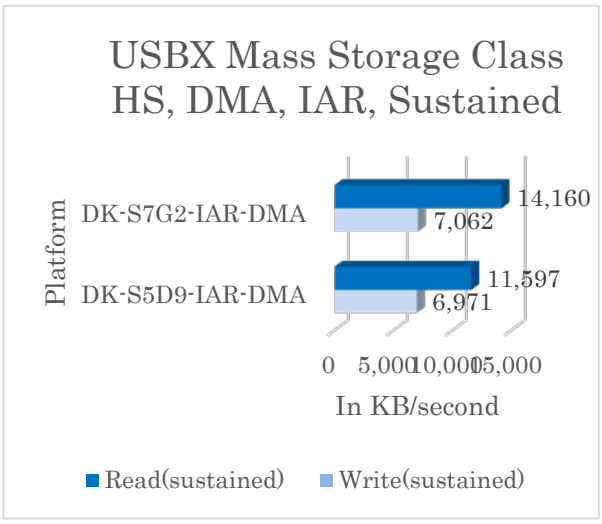
CPU Clock: 240 Mhz
IAR Version: 7071001
Optimize:-OHB
ROM: 4 MB, SDRAM: 640 KB, FLASH: 64KB
DK-S7G2: V3.1

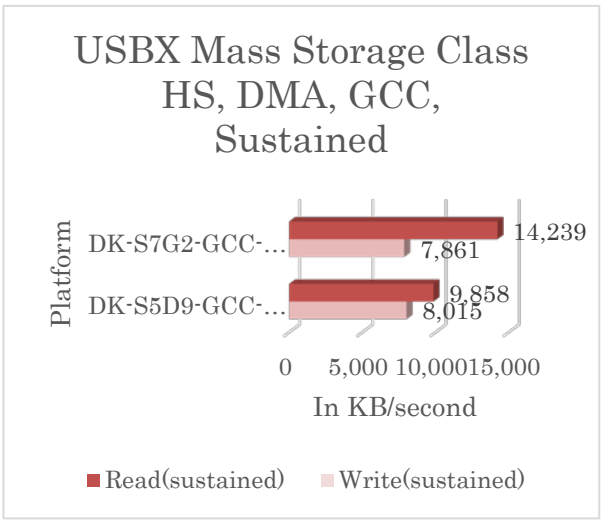
	IAR			
	DK-S5D9		DK-S7G2	
	OHB_CPU	OHB_DMA	OHB_CPU	OHB_DMA
Write(burst)	35 KB/s	50 KB/s	24 KB/s	59 KB/s
Write(sustained)	6,296 KB/s	6,971 KB/s	4,746 KB/s	7,062 KB/s
Read(burst)	135 KB/s	135 KB/s	151 KB/s	151 KB/s
Read(sustained)	9,820 KB/s	11,597 KB/s	9,820 KB/s	14,160 KB/s

	GCC			
	DK-S5D9		DK-S7G2	
	O2_CPU	O2_DMA	O2_CPU	O2_DMA
Write(burst)	51 KB/s	52 KB/s	61 KB/s	63 KB/s
Write(sustained)	6,574 KB/s	6,410 KB/s	8,015 KB/s	7,861 KB/s
Read(burst)	138 KB/s	138 KB/s	156 KB/s	156 KB/s
Read(sustained)	9,858 KB/s	14,239 KB/s	9,858 KB/s	14,239 KB/s









(1) HS means High Speed. DMA means using Direct Memory Access.

(2) Notes: Burst speed mode is slower than sustained speed in these particular tests because the buffer of burst test is just 512 Bytes, but the buffer of sustained mode is up to 256 Kbytes. The burst mode file size is small, compared with the sustained large file test, so there is more overhead on the transactions.

For the burst mode test, the effective load is lower, so the efficiency is also lower. This is similar to using a PC to copy a folder which has many files: it is slower than if you copy a single zipped package.

16.1.2 USBX Small and Large File Operations

The following data was measured on the same platforms as those used for the USBX low level measurements. For the time measurements, smaller is better.

	DK-S5D9-GCC-DMA	DK-S7G2-GCC-DMA	DK-S5D9-IAR-DMA	DK-S7G2-IAR-DMA
3MB File Create	3477 us	2,170 us	4046 us	2661 us
3MB File Write	233 KB/s	333 KB/s	332 KB/s	329 KB/s
3MB File Read	318 KB/s	321 KB/s	321 KB/s	322 KB/s
500MB File Write	6,400 KB/s	12,800 KB/s	4,266 KB/s	12800 KB/s
500MB File Read	12,800 KB/s	12,800 KB/s	12,800 KB/s	12,800 KB/s

16.2 SDMMC using FILEX

This measurement shows SDMMC throughput for reads and writes. The file storage media for this measurement is an SD Card. Then file read and write speed is limited by the SD card transmitting speed.

SD card speed is customarily rated by its sequential read or write speed. The newer families of SD card improve card speed by increasing the bus rate (the frequency of the clock signal that strobes information into and out of the card).

The SD Association defines standard speed classes for SDHC/SDXC cards indicating minimum performance (minimum serial data writing speed) to record video. Both read and write speeds must exceed the specified value. The specification defines these classes in terms of performance curves that translate into the minimum read-write performance levels on an empty card and suitability for different applications. Please refer to https://en.wikipedia.org/wiki/Secure_Digital for more information.

Different SD cards will exhibit different performance characteristics. The SD card using in this document is "SanDisk Ultra 40MB/s Class10 speed".

The following was measured using FILEX:

CPU Clock: 48 Mhz

GCC Ver: 4.9.3 20150529 (release) [ARM/embedded-4_9-branch revision 227977]

Optimize:-O3

DK-S3A7: V2

CPU Clock: 48 Mhz

IAR Version: 7071001

Optimize:-OHSpeed, No size constraints

DK-S3A7: V2.0

CPU Clock: 240 Mhz

GCC Ver: 4.9.3 20150529 (release) [ARM/embedded-4_9-branch revision 227977]

Optimize:-O3

DK-S7G2: V3.1

CPU Clock: 240 Mhz

IAR Version: 7071001

Optimize:-OH Speed, No size constraints

DK-S7G2: V3.1

	DK-S7G2 GCC	DK-S3A7 GCC	DK-S3A7 IAR	DK-S7G2 IAR
Time to Write 1,048,576 Bytes	316 Byte/ms	291 Byte/ms	296 Byte/ms	317 Byte/ms
Time to Read 1,048,576 Bytes	1,852 Byte/ms	1,129 Byte/ms	1,302 Byte/ms	1974 Byte/ms

16.3 SCI_I2C HAL driver

This use case acts as application level, calls SF_SCI_I2C framework and SCI_I2C HAL driver's API to finish test program. The following was measured at the HAL device driver level on a **DK-S3A7** as an I2C master. The EEPROM used is the CAT24C64 on DK-S3A7 board, as the I2C slave.

Configuration:

CPU Clock: 48 Mhz

GCC Ver: 4.9.3 20150529 (release) [ARM/embedded-4_9-branch revision 227977]

Optimize:-O3

ROM: 1 MB, SDRAM: 192 KB, FLASH: 16KB

DK-S3A7: V2.0

CPU Clock: 48 Mhz
IAR Version: 7071001
Optimize:-OHSpeed, No size constraints
ROM: 1 MB, SDRAM: 192 KB, FLASH: 16KB
DK-S3A7: V2.0

	DK-S3A7 GCC @ 48 MHz	DK-S3A7 IAR @48 MHz
Write Time, 1 MBits to EEPROM	3426 ms	3491 ms
Write Initiate Time, 1 MBits to EEPROM	1437 ns	1708 ns
Write Speed, 1 MBits to EEPROM	37 KB/s	36 KB/s
Read Time, 4 MBits from EEPROM	3484 ms	3530 ms
Read Initiate Time, 4 MBits from EEPROM	1520 ns	1687 ns
Read Speed, 4 MBits from EEPROM	36 KB/s	36 KB/s
Write Time, 32 bits to EEPROM	122 us	125 us
Write Initiate Time, 32 bits to EEPROM	1562 ns	2041 ns
Read Time, 32 bits from EEPROM	151 us	153 us
Read Initiate Time, 32 bits from EEPROM	1479 ns	1687 ns
Callback Test Initiate Time	541 ns	583 ns

16.4 SF_SCI_I2C Framework

This test measures the performance of the SCI_I2C software framework on a DK-S3A7. The SCI I2C transfer rate is up to 400Kbps (fast mode). The SCI_I2C Channel 4 is selected as the I2C master, The EEPROM (CAT24C64) on DK-S3A7 board is the I2C slave device.

Configuration:

CPU Clock: 48 Mhz
GCC Ver: 4.9.3 20150529 (release) [ARM/embedded-4_9-branch revision 227977]
Optimize:-O3
ROM: 1 MB, SDRAM: 192 KB, FLASH: 16KB
DK-S3A7: V2.0

CPU Clock: 48 Mhz
IAR Version: 7071001
Optimize:-OHSpeed, No size constraints

DK-S3A7: V2.0

	DK-S3A7 GCC @ 48 MHz	DK-S3A7 IAR @48 MHz
Write Time, 1 MBits to EEPROM	3448 ms	3499 ms
Write Initiate Time, 1 MBits to EEPROM	4770 ns	4937 ns
Write Speed, 1 MBits to EEPROM	37 KB/s	36 KB/s
Read Time, 4 MBits from EEPROM	3487 ms	3536 ms
Read Initiate Time, 4 MBits from EEPROM	4937 ns	4833 ns
Read Speed, 4 MBits from EEPROM	36 KB/s	36 KB/s
Write Time, 32 bits to EEPROM	142 us	144 us
Write Initiate Time, 32 bits to EEPROM	4916 ns	5354 ns
Read Time, 32 bits from EEPROM	170 us	172 us
Read Initiate Time, 32 bits from EEPROM	4958 ns	4833 ns
Callback Test Initiate Time	291 ns	437 ns

16.5 Cryptography

In the following data on SSP Cryptography functions, a variety of algorithms are shown. The following results for **IAR** on the **DK-S7G2**. All results are in "peripheral clock cycles" (the DK-S7G2 peripheral bus is clocked at 120 MHz).

Algorithm	Key Length	Mode	Data (Bytes)	Encode	Encode Time in ms	Decode	Decode Time in ms
AES	128	CBC	1,024	4,135	0.0345	4,159	0.034658
AES	128	CBC	2,048	7,783	0.0649	7,807	0.065058
TDES	192	CBC	512	2,875	0.0240	2,877	0.023975
SHA2	256		1,024	2,429	0.0202		
SHA1	160		1,024	2,681	0.0223		
TRNG			1,024	17,947	0.1496		

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Jan, 2016	-	Editorial updates. Performance data added.
1.01	Feb, 2016	-	Changes for SSP release 1.1.0-alpha. All performance data changed to TBD. New modules added.
1.10	May, 2016	-	Changes for SSP release 1.1.0
1.11	Jun, 2016	-	Added performance data for SSP modules
1.20	Dec, 2016	-	Updated for SSP v1.2.0-P1
1.30	Mar, 2017	-	Updated for SSP v1.2.0 broad release
1.38	May, 2017	-	Added Supplemental Reference Data (Section 16)

Synergy Software Package (SSP) v1.2.0 Datasheet

Publication Date: Rev.1.38 May 5, 2017

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.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

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: +44-1628-585-100, Fax: +44-1628-585-900

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, P. R. China 200333
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 II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 4.1

Renesas Synergy™ Platform
Synergy Software Package (SSP) v1.2.0

Renesas Electronics Corporation

R01DS0272EU0138