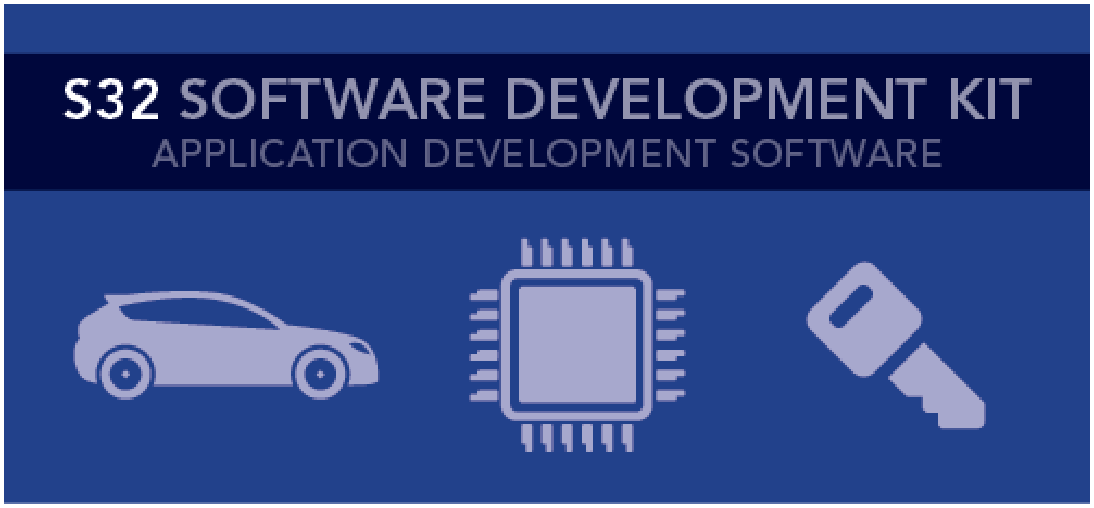
S32 SOFTWARE DEVELOPMENT KIT
APPLICATION DEVELOPMENT SOFTWARE

S32 SDK Release Notes

Version 3.0.2 RTM

Contents

1. Description

The S32 Software Development Kit (S32 SDK) is an extensive suite of peripheral abstraction layers, peripheral drivers, RTOS, stacks and middleware designed to simplify and accelerate application development on NXP S32K microcontrollers.

All software included in this release have RTM quality level in terms of features, testing and quality documentation, according to NXP software release criteria. RTM releases contain all planned features implemented and tested. RTM releases are candidates that can be used in production.

This is a service release designated for updating S32K1xx SDK RTM 3.0.0; it should be installed on top of the previous RTM.

This SDK can be used standalone or it can be used with S32 Design Studio IDE (see Supported hardware and compatible software).

Refer to *Product license (License.txt)* for licensing information and *Software content register (SW-Content-Register-S32-SDK.txt)* for the Software contents of this product. The files can be found in the root of the installation directory.

For support and issue reporting use the following ways of contact:

- NXP Support to https://www.nxp.com/support/support:SUPPORTHOME
- NXP Community https://community.nxp.com/community/s32/s32k

2. New in this release

2.1 New features from RTM 3.0.1

SBC_UJA116x (formerly SBC_UJA1169)

- Driver now supports UJA1169 as well as both UJA1168 and UJA1168A SBC variants.

Added SBC_FS45 driver for S32K148

Integrated AMMCLib RTM 1.1.18

2.2 List of fixed issues from RTM 3.0.1

Component	Issue
examples	Updated examples doxygen folder name to start with capital letter to be consistent across all platforms.
examples	Updated FreeRTOS examples for S32K116 and S32K118 with correct value for the EmbSys View.
examples	Changed target CPU family was not propagated into the example makefiles
rtc	Fixed the interrupt handler generations for RTC PEx component. Interrupt handlers will not be declared if the user does not set one of the interrupt sources.

3. Software Contents

3.1 Drivers

- ADC
- CMP
- CRC
- CSEc
- DMA
- EIM
- ENET
- ERM
- EWM
- FLASH
- FLASH_MX25L6433F
- FLEXCAN
- FLEXIO (I2C, SPI, I2S, UART profiles)
- FTM
- LIN
- LPI2C
- LPIT
- LPSPI
- LPTMR
- LPUART
- MCU (Clock Manager, Interrupt Manager, Power Manager)
- MPU
- PINS
- PDB
- PHY_TJA110x
- QSPI
- RTC
- SAI
- TRGMUX
- WDOG

3.2 PAL

- ADC
- CAN
- I2C
- I2S
- IC
- MPU
- OC
- PWM
- SECURITY

- SPI
- TIMING
- UART
- WDG

3.3 Middleware

- LIN stack – provides support for LIN 2.1, LIN 2.2 and J2602 communication protocols
- TCP/IP stack – available for S32K148, for more details see TCP/IP stack release notes (in the SDK installation folder)
- SBC drivers – provides support for UJA116x, UJA113x and FS45 System Basis Chips

Note: *For ISELED and NFC contact your Sales representative or FAE for more information.*

3.4 Libraries

- sCST RTM 1.0.0 – available for S32K11x
- sCST RTM 1.0.4 – available for S32K14x
- AMMCLib RTM 1.1.18 – available for S32K1xx

3.5 RTOS

- FreeRTOS version 10.0.1

4. Documentation

- Quick start guide available in *“doc”* folder.
- User and integration manual available at *“doc\Start_here.html”.*
- Driver user manuals available in *“doc”* folder*.*

5. Examples

Type	Name	Description
Driver examples	adc_hwtrigger	Uses PDB to trigger an ADC conversion with a configured delay and sends the result to host via LPUART.
	adc_pal_example	The application uses ADC PAL to trigger multiple executions of two groups of ADC conversions: first group configured for SW triggering and second group for HW triggering. For each execution of a group of conversions, an average conversion value is computed in SW, and the average value is printed on UART.
	adc_swtrigger	Uses software trigger to periodically trigger an ADC conversion and sends the result to host via LPUART.
	can_pal	Shows the usage of the CAN PAL module with Flexible Data Rate
	cmp_dac	Configures the analog comparator to compare the input from the potentiometer with the internal DAC (configured to output half of the reference voltage) and shows the result using the LEDs found on the board.
	crc_checksum	The CRC is configured to generate the cyclic redundancy check value using 16 and 32 bits wide result.
	csec_keyconfig	The example demonstrates how to prepare the MCU before using CSEc(Key configuration, flash partitioning).
	edma_transfer	Demonstrates the following eDMA use cases: single block memory to memory transfer, a loop memory to memory transfer, memory to memory transfer using scatter/gather, LPUART transmission/reception using DMA requests.
	eim_injection	The purpose of this demo is to provide the user check able correction of ECC. Module EIM enable user addition error to RAM (low). And enable user can use module ERM to read address that user already error to region RAM. User seen RED_LED off when ERM read right address which EIM injected error.
	enet_ping	Shows the usage of a basic ping application using the ENET driver
	erm_report	The purpose of this driver application is to show the user how to use the EWM from the S32K148 using the S32 SDK API. This Example only debug equal Flash This example use module EIM to addition error to RAM and use module ERM to read address and notify interrupt.

	ewm_interrupt	Shows the usage of the EWM driver.
	flash_partitioning	Writes, verifies and erases data on Flash.
	flexio_i2c	Demonstrates FlexIO I2C emulation. Use one instance of FlexIO and one instance of LPI2C to transfer data on the same board.
	flexio_i2s	Demonstrates FlexIO I2S emulation for both master and slave configurations. Use one instance of FlexIO to instantiate master and slave drivers to transfer data on the same board.
	flexio_spi	Demonstrates FlexIO SPI emulation for both master and slave configurations. Use one instance of FlexIO to instantiate master and slave drivers to transfer data on the same board.
	flexio_uart	Demonstrates FlexIO UART emulation for both TX and RX configurations. Use one instance of FlexIO to instantiate UART transmitter and receiver drivers to transfer data from/to the host.
	ftm_combined_pwm	Uses FTM PWM functionality using two combined channels to light two LEDs on the board with opposite pulse width. The light's intensity is increased and decreased periodically.
	ftm_periodic_interrupt	Uses FTM Timer functionality to trigger an interrupt at a given period which toggles a LED.
	ftm_pwm	Uses FTM PWM functionality using a single channel to light a LED on the board. The light's intensity is increased and decreased periodically.
	ftm_signal_measurement	Using one FTM instance the example application generates a PWM signal with variable frequency which is measured by another FTM instance configured in signal measurement mode.
	i2c_pal	Shows the usage of I2C PAL driver in both master and slave configurations using FLEXIO and LPI2C
	ic_pal	Shows the usage of the IC_PAL
	lin_master_baremetal	Shows the usage of LIN driver in master mode.
	lin_slave_baremetal	Shows the usage of LIN driver in slave mode.
	lpi2c_master	Shows the usage of the LPI2C driver in Master configuration
	lpi2c_slave	Shows the usage of the LPI2C driver in Slave configuration
	lpit_periodic_interrupt	Shows how to initialize the LPIT to generate an interrupt every 1 s. It is the starting point for any application using LPIT.
	lpspi_dma	The application uses two on board instances of LPSPI, one in master configuration and the other one is slave to communicate data via the SPI bus using DMA.

	lpspi_transfer	Uses one instance of the LPSPI as slave to send ADC data to the master LPSPI instance which is on the same board. The master uses data received to feed a FlexTimer PWM.
	lptmr_periodic_interrupt	Exemplifies to the user how to initialize the LPTIMER so that it will generate an interrupt every 1 second. To make the interrupt visible a LED is toggled every time it occurs.
	lptmr_pulse_counter	Shows the LPTIMER pulse count functionality by generating an interrupt every 4 rising edges.
	lpuart_echo	Simple example of a basic echo using LPUART.
	mpu_memory_protection	Configures MPU to protect a memory area and demonstrates that read access is correctly restricted.
	mpu_pal_memory_protection	The purpose of this demo application is to show you how to configure and use the Memory Protection Unit PAL
	oc_pal	Shows the Periodic Event Generation functionality of the OC_PAL
	pdb_periodic_interrupt	Configures the Programmable Delay Block to generate an interrupt every 1 second. This example shows the user how to configure the PDB timer for interrupt generation. The PDB is configured to trigger ADC conversions in ADC_HwTrigger_Example.
	phy_autoneg	Shows the usage of the PHY module with autonegociation
	power_mode_switch	Demonstrates the usage of Power Manager by allowing the user to switch to all power modes available.
	qspi_external_flash	The purpose of this demo is to present the usage of the flash_mx25l6433f (external serial flash) and QSPI drivers. The flash_mx25l6433f driver allows the application to use an external Macronix MX25L6433F serial flash device, using the QuadSPI interface for communication.
	rtc_alarm	Show the frequently used RTC use cases such as the generation of an interrupt every second and triggering an alarm.
	sai_transfer	Demonstrates the usage of the SAI module driver
	sbc_uja113x	Show the usage of the SBC UJA113x driver with low power modes
	sbc_uja1169	Show the usage of the SBC UJA1169 driver with low power modes
	security_pal	This is an application created to show the generation of rnd and CBC encryption/decryption of a string.
	spi_pal	The purpose of this application is to show you how to use the LPSPI and FLEXIO Interfaces on the S32K144 using the S32 SDK API.

		The application uses one board instance of LPSPI in slave configuration and one board instance of FLEXIO in master configuration to communicate data via the SPI bus using interrupts.
	timing_pal	The purpose of this application is to show you how to use the TIMING PAL over LPIT, LPTMR and FTM timers on the S32K144 using the S32 SDK API. The application uses one board instance of LPIT, LPTMR and FTM to periodically toggle 3 leds.
	trgmux_lpit	The purpose of this demo application is to show you how to use the Trigger MUX Control of the S32K14x MCU with this SDK.
	uart_pal_echo	The purpose of this demo is to show the user how UART PAL works over FLEXIO_UART or LPUART peripherals. The user can choose whether to use FLEXIO_UART or LPUART. The board sends a welcome message to the console with further instructions.
	wdg_pal_interrupt	The purpose of this driver application is to show the user how to use the WDG PAL from the S32K148 using the S32 SDK API. The examples uses the SysTick timer from the ARM core to refresh the WDG PAL counter for 30 times. After this the WDG PAL counter will expire and the CPU will be reset.
	wdog_interrupt	Shows the basic usage scenario and configuration for the Watchdog.
Demos	adc_low_power	This demo shows the user how to reduce CPU overhead and power usage by triggering ADC conversions with the LPIT via TRGMUX. The CPU is set in the STOP mode via the Power Manager API, with the wakeup condition being the validity of the ADC conversion result, the latter being a value greater than half of the ADC reference voltage achieved by using the hardware compare functionality. If the condition is met, the value in the form of a graph is sent using LPUART and DMA to further reduce the CPU usage.
	anfc	Shows the integration between Automotive NFC stack and S32SDK
	flexcan_encrypted	Uses two boards to demonstrate FlexCAN functionality with Flexible Data Rate on. LEDs on a board are toggled depending on the buttons actioned on the other board. Also demonstrates the use of SBC driver to configure the CAN transceiver from EVB board. The application is configured to use CSEc to encrypt the data on security enabled parts.
	freemaster	This demo uses the FreeMASTER Run-Time Debugging Tool to visualize ADC

		conversions and allows the user to monitor the ADC sampling rate for different ADC configurations (ADC sampling time and resolution can be controlled through FreeMASTER Variable Watch). The application uses FreeMASTER SCI driver for communication.
	freertos	This demo application demonstrates the usage of the SDK with the included FreeRTOS. Uses a software timer to trigger a led and waits for a button interrupt to occur.
	hello_world	This is a simple application created to show the basic configuration with S32DS
	hello_world_iar	This is a simple application created to show the basic configuration with IAR Embedded Workbench
	hello_world_mkf	This is a simple application created to show the basic configuration with makefile for the supported compilers
	lin_master	This demo application shows the usage of LIN stack in master mode.
	lin_slave	This demo application shows the usage of LIN stack in slave mode.
	lwip	Shows the usage of lwIP stack.
	sCST	Demo application created to demonstrate sCST integration with S32 SDK

6. Supported hardware and compatible software

6.1 CPUs

- S32K116_32 revision 1.0, maskset 0N96V
- S32K116_48 revision 1.0, maskset 0N96V
- S32K118_48 revision 1.0, maskset 0N97V
- S32K116_64 revision 1.0, maskset 0N97V
- S32K142_48 revision 1.0, maskset 0N33V
- S32K142_64 revision 1.0, maskset 0N33V
- S32K142_100 revision 1.0, maskset 0N33V
- S32K144_48 revision 2.1, maskset 0N57U
- S32K144_64 revision 2.1, maskset 0N57U
- S32K144_100_LQFP revision 2.1, maskset 0N57U
- S32K144_100_BGA revision 2.1, maskset 0N57U
- S32K146_64 revision 1.0, maskset 0N73V
- S32K146_100_LQFP revision 1.0, maskset 0N73V
- S32K146_100_BGA revision 1.0, maskset 0N73V
- S32K146_144 revision 1.0, maskset 0N73V
- S32K148_100_LQFP revision 1.0, maskset 0N20V
- S32K148_100_BGA revision 1.0, maskset 0N20V
- S32K148_144 revision 1.0, maskset 0N20V
- S32K148_176 revision 1.0, maskset 0N20V

The following processor reference manual has been used to add support:

- S32K1XXRM Rev. 9, 09/2018

6.2 Boards

- S32K-MB with mini module S32K144-100LQFP REV X1/X2
- S32K-MB with mini module S32K14xCVD-Q144 REV X3
- S32K-MB with mini module S32K1xxCVD-Q048 REV X1
- S32K-MB with mini module S32K1xxCVD-Q064 REV X2
- S32K144-EVB-Q100 REV X3
- S32K148-EVB-Q144 REV X2
- S32K142-EVB-Q100 REV X1
- S32K146-EVB-Q144 REV X1
- S32K116-EVB-Q048 REV X2
- S32K118-EVB-Q064 REV X2

6.3 Compiler and IDE versions

- GreenHills compiler v. 2017.1.4
- IAR compiler v. 8.11.2
- GCC compiler for ARM v. 6.3.1 20170509
- Wind River Diab Compiler v5.9.6.2
- ARM Compiler 6.6.1 Long Term Maintenance
- S32 Design Studio v2018.R1 IDE

7. Known issues and limitations

7.1 Standalone installation

- The installer will automatically append the new SDK path to the S32SDK_PATH variable. Please make sure that only the desired value is kept, if the variable is used by previous projects.
- Uninstalling the SDK will not remove references to it from S32 Design Studio, this will result in a broken path displayed in Window->Preferences->Processor Expert.
- Custom installation type is not fully supported, keep “All Packages” selection in Choose Components page.

7.2 Drivers

ALL DRIVERS

- Drivers may not respect the requirements for nesting level and cyclomatic complexity due to an issue in tools.

ADC_PAL, PDB

- instanceBackToBackEnable not initialized before use in ADC_ConfigGroup function and in PDB_DRV_GetDefaultConfig of the PDB. This can lead to unexpected configuration of the PDB

CPU

- When using DIAB toolchain on S32K11x and the interrupt handlers are overwritten with INT_SYS_InstallHandler, the core will not return from interrupt handlers that are not calling other functions or writing a global variable. Workaround: Make sure that all interrupt handlers are performing at least one function call or are writing a global variable.

CLOCK

- CLOCK_SYS_GetFreq function returns obsolete core clock frequency right after VLPR to HSRUN power mode transition because SCS bitfield from SCG_CSR register is not immediately updated (workaround: function to be called twice, second call returns correct value).

EDMA

- The naming of the EDMA request for ENET MAC0 TIMER did not have the same format as the other EDMA requests.

EIM

- An attempt to invert more than 2 bits in check bit mask or data mask might result in undefined behavior. To avoid this situation, you should invert a maximum of two bits.

ENET

- The wakeup interrupt is not enabled

FlexCAN

- In CAN-FD mode, the FLEXCAN_DRV_GetBitrate function can return wrong values.
- In case of using DMA transfer on RxFIFO avoid initiating another transfer from a callback function installed after DMA complete event

FlexIO, SAI

- FlexIO drivers and the SAI driver cannot be simultaneously used in DMA mode due to overlapping DMA requests.

FlexIO_I2C

- No STOP condition is generated when aborting a transfer due to NACK reception.
- No clock stretching when the application does not provide data fast enough, so Tx underflows and Rx overflows are possible.
- There is a maximum limit of 13 bytes on the size of any transfer.
- The driver does not support multi-master mode. It does not detect arbitration loss condition.
- Due to device limitations, it is not always possible to tell the difference between NACK reception and receiver overflow.

Note: FLEXIO I2C issues described above are caused by Hardware limitations.

FlexIO_SPI

- The driver does not support back-to-back transmission mode for CPHA = 1

FTM

- Module can be used only in one mode (e.g. only PWM, OC). For example, this configuration is not possible: 4 channels of FTM0 run in PWM and 4 channels of FTM0 run in input capture.
- Complementary channel is not enabled in all configurations for independent channels. The workaround is to use complementary channel only for combined channels.

FreeRTOS

- The UI configuration does not open method definition when the method is double-clicked in the method list.

LIN, LinStack

- Build will fain when adding LIN or Lin Stack components into a project that uses GHS in S32 Design Studio. Workaround: manually add eDMA component into the project
- LIN master node do not go to sleep mode right after sending sleep command.

LinStack

- Function ld_read_configuration return PIDs instead of ID

LPSPI

- When a SPI transfer in slave mode over DMA is initialized with an invalid address for the tx buffer, the driver can never finish the transfer

I2C_PAL, LPI2C

- When (LPI2C|I2C)_MasterAbortTransfer is called after a transfer operation was started and the address was not sent, the bus may hang. Workaround is to avoid calling the function shortly after a transfer was initiated.

LPI2C

- LPI2C_DRV_MasterAbortTransferData function can’t abort a master receive transfer because the module sees the whole receive as a single operation and will not stop it even if the FIFO is reset.
- Minimum allowed value for HIGH period of the SCL is sometimes violated in standard mode.

RTC

- When using LPO clock as input, the user may need to use LPO trimming to obtain the 32kHz frequency needed by RTC module.

SBC

- SBC_UJA116x: SBC_Init function clears all event status registers, so information about device reset cause is lost.
- SBC_FS45: FS45_LIN_SetMode function is not supported for this release. In code this function is available, but it is up to the user to enable it by defining FS45_FEATURE_LIN macro in the driver.

Timing_PAL

- For TIMING_PAL over FTM, the timer channel 6 and 7 operate incorrectly because the FTM7_Ch6_Ch7_IrqHandler is wrong.

7.3 Examples

- Running the FLASH driver example from the flash will secure the device. To unsecure the MCU a mass erase of the flash needs to be done.
- Redundant code for configuring pins can be found in the examples.
- Hello World project S32K146 cannot be debugged on IAR IDE, since the IDE version supported by the SDK does not support S32K146.
- After partitioning Flash for CSEc operation, using the JLink Flash configuration of any other project will not work anymore.
- Workaround:
- - Run csec_keyconfig example with ERASE_ALL_KEYS 0, using PEmicro debug configuration
- - Run csec_keyconfig example with ERASE_ALL_KEYS 1, using PEmicro debug configuration
- Example projects for IAR Embedded Workbench use simulator as default debugger. The user has to manually select and configure the debug probe prior to downloading to the target.
- FLASH partitioning example should be run in RAM configuration.
- An internal error may appear upon importing LPUART example for S32K148; clicking the Generate code button once again fixes the error and the example works fine.

8. Compiler options

The example projects are using the first level of optimizations (low optimizations).

For exceptions from the following compiler settings, additional information can be found in the SDK documentation, Build Tools section.

8.1 IAR Compiler/Linker/Assembler Options

Table 8.1 IAR Compiler Options

Option	Description
-Ol	Low optimizations
-e	Allow IAR extensions
--cpu=Cortex-M4 / --cpu Cortex-M0+	Selects target processor: Arm Cortex M4 / Arm Cortex M0+
--thumb	Selects generating code that executes in Thumb state.
--fpu VFPv4_sp / --fpu none	Use floating point instructions / Use software floating point
--debug	Include debug information
-D<cpu_define>	Define a preprocessor symbol for MCU
-warnings_are_errors	Treat code warnings as errors

Table 8.2 IAR Assembler Options

Option	Description
--cpu Cortex-M4 / --cpu Cortex-M0+	Selects target processor: Arm Cortex M4 / Arm Cortex M0+
--thumb	Selects generating code that executes in Thumb state.
--fpu VFPv4_sp / --fpu none	Use floating point instructions / Use software floating point
-DSTART_FROM_FLASH	Mandatory when flash target is used

Table 8.3 IAR Linker Options

Option	Description
--cpu Cortex-M4 / --cpu Cortex-M0+	Selects target processor: Arm Cortex M4 / Arm Cortex M0+
--thumb	Selects generating code that executes in Thumb state.
--fpu VFPv4_sp / --fpu none	Use floating point instructions / Use software floating point
--map <map_file>	Produce a linker memory map file
--entry Reset_Handler	Make the symbol Reset_Handler be treated as a root symbol and the start label of the

	application
--config <linker_file.icf>	Use the specified linker file

8.2 GCC Compiler/Linker/Assembler Options

Table 8.4 GCC Compiler Options

Option	Description
-mcpu=cortex-m4 / -mcpu=cortex-m0plus	Selects target processor: Arm Cortex M4 / Arm Cortex M0+
-mthumb	Selects generating code that executes in Thumb state.
-O1	Optimize
-funsigned-char	Let the type char be unsigned, like unsigned char
-funsigned-bitfields	Bit-fields are signed by default
-fshort-enums	Allocate to an enum type only as many bytes as it needs for the declared range of possible values.
-ffunction-sections	Place each function into its own section in the output file
-fdata-sections	Place data item into its own section in the output file
-fno-jump-tables	Do not use jump tables for switch statements
-std=c99	Use C99 standard
-g	Generate debug information
-D<cpu_define>	Define a preprocessor symbol for MCU
-mfloat-abi=hard / -mfloat-abi=soft	Use FPU instructions / Use software FP
-mfpu=fpv4-sp-d16	Specify the FPU variant (only for S32K14x)
-Wall	Produce warnings about questionable constructs
-Wextra	Produce extra warnings that -Wall
-Wstrict-prototypes	Warn if a function is declared or defined without specifying the argument types.
-pedantic	Issue all the warnings demanded by strict ISO C
-Wunused	Produce warnings for unused variables
-Werror	Treat warnings as errors
-Wsign-compare	Produce warnings when comparing signed type with unsigned type

Table 8.5 GCC Assembler Options

Option	Description
-mcpu=cortex-m4 / -mcpu=cortex-m0plus	Selects target processor: Arm Cortex M4 / Arm Cortex M0+
-mthumb	Selects generating code that executes in Thumb state.

-mfloat-abi=hard / -mfloat-abi=soft	Use FPU instructions / Use software FP
-mfpu=fpv4-sp-d16	Specify the FPU variant (only for S32K14x)
-Wall	Produce warnings about questionable constructs
-Wextra	Produce extra warnings that -Wall
-Wstrict-prototypes	Warn if a function is declared or defined without specifying the argument types.
-pedantic	Issue all the warnings demanded by strict ISO C
-Werror	Treat warnings as errors
-x assembler-with-cpp	Preprocess assembly files
-DSTART_FROM_FLASH	Mandatory when flash target is used

Table 8.6 GCC Linker Options

Option	Description
-mcpu=cortex-m4 / -mcpu=cortex-m0plus	Selects target processor: Arm Cortex M4 / Arm Cortex M0+
-mthumb	Selects generating code that executes in Thumb state.
--entry=Reset_Handler	Make the symbol Reset_Handler be treated as a root symbol and the start label of the application
-T<linker_file.ld>	Use the specified linker file
-mfloat-abi=hard / -mfloat-abi=soft	Use FPU instructions / Use software FP
-mfpu=fpv4-sp-d16	Specify the FPU variant (only for S32K14x)
-Xlinker –gc-sections	Remove unused sections
-Wl, -Map=<map_file>	Produce a map file
-lgcc	Link libgcc
-lc	Link C library
-lm	Link Math library

8.3 GHS Compiler/Linker/Assembler Options

Table 8.7 GHS Compiler Options

Option	Description
-cpu=cortexm0plus / -cpu=cortexm4	Selects target processor: Arm Cortex M4 / Arm Cortex M0+
-thumb	Selects generating code that executes in Thumb state.
-fhard / -fsoft	Use FPU instructions / Use software FP
-fpu=vfpv4_d16	Specify FPU type (only for S32K14x)
-c99	Use C99 standard
--gnu_asm	Enables GNU extended asm syntax support
-Ogeneral	Optimize
-gdwarf-2	Generate DWARF 2.0 debug information

-G	Generate debug information
-D<cpu_define>	Define a preprocessor symbol for MCU
--quit_after_warnings	Treat warnings as errors
-Wimplicit-int	Produce warnings if functions are assumed to return int
-Wshadow	Produce warnings if variables are shadowed
-Wtrigraphs	Produce warnings if trigraphs are detected
-Wundef	Produce a warning if undefined identifiers are used in #if preprocessor statements

Table 8.8 GHS Assembler Options

Option	Description
-cpu=cortexm0plus / -cpu=cortexm4	Selects target processor: Arm Cortex M4 / Arm Cortex M0+
-fhard / -fsoft	Use FPU instructions / Use software FP
-fpu=vfpv4_d16	Specify FPU type (only for S32K14x)
-preprocess_assembly_files	Preprocess assembly files
DSTART_FROM_FLASH	Mandatory when flash target is used

Table 8.9 GHS Linker Options

Option	Description
-cpu=cortexm0plus / -cpu=cortexm4	Selects target processor: Arm Cortex M4 / Arm Cortex M0+
-thumb	Selects generating code that executes in Thumb state.
-entry=Reset_Handler	Make the symbol Reset_Handler be treated as a root symbol and the start label of the application
-T<linker_file.ld>	Use the specified linker file
-map=<map_file>	Produce a map file
-larch	Link architecture specific library

8.4 DIAB Compiler/Linker/Assembler Options

Table 8.10 DIAB Compiler Options

Option	Description
-tARMCORTEXM4LV / -tARMCORTEXM0PLS	Selects target processor: Arm Cortex M4 / Arm Cortex M0+
-mthumb	Selects generating code that executes in Thumb state.
-Xdialect-c99	Use C99 standard
-D<cpu_define>	Define a preprocessor symbol for MCU
-g	Add debug information to the executable

-O	Optimize
-Xstop-on-warning	Treat warnings as errors

Table 8.11 DIAB Assembler Options

Option	Description
-tARMCORTEXM4LV / -tARMCORTEXM0PLS	Selects target processor: Arm Cortex M4 / Arm Cortex M0+
-mthumb	Selects generating code that executes in Thumb state.
-Xpreprocess-assembly	Preprocess assembly files
-DSTART_FROM_FLASH	Mandatory when flash target is used

Table 8.12 DIAB Linker Options

Option	Description
-tARMCORTEXM4LV / -tARMCORTEXM0PLS	Selects target processor: Arm Cortex M4 / Arm Cortex M0+
-Xremove-unused-sections	Removes unused code sections
-lc	Link the standard C library to the project in order to support elementary operations that are used by the drivers
-lm	Link the standard math library to the project in order to support elementary math operations that are used by the drivers
<linker_file.dld>	Use the specified linker file
-e Reset_Handler	Make the symbol Reset_Handler be treated as a root symbol and the start label of the application
-m6 > <map_file>	Produce a linker map

8.5 ARMC Compiler/Linker/Assembler Options

Table 8.13 ARMC Compiler Options

Option	Description
--target=arm-arm-none-eabi	Select arm-none-eabi as target architecture
--cpu=cortex-m4 / --cpu=cortex-m0plus	Selects target processor: Arm Cortex M4 / Arm Cortex M0+
-mthumb	Selects generating code that executes in Thumb state.
-O1	Optimize
-fshort-enums	Allocate to an enum type only as many bytes as it needs for the declared range of possible values.
-fdata-sections	Place data item into its own section in the output file
-std=c99	Use C99 standard

-g	Generate debug information
-D<cpu_define>	Define a preprocessor symbol for MCU
-mfloat-abi=hard / -mfloat-abi=soft	Use FPU instructions / Use software FP
-pedantic	Issue all the warnings demanded by strict ISO C
-Weverything	Produce warnings for unused variables
-Werror	Treat warnings as errors
-Wno-switch-enum	Do not issue warnings for enum values that are not explicitly treated in switch statements
-Wno-cast-align	Do not issue warnings for cast statements that increase the required alignment
-Wno-cast-qual	Do not issue warnings for cast statements that are discarding const qualifier.
-Wno-covered-switch-default	Do not issue warnings for “default” switch case being present when all enum values are covered in a switch
-Wno-reserved-id-macro	Do not issue warnings when macros starting with double underscore (e.g. __IO) are present in the code.
-Wno-padded	Do not issue warnings when padding is added.

Table 8.14 ARMC Assembler Options

Option	Description
--target=arm-arm-none-eabi	Select arm-none-eabi as target architecture
--cpu=cortex-m4 / --cpu=cortex-m0plus	Selects target processor: Arm Cortex M4 / Arm Cortex M0+
-mfloat-abi=hard / -mfloat-abi=soft	Use FPU instructions / Use software FP
--cpreproc	Instructs the assembler to call armcc to preprocess the input file before assembling it
--cpreproc_opts	Enables the assembler to pass options to the compiler when using the C preprocessor
-DSTART_FROM_FLASH	Mandatory define when flash target is used

Table 8.15 ARMC Linker Options

Option	Description
--target=arm-arm-none-eabi	Select arm-none-eabi as target architecture
--cpu=cortex-m4 / --cpu=cortex-m0plus	Selects target processor: Arm Cortex M4 / Arm Cortex M0+
--entry Reset_Handler	Make the symbol Reset_Handler be treated as a root symbol and the start label of the application
--scatter “<scatter_file>”	Use the specified scatter file
--datacompressor off	Turn off compression for data sections
--map	Produce a map file
--list=<map_file>	Assign a file for the map
--symbols	Save the symbol information in the map file

Note: The symbol <linker_file> must be replaced with the corresponding path and linker file name per device, memory model and target compiler.

E.g. *C:\NXP\S32_SDK\platform\devices\S32K144\linker\gcc\S32K144_64_flash.ld* - for S32K144, 64 KB of SRAM and Flash target on GCC.

Symbol <map_file> shall be replaced with the desired map file name.

Symbol <cpu_define> shall be replaced with CPU_S32K144HFT0VLLT for S32K144, CPU_S32K148 for S32K148, CPU_S32K142 for S32K142 and CPU_S32K146 for S32K146.

9. Acronyms

Acronym	Description
EAR	Early Access Release
JRE	Java Runtime Environment
EVB	Evaluation board
PAL	Peripheral Abstraction Layer
RTOS	Real Time Operating System
PEx	Processor Expert Configurator
PD	Peripheral Driver
RTM	Ready to Manufacture
S32DS	S32 Design Studio IDE
SDK	Software Development Kit
SOC	System-on-Chip
sCST	Structural Core Self Test

10. Version Tracking

Date (dd-Mmm-YYYY)	Version	Comments	Author
30-Oct-2015	1.0	First version for EAR 0.8.0	Vlad Baragan-Stroe
18-Dec-2015	1.1	Added patch 1	Vlad Baragan-Stroe
01-Apr-2016	2.0	Added drivers, new in release section, updated examples, known limitations for EAR 0.8.1	Vlad Baragan-Stroe
27-Oct-2016	3.0	Updated new in this release section, known limitations and examples description for EAR 0.8.2 release. Added “Compiler options” section. Updated header, footer and front page with new logos	Rares Vasile
21-Dec-2016	4.0	Updated Release Notes for 0.9.0 BETA release	Rares Vasile
23-Mar-2017	5.0	Updated Release Notes for 1.0.0 RTM release	Rares Vasile
04-May-2017	6.0	Updated Release Notes for 0.8.3 EAR release	Rares Vasile
10-May-2017	6.1	Updated Release Notes for 0.8.3 EAR release - Added drivers, new in release section, updated examples, known limitations for EAR 0.8.3	Cezar Dobromir
27-Jun-2017	7.0	Updated for EAR 0.8.4 release	Rares Vasile
31-Aug-2017	8.0	Updated for EAR 0.8.5 release	Rares Vasile
27-Nov-2017	9.0	Updated for EAR 0.8.6 release	Rares Vasile
3-May-2018	10.0	Updated for BETA 1.9.0 release	Rares Vasile
26-Jun-2018	11.0	Updated for RTM 2.0.0 release	Rares Vasile
21-Aug-2018	12.0	Updated for BETA 2.9.0 release	Rares Vasile
21-Nov-2018	13.0	Updated for BETA 2.9.2 release	Vlad Lionte
21-Feb-2019	14.0	Updated for RTM 3.0.0 release	Vlad Lionte
28-Mar-2019	14.1	Updated for RTM 3.0.1 service release	Vlad Lionte
11-Oct-2019	14.2	Updated for RTM 3.0.2 service release	Ovidiu-Marius Alexandru