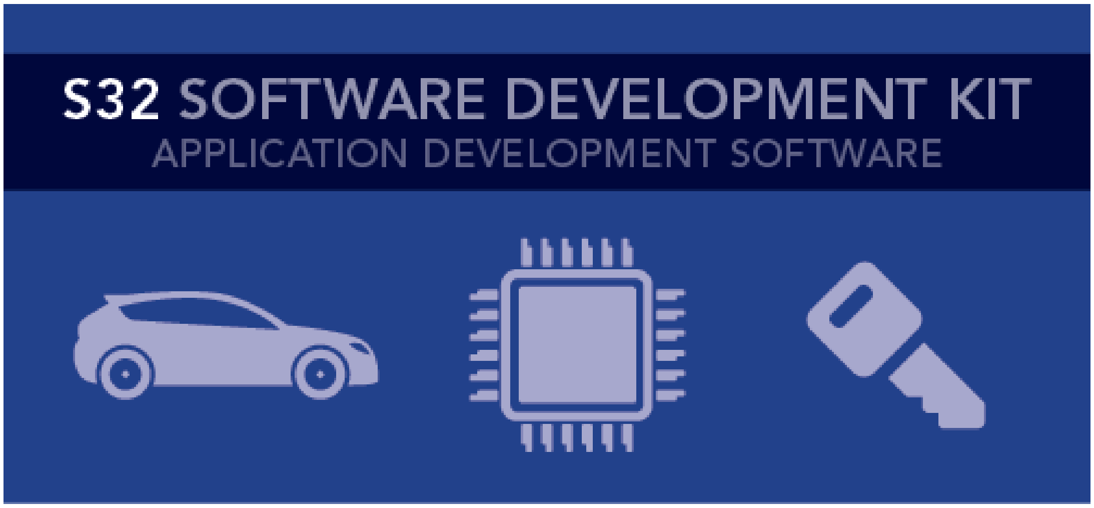
S32 SOFTWARE DEVELOPMENT KIT
APPLICATION DEVELOPMENT SOFTWARE

S32 SDK Release Notes

Version 1.0.0 RTM

Contents

1. Description

The S32 Software Development Kit (S32 SDK) is an extensive suite of robust hardware interface and hardware 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 has RTM quality level in terms of features, testing and quality documentation, according to NXP software release criteria.

This SDK can be used as is (see Documentation) or it can be used with S32 Design Studio IDE.

Refer to *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:

- Email to support@nxp.com
- NXP Community https://community.nxp.com/

2. New in this release

2.1 Examples

- Added new driver examples for SBC UJA1169, AMMCLib
- Aligned folder and project naming
- Removed Clock Setup, FreeMASTER BDM, FreeRTOS IAR, FreeRTOS makefile and LIN + FreeRTOS examples
- Enabled “Print Size” option for examples using S32 DS

2.2 Drivers:

- **Header**
  - Updated as per S32K14XRM Rev. 2, 02/2017
- **Libraries**
  - Integrated AMMCLib v1.1.7 free version
- **LIN**
  - Updated support for autobaud in Slave Mode
- **LIN Stack**
  - Updated handling of flag of frames and signal flags that are automatically set by the LIN core and can only be tested or cleared by the application program.
  - Added flags for master request and slave response frames.
  - Fixed word status byte

3. Software Contents

1. Drivers (HAL and PD)

- ADC
- CMP
- CRC
- CSEc
- DMA
- EWM
- EIM
- ERM
- FLASH
- FLEXCAN
- FLEXIO (I2C, SPI, I2S, UART profiles)
- FTM
- GPIO
- LIN
- LPI2C
- LPIT
- LPSPI
- LPTMR
- LPUART
- MCU (ClockManager, InterruptManager, PowerManager)
- MPU
- PDB
- PORT
- RTC
- TRGMUX
- WDOG

2. Middleware

- LIN stack - provides support for LIN2.1 and J2602 communication protocols
- SBC stack - provides support for UJA1169 System Basis Chips

3. RTOS

- FreeRTOS version 8.2.1

4. Libraries

- AMMCLib version 1.1.7

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_swtrigger	Uses software trigger to periodically trigger an ADC conversion and sends the result to host via LPUART.
	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.
	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_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_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_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_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_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_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.
	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_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.
	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.
	power_mode_switch	Demonstrates the usage of Power Manager by allowing the user to switch to all power modes available.
	sbc_uja1169	Show the usage of the SBC driver with low power modes
	rtc_alarm	Show the frequently used RTC use cases such as the generation of an interrupt every second and triggering an alarm.
	wdog_interrupt	Shows the basic usage scenario and configuration for the Watchdog.
Demos	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

	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.
	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.
	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.
	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.
	ammclib	Demo application created to demonstrate AMMCLib integration with S32 SDK.
	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.

6. Quality Package

Quality package is available for S32 SDK S32K144 RTM 1.0.0. The quality package contains the following documents for each driver:

- MISRA Report
- Test Specification
- Test Report
- PEx Test Report
- PEx Test Specification
- Compiler Warnings Report
- Code Coverage Summary Report
- Static Analysis Report
- Traceability Matrix Report
- Code Size Report
- RAM Size Report.

Please contact your FAE or sales representative for details on how to get this package.

7. Supported hardware and compatible software

CPUs

- S32K144_64 revision 2.1, maskset 0N57U
- S32K144_100 revision 2.1, maskset 0N57U
- S32K144_100_BGA revision 2.1, maskset 0N57U

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

- S32K14XRM Rev. 2, 02/2017

Boards

- S32K144-MB with mini module S32K144-100LQFP REV X1/X2
- S32K144-EVB-Q100 (**CPU maskset 0N47T**)

Compiler and IDE versions:

- GreenHills compiler v. 2015.1.4
- IAR compiler v. 7.50.3
- GCC compiler for ARM v. 4.9.3 20150529
- COSMIC Software CORTEX-M C Cross Compiler v4.3.10
- Wind River Diab Compiler v5.9.4.8
- S32 Design Studio v1.3 IDE

Debuggers:

- SEGGER J-Link (with SEGGERGDB Server)
- P&E Multilink (with P&E GDB Server)

8. Known problems and limitations

1. Installer

- Due to an installer issue, before installing the new SDK, please make sure that the S32SDK_PATH environment variable is empty. This can be done either manually or by uninstalling the previous SDK version using the uninstall provided.
- The uninstaller does not delete configuration files copied in S32 DS build.
- If the FreeRTOS component is not selected to be installed, then the drivers will be affected as the OSIF component will not be installed.
- Custom install type is not fully supported, keep “AllPackages” selection in Choose Components page.

2. Drivers

CLOCK

- Clock Manager PEx component shows lower maximum range for SOSC. It should be 8-40 MHz, but it is shown as 8-32 MHz.

CMP

- CMP_HAL_GetFunctionalMode will return a wrong mode when it is set as CMP_SAMPLED_NONFILTRED_INT_CLK with CMP_HAL_SetFunctionalMode

CPU

- If main function is exited the CPU will remain blocked in an infinite loop from startup_S32K144.S
- Doxygen Documentation link is missing from the CPU ProcessorExpert Component.

EIM

- If more than 2 bits are flipped in DATA_MASK or CHECKBIT_MASK bitfields in EIM control registers that there is no guarantee in design what type of error is generated

FLASH

- If the application uses both flash and asynchronous CSEc mode or uses flash and CSEc drivers from different threads, it must ensure that they don’t execute commands at the same time. The flash and CSEc drivers don’t currently implement an inter-synchronization mechanism, so the application will have to ensure it if needed.

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. For example, this configuration is not possible: 4 channel of FTM0 run in PWM and 4 channel of FTM0 run in input capture.

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.

OSIF

- Current implementation offers support only for FreeRTOS and bare-metal.

PORT

- PinSettings component will not issue warnings when pins routed by default are overwritten. This may impact the debug functionality when JTAG or SWD pins are silently rerouted when other functionality is selected for the shared pins.

RTC

- RTC_HAL_Init will return error only if the registers are locked and not if the Time Invalid Flag is set. The last case cannot be reached as the flow of the function will not trigger the flag.

SBC

- Parameter naming are not correct in PEx method description for SBC_GetWatchdogStatus and SBC_SetWakePin functions

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.

9. Compiler options

This release was developed and tested with:

- GreenHills compiler v. 2015.1.4
- IAR compiler v. 7.50.3
- GCC compiler for ARM v. 4.9.3 20150529
- COSMIC Software CORTEX-M C Cross Compiler v4.3.10
- Wind River Diab Compiler v5.9.4.8

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	Selects target processor: Arm Cortex M4
--thumb	Selects generating code that executes in Thumb state.
--fpu VFPv4_sp	Use floating point instructions
--debug	Include debug information
-DCPU_S32K144HFT0VLLT	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	Selects target processor: Arm Cortex M4
--thumb	Selects generating code that executes in Thumb state.
--fpu VFPv4_sp	Use floating point instructions

Table 8.3 IAR Linker Options

Option	Description
--cpu Cortex-M4	Selects target processor: Arm Cortex M4
--thumb	Selects generating code that executes in Thumb state.
--fpu VFPv4_sp	Use floating point instructions
--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	Selects target processor: Arm Cortex M4
-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
-DCPU_S32K144HFT0VLLT	Define a preprocessor symbol for MCU
-mfloat-abi=hard	Use FPU instructions
-mfpu=fpv4-sp-d16	Specify the FPU variant
-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	Selects target processor: Arm Cortex M4
-mthumb	Selects generating code that executes in Thumb state.
-mfloat-abi=hard	Use FPU instructions
-mfpu=fpv4-sp-d16	Specify the FPU variant
-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

Table 8.6 GCC Linker Options

Option	Description
-mcpu=cortex-m4	Selects target processor: Arm Cortex M4
-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	Use FPU instructions
-mfpu=fpv4-sp-d16	Specify the FPU variant
-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=cortexm4f	Selects target processor: Arm Cortex M4
-thumb	Selects generating code that executes in Thumb state.
-fhard	Use FPU instructions
-fpu=vfpv4_d16	Specify FPU type
-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
-DCPU_S32K144HFT0VLLT	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=cortexm4	Selects target processor: Arm Cortex M4
-fhard	Use FPU instructions
-fpu=vfpv4_d16	Specify FPU type

Table 8.9 GHS Linker Options

Option	Description
-cpu=cortexm4f	Selects target processor: Arm Cortex M4
-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 COSMIC Compiler/Linker/Assembler Options

Table 8.10 COSMIC Compiler Options

Option	Description
-gsf	Place each function into its own section in the output file
-pc99	Use C99 standard
-dCPU_S32K144HFT0VLLT	Define a preprocessor symbol for MCU
+modlc	Use memory model with unlimited data and constants size
+fpu	Use FPU
+debug	Add debug information to the executable
-pp	Enforce function prototype checking
+strict	Enforce stronger type checking

Table 8.11 COSMIC Assembler Options

Option	Description
Not needed	

Table 8.12 COSMIC Linker Options

Option	Description
<linker_file.lkf>	Use the specified linker file
-m <map_file>	Produce a linker map
-e Reset_Handler	Make the symbol Reset_Handler be treated as a root symbol and the start label of the application

8.5 DIAB Compiler/Linker/Assembler Options

Table 8.13 DIAB Compiler Options

Option	Description
-tARMCORTEXM4LV	Selects target processor: Arm Cortex M4
-mthumb	Selects generating code that executes in Thumb state.
-Xdialect-c99	Use C99 standard
-DCPU_S32K144HFT0VLLT	Define a preprocessor symbol for MCU
-g	Add debug information to the executable
-O	Optimize
-Xstop-on-warning	Treat warnings as erros

Table 8.14 DIAB Assembler Options

Option	Description
-tARMCORTEXM4LV	Selects target processor: Arm Cortex M4
-mthumb	Selects generating code that executes in Thumb state.

Table 8.15 DIAB Linker Options

Option	Description
-tARMCORTEXM4LV	Selects target processor: Arm Cortex M4
-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

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 replace with the desired map file name.

10. Acronyms

Acronym	Description
EAR	Early Access Release
JRE	Java Runtime Environment
EVB	Evaluation board
HAL	Hardware Abstraction Layer
RTOS	Real Time Operating System
PEx	Processor Expert Configurator
PD	Peripheral Driver
S32DS	S32 Design Studio IDE
SDK	Software Development Kit
SOC	System-on-Chip

11. 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
1-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-17	5.0	Updated Release Notes for 1.0.0 RTM release	Rares Vasile