S32 SOFTWARE DEVELOPMENT KIT
APPLICATION DEVELOPMENT SOFTWARE
© 2016 Freescale Semiconductor, Inc.

S32 SDK Release Notes

Version 0.9.0 BETA

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 BETA 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 FlexIO, Power Manager
- Moved examples to examples/S32K144 folder
- Added derivative name prefix to the examples

2.2 Drivers:

- **All drivers**
  - Implemented common error codes, updated copyright and branding.
  - Changes above noted are breaking the backwards compatibility of existing projects.
- **CPU**
  - Renamed linker files to reflect derivative memory size.
- **LPUART**
  - Added support for 9 bits per character transfer
- **eDMA**
  - Removed dynamic allocation for channels; applications must ensure the channels are correctly allocated and pass channel numbers to other drivers using these resources. Added checking in configuration component for erroneous channel allocation.
- **CSEc**
  - Added asynchronous API for encryption, decryption, MAC generation and MAC verification

- **Clock**
  - Improved CLOCK_SYS_GetFrequency API.
- **FlexCAN**
  - Added Pretending Networking support
  - Added option to configure frames as remote frames or data frames
- **LPI2C**
  - Removed High-Speed and Ultra-Fast modes, since the pins do not implement the current source pull-up required by the higher baudrate specifications of I2C standard. Without this, High-speed mode can’t achieve higher baud rates than standard/fast mode
- **LPIT**
  - Modified driver configuration structure and made changes to enums
  - Added support for setting timer period in Dual 16 mode in microseconds and counter ticks.
- **LPTMR**
  - Merged Timer and Pulse Counter configurations and adjusted API functions accordingly.
  - Updated PEx component to reflect those changes.
- **PINS**
  - Added port gating off check
- **LINSTACK**
  - Added support for slave node serial number
  - Improved timing for signal transmission and reception
  - Improved ld_receive_message to be non-blocking

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. 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 Hardware Trigger	Uses PDB to trigger an ADC conversion with a configured delay and sends the result to host via LPUART.
	ADC Software Trigger	Uses software trigger to periodically trigger an ADC conversion and sends the result to host via LPUART.
	CMP	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	The CRC is configured to generate the cyclic redundancy check value using 16 and 32 bits wide result.
	CSEc Key Configuration	The example demonstrates how to prepare the MCU before using CSEc(Key configuration, flash partitioning).
	eDMA	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	Shows the usage of the EWM driver.
	FLASH	Writes, verifies and erases data on Flash.
	FlexIO I2C Example	Demonstrates FlexIO I2C emulation. Use one instance of FlexIO and one instance of LPI2C to transfer data on the same board.
	FlexIO SPI Example	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 Example	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 Example	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 Timer	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 Example	Shows the usage of the LPI2C driver in Master configuration
	LPI2C Slave Example	Shows the usage of the LPI2C driver in Slave configuration
	LPIT	Shows how to initialize the LPIT to generate an interrupt every 1 s. It is the starting point for any application using LPIT.
	LPSPI	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 Timer	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	Simple example of a basic echo using LPUART.
	MPU	Configures MPU to protect a memory area and demonstrates that read access is correctly restricted.
	PDB	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 Manager	Demonstrates the usage of Power Manager by allowing the user to switch to all power modes available.
	RTC	Show the frequently used RTC use cases such as the generation of an interrupt every second and triggering an alarm.
	WDOG	Shows the basic usage scenario and configuration for the Watchdog.
Demos	Blinking LED	This is a simple application created to show configuration for supported compilers (GCC, IAR, GHS …).
	Clock setup	Start-up project with maximal clock settings for the platform.
	FlexCAN	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	This demo application shows the usage of LIN stack. There are slave and master applications, both configuration having baremetal and FreeRTOS variants.

	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.
	FreeMASTER BDM	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 BDM for communication.
	FreeMASTER SerialCommunication	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. Supported hardware and compatible software

CPUs

- S32K144_64 revision 2.0, maskset N47T
- S32K144_100 revision 2.0, maskset N47T
- S32K144_100_BGA revision 2.0, maskset N47T

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

- S32K14XRM Rev. 1, 08/2016

Boards

- S32K144-MB with mini module S32K144-100LQFP REV X1/X2
- S32K144-EVB_Q100

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 v4.3.9
- 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)

7. 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.

2. Drivers

ADC

- The trigger latch feature of the ADC is not supported in this release.

eDMA

- Doxygen documentation is not accessible from context menu for eDMA and DMAMUX HAL components.

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 that what type of error get generated

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.

LPSPI

- Continuous mode is not supported.

- LPSPI Cannot transmit data between master and slave onboard

LPUART

- 9th bit is not sent correctly when configured with 10 bits per char
- LPUART_HAL_SetTxSpecialChar function queues only break characters for transmission; idle characters cannot be queued using this function.

OSIF

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

PORT

- Default state of several pins is not correct
- PinSettings component will not issue warnings when pins routed by default are overwritten

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.

8. 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 v4.3.9
- Wind River Diab Compiler v5.9.4.8

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

8.1 IAR Compiler/Linker/Assembler Options

Table 8.1 IAR Compiler Options

Option	Description
-On	No optimization (best for debug)
-e	Allow IAR extensions
--cpu=Cortex-M4	Selects target processor: Arm Cortex M4
--thumb	Selects generating code that executes in Thumb state.
--no_wrap_diagnostics	Disable line wrapping of diagnostic messages
--debug	Include debug information
-DCPU_S32K144HFT0VLLT	Define a preprocessor symbol for MCU

Table 8.2 IAR Assembler Options

Option	Description
Not needed	

Table 8.3 IAR Linker Options

Option	Description
--map main.map	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
--no_wrap_diagnostics	Keep diagnostic messages on single line
--config S32K144_100_ram.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.
-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
-Os	Optimize for size
-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=gnu99	Use C99 standard
-save-temps=obj	Store the usual “temporary” intermediate files permanently
-g	Generate debug information
-DCPU_S32K144HFT0VLLT	Define a preprocessor symbol for MCU

Table 8.5 GCC Assembler Options

Option	Description
Not needed	

Table 8.6 GCC Linker Options

Option	Description
--specs=rdimon.specs	Select library librdimon.a for system calls.
-lgcc	Link libgcc.a
-lc	Link libc.a
-lm	Link libm.a
-lrdimon	Link librdimon.a
-Wl,-Map=main.map	Produce a linker map

8.3 GHS Compiler/Linker/Assembler Options

Table 8.7 GHS Compiler Options

Option	Description
-cpu=cortexm4	Selects target processor: Arm Cortex M4
-mthumb	Selects generating code that executes in Thumb state.
-C99	Use C99 standard
--gnu_asm	Enables GNU extended asm syntax support
-O0	No optimization (best for debug)
-gdwarf-2	Generate DWARF 2.0 debug information
-G	Generate debug information
-DCPU_S32K144HFT0VLLT	Define a preprocessor symbol for MCU

Table 8.8 GHS Assembler Options

Option	Description
Not needed	

Table 8.9 GHS Linker Options

Option	Description
Not needed	

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

Table 8.11 COSMIC Assembler Options

Option	Description
Not needed	

Table 8.12 COSMIC Linker Options

Option	Description
-m app.cxm.map	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

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 math operations that are used by the drivers
-e Reset_Handler	Make the symbol Reset_Handler be treated as a root symbol and the start label of the application
-m6 > app.map	Produce a linker map

9. Acronyms

Acronym	Description
EAR	Early Access Release
JRE	Java Runtime Environment
FRDM	Freedom board
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

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
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