S32 SDK Release Notes

Version 0.8.6 EAR

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 EAR quality level in terms of features, testing and quality documentation, according to NXP software release criteria.

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

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/community/s32/s32k

2. New in this release

2.1 Examples

- Added CAN PAL, SECURITY PAL, MPU PAL, I2C PAL, OC PAL and LWIP examples

2.2 Drivers

- **CPU**
  - Removed COSMIC support
- **PAL**
  - Added CAN, I2C, I2S, IC, MPU, OC, PWM, SECURITY
- **WDOG**
  - Added WDOG_DRV_ClearIntFlag() function.
- **WDG PAL**
  - Add WDG_ClearIntFlag() function support over WDOG module

2.3 Middleware

Touch sense and ISELED libraries are available. Please contact with Sales representative or FAE for more information how to get it.

2.4 Libraries

- Updated sCST to 1.0.3 RTM

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 (ClockManager, InterruptManager, PowerManager)
- MPU
- PINS
- PDB
- PHY
- 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
- SBC - provides support for UJA1169 and UJA113x System Basis Chips
- TCP/IP

3.4 RTOS

- FreeRTOS version 8.2.1

3.5 Libraries

- AMMCLib version 1.1.8
- sCST version 1.0.3

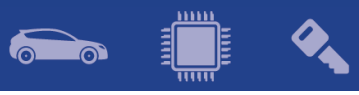
4. Documentation

- 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.
	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.
	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_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.
	i2c_pal	Shows the usage of I2C PAL driver in both master and slave configurations using FLEXIO and LPI2C
	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.
	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.
	sai_transfer	Demonstrates the usage of the SAI module driver
	sbc_uja1169	Show the usage of the SBC 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.
	rtc_alarm	Show the frequently used RTC use cases such as the generation of an interrupt every second and triggering an alarm.
	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.
	wdog_interrupt	Shows the basic usage scenario and configuration for the Watchdog.
	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.
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.
	scst	Demo application created to demonstrate sCST integration with S32 SDK.
	touchsense	Demo application created to demonstrate touch sensing capabilities of the S32K devices
	lwip	Shows the usage of lwIP stack.

6. Supported hardware and compatible software

6.1 CPUs

- S32K142_64 revision 1.0, maskset 0N33V
- S32K142_100 revision 1.0, maskset 0N33V
- S32K144_64 revision 2.1, maskset 0N57U
- S32K144_100 revision 2.1, maskset 0N57U
- S32K144_100_BGA revision 2.1, maskset 0N57U
- S32K146_100 revision 1.0, maskset 0N73V
- S32K146_100_BGA revision 1.0, maskset 0N73V
- S32K146_144 revision 1.0, maskset 0N73V
- 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. 4, 06/2017

6.2 Boards

- S32K-MB with mini module S32K144-100LQFP REV X1/X2
- S32K-MB with mini module S32K14xCVD-Q144 REV X3
- S32K144-EVB-Q100 REV X2
- S32K148-EVB-Q144
- S32K142-EVB-Q100

6.3 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
- Wind River Diab Compiler v5.9.4.8
- S32 Design Studio v2.0 IDE

6.4 Debug Probes

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

7. Known issues and limitations

7.1 Debugger support

- Debugging in flash with J-Link for S32K142 is not working correctly because of debugger support and can brick the board making it unusable.
  DO NOT USE J-LINK flash debug on the S32K142 target.

7.2 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.

7.3 Drivers

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.

CLOCK

- CLOCK_SYS_GetFreq function returns obsolete core clock frequency right after VLPR to HSRUN power mode transition (workaround: function to be called twice, second call returns correct value).

EIM

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

FLASH

- If the application 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.

FLEXCAN

- FLEXCAN_DRV_AbortTransfer does not abort the transfer for the selected message buffer
- Remote requests are ignored by the driver

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

FREERTOS

- PEx component does not open function definition when the function is double-clicked in the method list

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

- LPSPI does not work correctly in half duplex mode. The workaround is to use a dummy buffer for Tx/Rx.

LPUART

- Framing error not detected by the driver

MPU

- MPU does not cover QSPI master because QSPI wrongly connected to MPU slave port.

OSIF

- OSIF no longer generates configuration files (osif1.c and osif1.h), but the automatically included osif1.h will not be removed by PEx. All projects migrated to EAR 0.8.6 will need to manually remove the '#include "osif1.h"’ directive from main.c
- osif.h not included in CPU.h when PEx component is present in project

PINS

- 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.

7.4 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 projects for S32K148, S32K142 and S32K146 cannot be debugged on IAR IDE, since the IDE version currently supported by the SDK does not support S32K derivatives other than S32K144.

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	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
-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	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
-D<cpu_define>	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
-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=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 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
-D<cpu_define>	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 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
AMMCLib	Automotive Math and Motor Control Library
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