StellarESDK Version 1.6.0

[Tag: SESDK_1.6.0]

What's new

- Added support for EVBE3000D
- Added PMU low level driver
- Added LIN low level driver
- Enhanced TIM module adding support for channels 5 and 6
- Enhanced TIM module adding support for PWM Active Low mode
- Enhanced TIM module adding support for center-aligned mode
- Enhanced TIM module adding setup for the maximum deadtime
- Enhanced TIM module adding break event management
- Enhanced TIM module adding support for new output compare modes
- Enhanced TIM module adding new APIs to get DIR and CNT values
- Enhanced TIM module adding setting for the input 1 selection
- Enhanced TIM module adding new API for COM generation
- Enhanced TIM module adding COM event callback management
- Enhanced TIM module adding Capture/Compare preload control management
- Enhanced TIM module adding Quadrature Encoder management
- Updated HRTIM module adding fault blanking enabling
- Updated CAN module fixing clock initialization
- Enhanced EED module adding power loss management
- Enhanced EED module adding support for all Data Flash blocks
- Updated FLASH module fixing issues when compiled in release mode
- Updated SARADC module to correctly manage dual mode
- Updated SARADC module to correctly handle master instances
- Updated SARADC module fixing DIFSEL data type
- Updated DAC module to avoid glitches during waveform generation
- Added console for FreeRTOS Command Line Interface
- Updated FreeRTOS module adding Command Line Interface support
- Fixed Runtime IO to avoid IO operations are buffered when Hightec compiler is used
- Added IMA header file
- Updated SDK examples removing switch-off on unused leds
- Added LIN example
- Added PMU examples
- Added new TIM example for break event management
- Updated CRC examples using predefined CRC32 configuration
- Updated WWDG example to make it compatible for the execution from core 2
- Updated SARADC tests fixing channel indexes printed on serial port during test execution
- Updated StellarESDK Quick Start Guide

Known Limitations

- CMSIS support from GHS available on request.
- I2S configuration Master RX - Slave TX not fully tested.
- FLASH driver is fully compatible and fully tested with GCC, Hightec and GHS compilers.
- EVBE3000E not tested.

Modules version table

Boot	VER_1.2.0	
BuildSystem	VER_1.6.0	new version
COMP	VER_1.2.1	
DAC	VER_2.1.1	new version
SARADC	VER_2.1.0	new version
SDADC	VER_2.0.0	
TSENS	VER_1.0.0	
CAN	VER_1.3.1	new version
I2C	VER_1.1.0	
I2S	VER_1.3.0	
LIN	VER_1.0.0	new
SENT	VER_1.2.0	
SPI	VER_1.4.0	
UART	VER_1.3.0	
ILI9341	VER_1.2.0	
EED	VER_2.0.0	new version
FLASH	VER_1.4.1	new version
CEM	VER_2.0.0	
CMU	VER_1.0.0	
FCCU	VER_1.2.1	
CORDIC	VER_1.3.0	
CRC	VER_1.1.1	new version
DMA	VER_1.3.0	
EXTI	VER_1.2.0	
GPIO	VER_1.3.0	
HSEM	VER_1.2.0	
IWDG	VER_1.2.0	
MPU	VER_1.3.0	
PMU	VER_1.0.0	new
SMPU	VER_1.0.1	
WWDG	VER_1.2.0	
DWT	VER_1.2.0	
HRTIM	VER_3.0.0	new version
RTC	VER_1.0.0	
SYSTICK	VER_1.3.0	
TIM	VER_1.6.0	new version
TIM_TS	VER_1.0.0	
FreeRTOS(*)	VER_1.1.0	new version
OSAL	VER_1.5.0	
Console	VER_1.0.0	new
Board	VER_1.5.0	new version
Clock	VER_1.5.0	
CMSIS(**)	VER_0.9.0	
IRQ	VER_1.2.0	
MCU	VER_1.6.0	new version
IO	VER_1.2.1	new version
SDKTests	VER_2.2.0	new version

(*) FreeRTOS VER_1.1.0 is based on FreeRTOS v10.4.6 (www.freertos.org)
(**) CMSIS VER_0.9.0 is based on CMSIS 5.8.0 (www.arm.com/technologies/cmsis)

StellarE SDK Tests

Path	Test name	Tested on

Analog/COMP	COMP_01	evbe7000p, evbe3000p, evbe7000e
Analog/DAC	DAC_01	evbe7000p, evbe3000p, evbe7000e
Analog/SARADC	SARADC_01	evbe7000p, evbe3000p, evbe7000e
Analog/SARADC	SARADC_02	evbe7000p, evbe3000p, evbe7000e
Analog/SARADC	SARADC_03	evbe7000p, evbe7000e
Analog/SARADC	SARADC_04	evbe7000p, evbe7000e
Analog/SARADC	SARADC_05	evbe7000p, evbe3000p
Analog/SARADC	SARADC_06	evbe7000p, evbe3000p, evbe7000e
Analog/SARADC	SARADC_07	evbe7000p, evbe3000p, evbe7000e
Analog/SDADC	SDADC_01	evbe7000p, evbe7000e
Analog/SDADC	SDADC_02	evbe7000p, evbe7000e
Analog/TSENS	TSENS_01	evbe7000p, evbe3000p, evbe7000e
Benchmarks/COREMARK	COREMARK_01	evbe7000p, evbe3000p, evbe7000e
Comms/CAN	CAN_01	evbe7000p, evbe3000p, evbe7000e
Comms/CAN	CAN_02	evbe7000p, evbe3000p, evbe7000e
Comms/CAN	CAN_03	evbe7000p, evbe3000p, evbe7000e
Comms/I2C	I2C_01	evbe7000p
Comms/I2C	I2C_02	evbe7000p
Comms/I2S	I2S_01	evbe7000p, evbe3000p, evbe7000e
Comms/LIN	LIN_01	evbe7000p
Comms/SENT	SENT_01	evbe7000p, evbe3000p, evbe7000e
Comms/SPI	SPI_01	evbe7000p, evbe3000p, evbe7000e
Comms/SPI	SPI_02	evbe7000p, evbe3000p, evbe7000e
Comms/SPI	SPI_03	evbe7000p
Comms/SPI	SPI_04	evbe7000p, evbe3000p, evbe7000e
Comms/SPI	SPI_05	evbe7000p, evbe3000p, evbe7000e
Comms/SPI	SPI_06	evbe7000p, evbe3000p, evbe7000e
Comms/SPI	SPI_07	evbe7000p, evbe3000p, evbe7000e
Comms/SPI	SPI_08	evbe7000p, evbe3000p, evbe7000e
Comms/SPI	SPI_09	evbe7000p, evbe3000p, evbe7000e
Comms/UART	UART_01	evbe7000p, evbe3000p, evbe7000e, evbe3000d
Graphics/ILI9341	ILI9341_01	evbe7000p, evbe3000p, evbe7000e
Memories/EED	EED_01	evbe7000p, evbe3000p, evbe7000e
Memories/FLASH	FLASH_01	evbe7000p, evbe3000p, evbe7000e
Miscellaneous	MISC_01	evbe7000p, evbe3000p, evbe7000e
MultiCore	MULTICORE_01	evbe7000p, evbe3000p, evbe7000e
MultiCore	MULTICORE_02	evbe7000p, evbe3000p, evbe7000e
MultiCore	MULTICORE_03	evbe7000p, evbe3000p, evbe7000e
OS/FREERTOS	FREERTOS_01	evbe7000p, evbe3000p, evbe7000e
Safety/CEM	CEM_01	evbe7000p, evbe3000p, evbe7000e
Safety/CMU	CMU_01	evbe7000p, evbe3000p, evbe7000e
Safety/CMU	CMU_02	evbe7000p, evbe3000p, evbe7000e
Safety/CMU	CMU_03	evbe7000p, evbe3000p, evbe7000e
Safety/FCCU	FCCU_01	evbe7000p, evbe3000p, evbe7000e
System/CORDIC	CORDIC_01	evbe7000p, evbe3000p, evbe7000e
System/CRC	CRC_01	evbe7000p, evbe3000p, evbe7000e
System/CRC	CRC_02	evbe7000p, evbe3000p, evbe7000e
System/DMA	DMA_01	evbe7000p, evbe3000p, evbe7000e
System/DMA	DMA_02	evbe7000p, evbe3000p, evbe7000e
System/DMA	DMA_03	evbe7000p, evbe3000p, evbe7000e
System/EXTI	EXTI_01	evbe7000p, evbe3000p, evbe7000e
System/GPIO	GPIO_01	evbe7000p, evbe3000p, evbe7000e, evbe3000d
System/HSEM	HSEM_01	evbe7000p, evbe3000p, evbe7000e
System/IWDG	IWDG_01	evbe7000p, evbe3000p, evbe7000e
System/MPU	MPU_01	evbe7000p, evbe3000p, evbe7000e
System/MPU	MPU_02	evbe7000p, evbe3000p, evbe7000e
System/PMU	PMU_01	evbe7000p
System/PMU	PMU_02	evbe7000p
System/SMPU	SMPU_01	evbe7000p, evbe3000p, evbe7000e
System/WWDG	WWDG_01	evbe7000p, evbe3000p, evbe7000e
Timers/HRTIM	HRTIM_01	evbe7000p, evbe3000p, evbe7000e
Timers/RTC	RTC_01	evbe7000p, evbe3000p, evbe7000e
Timers/TIM	TIM_01	evbe7000p, evbe3000p, evbe7000e
Timers/TIM	TIM_02	evbe7000p, evbe3000p, evbe7000e
Timers/TIM	TIM_03	evbe7000p, evbe3000p, evbe7000e
Timers/TIM	TIM_04	evbe7000p
Timers/TIM_TS	TIM_TS_01	evbe7000p, evbe3000p, evbe7000e

StellarE SDK Tests Description

COMP_01

COMP test.

Configures a comparator and compares the voltage on input plus with the voltage on input minus.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

DAC_01

DAC test.

Configures DAC to generate 2 signals with a variable amplitude.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

SARADC_01

SARADC test: regular conversion in continuos mode.

Executes an ADC regular conversion in continuous mode and prints the result on serial [UART] terminal.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

SARADC_02

SARADC test: regular conversion in single mode with trigger.

Executes an ADC regular conversion in single mode triggered by TIM and prints the results on serial [UART] terminal.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

SARADC_03

SARADC test: regular conversion in single mode + injected conversion with trigger.

Executes an ADC regular conversion in single mode and an ADC injected conversion triggered by TIM and prints the results on serial [UART] terminal.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

SARADC_04

SARADC test: regular conversion in continuos mode + injected conversion.

Executes an ADC regular conversion in continuous mode and an ADC injected conversion and prints the results of injected conversions on serial [UART] terminal.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

SARADC_05

SARADC test: analog watchdog thresholds.

Shows how to configure the thresholds to trigger the SARADC watchdogs.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

SARADC_06

SARADC test: dual mode.

Shows how to configure and run regular conversions in dual mode.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

SARADC_07

SARADC self test.

Shows how to configure and run the SARADC self test.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

SDADC_01

SDADC test.

Runs an SDADC conversion in single ended input mode and prints the result on serial [UART] terminal.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

SDADC_02

SDADC timestamp test.

Shows to configure and use the timestamp mode based on TIM_TS module.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

TSENS_01

TSENS test.

Shows how to use the TSENS to get the micro temperature.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

COREMARK_01

CoreMark single core test.

Executes CoreMark benchmark in single core and prints the results on serial [UART] terminal.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

CAN_01

CAN test.

Transmits CAN standard frames from CAN TX to CAN RX.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

CAN_02

CAN DMU test.

Transmits CAN standard frames from CAN TX to CAN RX via DMU.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

CAN_03

CAN FD frame test.

Transmits CAN FD frames from CAN TX to CAN RX.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

I2C_01

I2C test: Master TX - Slave RX, Master RX - Slave TX, interrupt mode.

Transmits data from I2C master to I2C slave and from I2C slave to I2C master in interrupt mode.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

I2C_02

I2C test: Master TX - Slave RX, Master RX - Slave TX, DMA mode.

Transmits data from I2C master to I2C slave and from I2C slave to I2C master in DMA mode.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

I2S_01

I2S test: Master TX - Slave RX.

Transmits data from I2S master to I2S slave.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

LIN_01

LIN test: Master TX - Slave RX, Master RX - Slave TX

Transmits data from LIN master to LIN slave and from LIN slave to LIN master.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

SENT_01

SENT test.

Shows how to configure and use the SENT to receive a frame.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

SPI_01

SPI test: Master TX - Slave RX, DMA mode, full duplex.

Transmits data from SPI master to SPI slave in DMA mode (full duplex).

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

SPI_02

SPI test: Master RX - Slave TX, DMA mode, full duplex.

Transmits data from SPI slave to SPI master in DMA mode (full duplex).

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

SPI_03

SPI multi-slave test (full duplex).

Shows how to implement a SPI multi-slave communication transmitting same data from a SPI master to 2 SPI slaves.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

SPI_04

SPI test: Master TX - Slave RX, interrupt mode, full duplex.

Transmits data from SPI master to SPI slave in interrupt mode (full duplex).

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

SPI_05

SPI test: Master RX - Slave TX, interrupt mode, full duplex.

Transmits data from SPI slave to SPI master in interrupt mode (full duplex).

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

SPI_06

SPI test: Master TX - Slave RX, DMA mode, half duplex.

Transmits data from SPI master to SPI slave in DMA mode (half duplex).

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

SPI_07

SPI test: Master RX - Slave TX, DMA mode, half duplex.

Transmits data from SPI slave to SPI master in DMA mode (half duplex).

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

SPI_08

SPI test: Master TX - Slave RX, interrupt mode, half duplex.

Transmits data from SPI master to SPI slave in interrupt mode (half duplex).

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

SPI_09

SPI test: Master RX - Slave TX, interrupt mode, half duplex.

Transmits data from SPI slave to SPI master in interrupt mode (half duplex).

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

UART_01

UART test.

Prints 'Hello World!!!' string on serial [UART] terminal.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

ILI9341_01

ILI9341 test.

Configures and uses a display TFT ILI9341 via SPI driver.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

EED_01

EED test.

Shows how to configure and use the EPROM Emulation driver to perform the basic operations (read/write/delete) on the emulated EEPROM.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

FLASH_01

FLASH test.

Modifies the content of the FLASH device and checks it.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

MISC_01

CLKOUT test.

Configures the Microcontroller Clock Output (MCO) to output PLL1 clock.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

MULTICORE_01

Dual core test.

Shows how to run from core1 the core2.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

MULTICORE_02

Dual core test (single elf).

Shows how to run from core 1 the core 2. Both applications for core1 and core2 are encapsuled in a single elf file.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

MULTICORE_03

SEV test (single elf).

Shows how to generate an interrupt on core1 using the Send Event instruction from core2.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

FREERTOS_01

FreeRTOS test.

Shows how to create FreeRTOS tasks and execute them.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

CEM_01

CEM test.

Configures the CEM in conjunction with the FCCU module for the fault management.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

CMU_01

CMU test (SYSCLK check).

Shows how to configure and use the CMU1 to use the clock supervisor.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

CMU_02

CMU test (SARADC clock check).

Shows how to configure and use the CMU4 to use the clock supervisor on SARADC.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

CMU_03

CMU test (IRCOSC clock metering).

Shows how to configure and use the CMU0 to use the frequency meter.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

FCCU_01

FCCU test.

Configures the FCCU to react to some specific faults.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

CORDIC_01

CORDIC test.

Configures the CORDIC to calculate the sine and cosine.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

CRC_01

CRC test.

Calculates the CRC related to a buffer of elements both processing the buffer in one shot and processing the buffer sequentially in 2 steps.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

CRC_02

CRC DMA mode test.

Calculates the CRC related to a buffer of elements both processing the buffer in one shot and processing the buffer sequentially in 2 steps in DMA mode.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

DMA_01

DMA test: RAM to DTCM transfer.

Shows how to configure the DMA to transfer a buffer from RAM to DTCM (memory to memory) using the DTCM indirect address.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

DMA_02

DMA test: RAM to RAM transfer with DCache enabled (cache invalidate).

Shows how to configure the DMA to transfer a buffer from RAM to RAM (memory to memory) when the DCache is enabled.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

DMA_03

DMA test: RAM to RAM transfer with DCache enabled (MPU setting).

Shows how to configure the DMA to transfer a buffer from RAM to RAM (memory to memory) when the DCache is enabled and the MPU is used to mark the region containing the destination buffer as not cachable.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

EXTI_01

EXTI test.

Configures a pin as external interrupt and generates an interrupt on both edges of a PWM signal generated by a TIM connected to the external interrupt pin.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

GPIO_01

LED test.

Blinks a LED.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

HSEM_01

HSEM test.

Shows how to configure and use the hardware semaphores for multi-core synchronization.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

IWDG_01

IWDG test.

Shows how to configure and use the Indipendent Watchdog.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

MPU_01

MPU test.

Configures MPU and shows how a Memory Management Unit fault is raised when the core tries to access a region protected by MPU.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

MPU_02

MPU test.

Configures MPU and shows how a Memory Management Unit fault is raised when the core in unprivileged mode tries to access a region configured by MPU as accessible only in privileged mode.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

PMU_01

PMU test.

Shows how to configure and use the Power Management Unit.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

PMU_02

PMU test with FCCU.

Shows how to configure and use the Power Management Unit with FCCU.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

SMPU_01

SMPU test.

Configures SMPU and shows how an Hard Fault is raised when the core1 master tries to access a region that is write protected.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

WWDG_01

WWDG test.

Shows how to configure and use the System Window Watchdog.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

HRTIM_01

HRTIM test.

Generates 2 complementary PWM signals with a specified frequency, duty cycle and deadtime.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

RTC_01

RTC test.

Shows how to configure and use the Real Time Clock peripheral.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

TIM_01

TIM test.

Generates on a TIM channel a PWM signal and measures its frequency using another TIM channel configured as input capture.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

TIM_02

TIM test: duty and frequency measurement.

Show how to configure the TIM to measure duty cycle and frequency of a PWM signal.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

TIM_03

TIM DMA mode test.

Generates on a TIM channel a PWM signal and measures its frequency using another TIM channel configured as input capture using the DMA.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

TIM_04

TIM break event test.

Shows how to configure a break event that stops the PWM waveform generation.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

TIM_TS_01

TIM_TS test.

Shows how to configure the TIM_TS to generate an interrupt with a frequency of 1Hz.

Please, check the readme file in the test folder for more details.

Build command:

```
$ make all
```

Detailed notes

BuildSystem [VER_1.6.0]

Added support for EVBE3000D.
Updated build system to support new module LIN.
Updated build system to support new module PMU.
Updated build system to support console for FreeRTOS Command Line Interface.

DAC [VER_2.1.1]

Fixed DAC settings to avoid glitches during waveform generation.
Moved clock enabling from init API to start API aligning DAC with all the other SDK modules.

SARADC [VER_2.1.0]

Fixed dual mode management.
Fixed DIFSEL data type.

CAN [VER_1.3.1]

Fixed clock enabling fitting the hardaware requirement that forces to enable all FDCAN instances before to enable the FDCAN Message RAM.

LIN [VER_1.0.0]

Added LIN driver.

EED [VER_2.0.0]

Added support to use all blocks available in Data Flash.
Added power loss management.

FLASH [VER_1.4.1]

Fixed issues in program and erase operation when the module is compiled in release mode.

CRC [VER_1.1.1]

Fixed typos.

PMU [VER_1.0.0]

Added PMU driver.

HRTIM [VER_3.0.0]

Added fault blanking enabling.
Fixed typos.

TIM [VER_1.6.0]

Added new API for COM event generation.
Added COM event callback management.
Added capture/compare control management.
Added new API for the input 1 selection setting.
Added new APIs to get DIR and CNT values.
Added Quadrature Encoder management.
Added support for the following output compare modes: Frozen, Active level on match, Inactive level on match, Combined PWM mode 1, Combined PWM mode 2, Asymmetric PWM mode 1 and Asymmetric PWM mode 2.
Added maximum deadtime management.
Added support for center-aligned mode.
Added break event management.
Added support for internal channels 5 and 6.
Added support for PWM Active Low mode.

FreeRTOS [VER_1.1.0]

Added support for Command Line Interface.

Console [VER_1.0.0]

Added console for FreeRTOS Command Line Interface.

Board [VER_1.5.0]

Added support for EVBE3000D.

MCU [VER_1.6.0]

Added interrupt vector and interrupt handler for PMU module.
Added IMA header file.

IO [VER_1.2.1]

Fixed IO module to avoid IO operations are buffered when Hightec compiler is used.

SDKTests [VER_2.2.0]

Added support for EVBE3000D.
Updated SDK examples making OSAL enabled by default.
Updated SDK examples removing switch-off for unused leds.
Added new example for LIN module.
Added two new examples for PMU module.
Added new example for TIM module related to break event management.
Updated SARADC examples fixing channel indexes printed on serial port during test execution.
Updated UART and GPIO examples to make them compatible with EVBE3000D.
Updated SARADC example related to dual mode.
Updated CRC examples using predefined CRC32 configuration.
Updated WWDG example making it compatible for the execution from core 2.

Added Comms/LIN/LIN_01.
Added System/PMU/PMU_01.
Added System/PMU/PMU_02.
Added Timers/TIM/TIM_04.