StellarESDK Version 1.5.0

[Tag: SESDK_1.5.0]

What's new

- Added support for EVBE3000P
- Added support for EVBE7000E and EVBE3000E
- New TIM_TS driver
- Updated DMA setting in DAC, SARADC, CAN, I2C, I2S, SPI, UART, CORDIC, CRC and TIM modules
- Enhanced SDADC module adding capture timestamp request management
- Enhanced SDADC module adding trigger support
- Enhanced SARADC module adding support for thresholds management
- Enhanced SARADC module adding support for dual mode
- Updated FLASH module providing FLASH driver source code
- Enhanced TIM module adding new API for DMA triggering
- Updated TIM module adding new API to return the TIM clock frequency
- Updated TIM module renaming trigger macros
- Fixed TIM module removing register pointer deinitilization from the API tim_stop
- Enhanced SENT module to manage different nibbles lengths
- Updated SENT module adding new API for setting the frequency of the TIM used for SENT emulation
- Fixed endianess issue in SENT module
- Fixed multi-channel issue in SENT module
- Updated SYSTICK and OSAL modules adding the support for timeout management
- Updated UART module adding timeout management
- Updated SPI module adding timeout management
- Updated Clock module adding the check of the frequency range after the prescaler
- Updated GPIO module to garantee the settings of pin parameters with a single register access
- Updated MPU module adding new API for region clearing/disabling
- Updated MPU module making region configurations updatable
- Updated API related to the GROUP_WRITE_ ENABLE command in CEM module
- Fixed FIFO size setting in SDADC module
- Fixed bus off management in CAN module
- Fixed ADC post-scaler setting in HRTIM module
- Fixed ADC trigger setting in HRTIM module
- Fixed blanking and windowing macros doxygen description in HRTIM module
- Fixed single mode value setting in DAC module
- Updated makefiles of SDK examples adding compiling target management
- Added SARADC self test example
- Added SARADC example with thresholds management
- Added SDADC example with timestamp mode
- Added DMA examples
- Added TIM_TS example
- Added new MPU example related to core privileged/unprivileged mode
- Updated ILI9341 example based on latest modifications to the OSAL module
- Updated Coremark example based on latest modifications to the OSAL module
- Updated FLASH example to allow its execution from flash
- 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_1.5.0	new version
COMP	VER_1.2.1	new version
DAC	VER_2.1.0	new version
SARADC	VER_2.0.0	new version
SDADC	VER_2.0.0	new version
TSENS	VER_1.0.0	
CAN	VER_1.3.0	new version
I2C	VER_1.1.0	new version
I2S	VER_1.3.0	new version
SENT	VER_1.2.0	new version
SPI	VER_1.4.0	new version
UART	VER_1.3.0	new version
ILI9341	VER_1.2.0	
EED	VER_1.2.1	
FLASH	VER_1.4.0	new version
CEM	VER_2.0.0	new version
CMU	VER_1.0.0	
FCCU	VER_1.2.1	
CORDIC	VER_1.3.0	new version
CRC	VER_1.1.0	new version
DMA	VER_1.3.0	
EXTI	VER_1.2.0	
GPIO	VER_1.3.0	new version
HSEM	VER_1.2.0	
IWDG	VER_1.2.0	
MPU	VER_1.3.0	new version
SMPU	VER_1.0.1	new version
WWDG	VER_1.2.0	
DWT	VER_1.2.0	
HRTIM	VER_2.2.0	new version
RTC	VER_1.0.0	
SYSTICK	VER_1.3.0	new version
TIM	VER_1.5.0	new version
TIM_TS	VER_1.0.0	new
FreeRTOS(*)	VER_1.0.0	
OSAL	VER_1.5.0	new version
Board	VER_1.4.0	new version
Clock	VER_1.5.0	new version
CMSIS(**)	VER_0.9.0	
IRQ	VER_1.2.0	
MCU	VER_1.5.0	new version
IO	VER_1.2.0	
SDKTests	VER_2.1.0	new version

(*) FreeRTOS VER_1.0.0 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/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
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
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/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_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
```

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
```

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_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_1.5.0]

Added support for EVBE3000P.
Added support for EVBE7000E and EVBE3000E.
Updated build system to support new module TIM_TS.
Renamed default compiling target.

COMP [VER_1.2.1]

Fixed typos.

DAC [VER_2.1.0]

Updated DMA configuration settings.
Fixed single mode value setting.
Fixed typos + updated doxygen comments.

SARADC [VER_2.0.0]

Updated DMA configuration settings.
Added support for thresholds management.

Added support for dual mode.

SDADC [VER_2.0.0]

Fixed OSD minimum value.
Added capture timestamp request management.
Added trigger support.
Fixed FIFO size setting.

CAN [VER_1.3.0]

Updated DMA configuration settings.
Fixed bus off management.

I2C [VER_1.1.0]

Updated DMA configuration settings.

I2S [VER_1.3.0]

Updated DMA configuration settings.

SENT [VER_1.2.0]

Added management for different nibbles lengths.
Added API for the frequency setting of the TIM used for SENT emulation.
Fixed endianess issue.
Fixed multi-channel issue.
Enhanced driver performances.

SPI [VER_1.4.0]

Updated DMA configuration settings.
Added timeout management.

UART [VER_1.3.0]

Updated DMA configuration settings.
Added timeout management.
Fixed typos.

FLASH [VER_1.4.0]

Provided module source code.

CEM [VER_2.0.0]

Updated API related to the GROUP_WRITE_ ENABLE command.

CORDIC [VER_1.3.0]

Updated DMA configuration settings.

CRC [VER_1.1.0]

Updated DMA configuration settings.

GPIO [VER_1.3.0]

Updated APIs to garantee the set of the single parameter with a single register access.

MPU [VER_1.3.0]

Added API for region clearing/disabling.
Made region configurations updatable.

SMPU [VER_1.0.1]

Fixed typos.

HRTIM [VER_2.2.0]

Fixed ADC post-scaler setting.
Fixed ADC trigger setting.
Fixed blanking and windowing macros doxygen description.

SYSTICK [VER_1.3.0]

Added support for timeout management.

TIM [VER_1.5.0]

Updated DMA configuration settings.
Added new API for DMA triggering.
Added new API to return the TIM clock frequency.
Renamed trigger macros.
Removed register pointer deinitilization from the API tim_stop.

TIM_TS [VER_1.0.0]

Added TIM_TS driver.

OSAL [VER_1.5.0]

Added support for the timeout management.

Board [VER_1.4.0]

Added support for EVBE3000P, EVBE7000E and EVBE3000E.

Clock [VER_1.5.0]

Added APIs to manage the clock of the TIM_TS module.
Added check of the frequency range after the prescaler.

MCU [VER_1.5.0]

Added interrupt vector and interrupt handler for TIM_TS module.
Updated CAN header file.

SDKTests [VER_2.1.0]

Updated SDK examples based on SDK module updates.
Updated SDK examples to make them compatible with EVBE3000P and EVBE7000E.
Updated makefiles adding compiling target management.

Added Analog/SARADC/SARADC_06.
Added Analog/SARADC/SARADC_07.
Added Analog/SDADC/SDADC_02.
Added System/DMA/DMA_01.
Added System/DMA/DMA_02.
Added System/DMA/DMA_03.
Added System/MPU/MPU_02.
Added Timers/TIM_TS/TIM_TS_01.