life.augmented

Stellar-E SDK – Quick Start Guide

Introduction

This document is intended for software developers who want to develop an application for STM microcontrollers of Stellar-E family using the Stellar-E Software Development Kit (SDK).

This page was intentionally left blank.

life.augmented

Table of Contents

List of Tables

List of Figures

1 Overview

Stellar-E Software Development Kit (SDK) is designed to develop applications for microcontrollers of Stellar-E family. Stellar-E microcontrollers are listed in Table 1.

Family	Device
Stellar-E	SR5E1

Table 1: Stellar-E microcontrollers.

Stellar-E SDK includes a build system with the support for different compilers, a free toolchain (GNU ARM), a set of modules for core initialization and peripheral drivers and a set of examples to show how to configure and use the device and related peripherals. The build system is based on makefiles.

Stellar-E SDK is distributed as a standalone package (in this case, a second standalone package containing the free toolchain can be provided) or as part of StellarStudio, a framework based on Eclipse that in addition to the SDK includes the free toolchain, a set of plugins for Stellar-E SDK module configuration and for code debugging.

Stellar-E SDK is organized in two folders:

- Modules
- Projects

Modules contains a set of modules that can be used to create the user application. Some modules are specific for core initialization (boot procedure, board initialization, clock initialization, interrupt management), some other are specific for peripheral management (low level drivers).

Projects contains a set of examples that show how to configure the core and the related peripherals using the SDK modules.

2 Stellar-E SDK modules

The Stellar-E SDK modules are listed in Table 2.

Module	Description
Boot	Boot initialization module (mandatory)
Board	Board GPIO initialization module (based on GPIO low level driver)
Clock	Clock initialization module
CMSIS	ARM Common Microcontroller Software Interface Standard
IRQ	Interrupt module
MCU	Microcontroller peripheral header files and IRQ vectors
CAN	FDCAN low level driver module
CEM	CEM low level driver module
CMU	CMU low level driver module
COMP	COMPARATOR low level driver module
CORDIC	CORDIC low level driver module
CRC	CRC low level driver module
DAC	DAC low level driver module
DMA	DMA low level driver module
DWT	DWT low level driver module
EED	EEPROM Emulation driver module
EXTI	EXTI low level driver module
FCCU	FCCU low level driver module
FLASH	FLASH low level driver module
FreeRTOS	FreeRTOS module
GPIO	GPIO low level driver module
HRTIM	HRTIM low level driver module
HSEM	Hardware Semaphore low level driver module
I2C	I2C low level driver module
I2S	I2S low level driver module
ILI9341	ILI9341 display low level driver module
IWDG	IWDG low level driver module
MPU	Memory Protection Unit low level driver module
RTC	RTC low level driver module
SARADC	SARADC low level driver module
SDADC	SDADC low level driver module
SENT	SENT low level driver module

Module	Description
SMPU	SMPU low level driver module
SPI	SPI low level driver module
SYSTICK	SYSTICK low level driver module
TIM	TIM low level driver module
TIM_TS	TIM_TS low level driver module
TSENS	Temperature Sensor low level driver module
UART	UART low level driver module
WWDG	WWDG low level driver module
OSAL	Operating System Abstraction Layer module
RuntimeIO	Runtime IO module (based on UART low level driver)

Table 2: Stellar-E SDK modules.

Each module has a dedicated makefile and a dedicated build variable that allows to include/exclude it in/from the building of a specific project. To include a module in a project, it's enough to set to '1' the related build variable in the project makefile. The build variables of each module are listed in the

Module Build Variable	Description	Default	Dependencies
BUILD_BOOT	set to '1' to build Boot module, '0' to exclude it from build.	1	
BUILD_PLATFORM_BOARD	set to '1' to build Board module, '0' to exclude it from build.	1	BOARD module is based on GPIO module, so GPIO module is required (BUILD_DRIVERS_GPIO must be set).
BUILD_PLATFORM_CLOCK	set to '1' to build Clock module, '0' to exclude it from build.	1	
BUILD_PLATFORM_CMSIS	set to '1' to build CMSIS module, '0' to exclude it from build.	1	
BUILD_PLATFORM_IRQ	set to '1' to build IRQ module, '0' to exclude it from build.	1	
BUILD_PLATFORM_MCU	set to '1' to build MCU module, '0' to exclude it from build.	1	
BUILD_DRIVERS_CAN	set to '1' to build FDCAN driver, '0' to exclude it from build.	0	CAN module is based on DMA module, so DMA module is required (BUILD_DRIVERS_DMA must be set).
BUILD_DRIVERS_CEM	set to '1' to build CEM driver, '0' to exclude it from build.	0	
BUILD_DRIVERS_CMU	set to '1' to build CMU driver, '0' to exclude it from build.	0	
BUILD_DRIVERS_COMP	set to '1' to build COMPARATOR module, '0' to exclude it from build.	0	COMP module is based on EXTI module, so EXTI module is required (BUILD_DRIVERS_EXTI must be set).
BUILD_DRIVERS_CORDIC	set to '1' to build CORDIC module, '0' to exclude it from build.	0	CORDIC module is based on DMA module, so DMA module is required (BUILD_DRIVERS_DMA must be set).
BUILD_DRIVERS_CRC	set to '1' to build CRC module, '0' to exclude it from build.	0	CRC module is based on DMA module, so DMA module is required (BUILD_DRIVERS_DMA must be set).
BUILD_DRIVERS_DAC	set to '1' to build DAC driver, '0' to exclude it from build.	0	DAC module is based on DMA module, so DMA module is required (BUILD_DRIVERS_DMA must be set).

Module Build Variable	Description	Default	Dependencies
BUILD_DRIVERS_DMA	set to '1' to build DMA driver, '0' to exclude it from build.	0	
BUILD_DRIVERS_DWT	set to '1' to build DWT driver, '0' to exclude it from build.	0	
BUILD_DRIVERS_EED	set to '1' to build EED driver, '0' to exclude it from build.	0	EEPROM Emulation Driver module is based on FLASH module, so FLASH module is required (BUILD_DRIVERS_FLASH must be set).
BUILD_DRIVERS_EXTI	set to '1' to build EXTI driver, '0' to exclude it from build.	0	
BUILD_DRIVERS_FCCU	set to '1' to build FCCU driver, '0' to exclude it from build.	0	
BUILD_DRIVERS_FLASH	set to '1' to build FLASH driver, '0' to exclude it from build.	0	
BUILD_DRIVERS_GPIO	set to '1' to build GPIO driver, '0' to exclude it from build.	1	
BUILD_DRIVERS_HRTIM	set to '1' to build HRTIM driver, '0' to exclude it from build.	0	
BUILD_DRIVERS_HSEM	set to '1' to build HSEM driver, '0' to exclude it from build.	0	
BUILD_DRIVERS_I2C	set to '1' to build I2C driver, '0' to exclude it from build.	0	I2C module is based on DMA module, so DMA module is required (BUILD_DRIVERS_DMA must be set).
BUILD_DRIVERS_I2S	set to '1' to build I2S driver, '0' to exclude it from build.	0	I2S module is based on DMA module, so DMA module is required (BUILD_DRIVERS_DMA must be set).
BUILD_DRIVERS_ILI9341	set to '1' to build ILI9341 driver, '0' to exclude it from build.	0	ILI9341 module is based on SPI and DMA modules, so SPI and DMA modules are required (BUILD_DRIVERS_SPI and BUILD_DRIVERS_DMA must be set).
BUILD_DRIVERS_IWDG	set to '1' to build IWDG driver, '0' to exclude it from build.	0	
BUILD_DRIVERS_MPU	set to '1' to build MPU driver, '0' to exclude it from build.	0	
BUILD_DRIVERS_RTC	set to '1' to build RTC driver, '0' to exclude it from build.	0	
BUILD_DRIVERS_SARADC	set to '1' to build SARADC driver, '0' to exclude it from build.	0	SARADC module is based on DMA module, so DMA module is required (BUILD_DRIVERS_DMA must be set).
BUILD_DRIVERS_SDADC	set to '1' to build SDADC driver, '0' to exclude it from build.	0	
BUILD_DRIVERS_SENT	set to '1' to build SENT driver, '0' to exclude it from build.	0	SENT module is based on TIM module, so TIM module is required (BUILD_DRIVERS_TIM must be set).
BUILD_DRIVERS_SMPU	set to '1' to build SMPU driver, '0' to exclude it from build.		
BUILD_DRIVERS_SPI	set to '1' to build SPI driver, '0' to exclude it from build.	0	SPI module is based on DMA module, so DMA module is required (BUILD_DRIVERS_DMA must be set).
BUILD_DRIVERS_TIM	set to '1' to build TIM driver, '0' to exclude it from build.	0	TIM module is based on DMA module, so DMA module is required (BUILD_DRIVERS_DMA must be set).
BUILD_DRIVERS_TIM_TS	set to '1' to build TIM_TS driver, '0' to exclude it from build.	0	
BUILD_DRIVERS_TSENS	set to '1' to build TSENS driver, '0' to exclude it from build.	0	

Module Build Variable	Description	Default	Dependencies
BUILD_DRIVERS_UART	set to '1' to build UART driver, '0' to exclude it from build.	0	UART module is based on DMA module, so DMA module is required (BUILD_DRIVERS_DMA must be set).
BUILD_DRIVERS_WWDG	set to '1' to build WWDG driver, '0' to exclude it from build.	0	
BUILD_OS_OSAL	set to '1' to build OSAL module, '0' to exclude it from build.	0	Since OSAL module is based on SYSTICK module, SYSTICK module is automatically included (BUILD_DRIVERS_SYSTICK is set) when BUILD_OS_OSAL is set.
BUILD_OS_FREERTOS	set to '1' to build FreeRTOS module, '0' to exclude it from build.	0	
BUILD_RUNTIME_IO	set to '1' to build RuntimeIO module, '0' to exclude it from build.	0	Runtime IO module is based on UART module, so UART module is required (BUILD_DRIVERS_UART and BUILD_DRIVERS_DMA must be set).

Table 3: Stellar-E SDK module build variables.

CMSIS (Common Microcontroller Software Interface Standard) module is a vendor-independent abstraction layer provided by ARM for microcontroller based on ARM Cortex processor. The version on which the Stellar-E SDK is based on is the CMSIS 5.8.0 (www.arm.com/technologies/cmsis).

3 Stellar-E SDK examples

Stellar-E SDK includes a set of examples/projects located in the folder ***<StellarESDK>\Projects\SDKTests*** to show how to configure and use the Stellar-E SDK modules. For each example a file ***readme.md*** with the description and requirements is provided in the same folder.

For all SDK examples the following build variables are enabled by default

- BUILD_BOOT
- BUILD_PLATFORM_BOARD
- BUILD_PLATFORM_CLOCK
- BUILD_PLATFORM_CMSIS
- BUILD_PLATFORM_IRQ
- BUILD_PLATFORM_MCU

since the related modules are mandatory for the SDK examples. If it is necessary to exclude one of these modules for a specific project, it is enough to set to 0 the related build variable in the project makefile.

4 Stellar-E SDK example compilation

If the Stellar-E SDK standalone package is used, to compile an example with the default compiling options and with the default compiler, the following steps are needed:

1. Uncompress the Stellar-E SDK standalone package (***StellarESDK_x.y.z.tgz*** where x.y.z is the Stellar-E SDK version) and additional standalone package (***Tools_x.y.z.zip*** where x.y.z is the Tool version) including the free compiler in the same folder ***<SESDK>***. The folder ***<SESDK>*** will have the following structure:
   <SESDK>/Modules
   <SESDK>/Projects
   <SESDK>/Tools
2. Select the example to compile in the folder ***<SESDK>/Projects/SDKTests*** (e.g. System/GPIO/GPIO_01)
3. Open a shell in its folder (e.g. ***<SESDK>/Projects/SDKTests/System/GPIO/GPIO_01***) and run the following build command:

   ```
   make all
   ```

When the build process is completed, the binary file will be located in the subfolder ***build*** within the example folder (e.g. ***<SESDK>/Projects/SDKTests/System/GPIO/GPIO_01/build***). In particular, the folder ***build*** will contain some subfolders. The exact position of the binary file depends on the compiling options (e.g. if the example is compiled for device = SR5E1, platform = EVBE7000P, core = CORE1 and optimization level = Debug, the related binary file will be located in the subfolder ***sr5e1/evbe7000p/core1/Debug*** within folder ***build*****)**. For more details on compiling options, refer to the next section.

If StellarStudio is used, Stellar-E SDK and free toolchain are already included in StellarStudio. The compiling procedure (steps 2-3 above) is the same, but it must be directly executed from StellarStudio internal shell.

5 Stellar-E SDK compiling options

Stellar-E SDK build system offers a set of compiling options to customize the build process. The build options are summarized in the Table 4.

Compiling option	Values	Default	Description
TARGET_BUILD	Debug Release	Debug	Specifies the compiling optimization level. If this option is not specified, the default value (Debug) is used.
VERBOSE	0 1	0	Enables/Disables the verbose output during compiling process. If this option is not specified, the default value (0, no verbose) is used.
TOOLCHAIN	ARM HIGHTEC GHS IAR	ARM	Specifies the compiler used to compile the project. If this option is not specified, the default value (ARM) is used.
CONFIG_DEVICE	sr5e1	sr5e1	Specifies the device for which the project is compiled. Currently only sr5e1 is supported. If this option is not specified, the default value (sr5e1) is used.
CONFIG_BOARD	evbe7000p evbe7000s evbe3000p evbe7000e evbe3000e	evbe7000p	Specifies the board for which the project is compiled. Currently only evbe7000p is fully supported. If this option is not specified, the default value (evbe7000p) is used.
CONFIG_TARGET_CORE	core1 core2	core1	Specifies the core for which the project is compiled. If this option is not specified, the default value (core1) is used.
CONFIG_TARGET_MEMORY	ram nvm flash	ram	Specifies the memory for which the project is compiled. If this option is not specified, the default value (ram) is used. NOTE: flash and nvm are equivalent.
CONFIG_FPU	hard softfp	hard	Specifies if floating points are fully managed by FPU or not. If this option is not specified, the default value (hard) is used.
MAIN_STACK_SIZE	-	4k	Specifies the size of the main stack. If this option is not used, the default value (4k) is used.
PROCESS_STACK_SIZE	-	4k	Specifies the size of the process stack. If this option is not used, the default value (4k) is used.
HEAP_SIZE	-	1k	Specifies the size of the heap. If this option is not used, the default value (1k) is used. NOTE: This option is used only with IAR compiler. For the other compilers, the size of the heap is automatically set to the size of RAM not used by the code and data.

Table 4: Stellar-E SDK compiling options.

So, for example, the following build command

```
make CONFIG_TARGET_CORE=core2 CONFIG_TARGET_MEMORY=nvm all
```

will compile the project for the following options

- TARGET_BUILD=Debug
- VERBOSE=0
- TOOLCHAIN=ARM
- CONFIG_DEVICE=sr5e1
- CONFIG_BOARD=evbe7000p
- CONFIG_TARGET_CORE=core2
- CONFIG_TARGET_MEMORY=nvm
- CONFIG_FPU=hard
- MAIN_STACK_SIZE=4k
- PROCESS_STACK_SIZE=4k
- HEAP_SIZE=1k

Table 5 provides more details about the supported compilers.

Compiler
GNU ARM Embedded Toolchain 10.3-2021.10
HighTec clang version 8.1.0
IAR ANSI C/C++ Compiler V9.20.4.327/W64
GHS MULTI Compiler v2021.1

Table 5: Stellar-E SDK supported compilers.

NOTE: To use Stellar-E SDK with GHS compiler, CMSIS support for GHS is required. Please, refer to GHS for it.

6 Stellar-E SDK linker script

Stellar-E SDK default linker script divides the NVM and RAM in two parts: the first one is reserved to the core1, the second one to the core2. For SR5E1 the memory is divided as below

```
core1_nvm (rx): org = 0x08000000, len = 960k
core2_nvm (rx): org = 0x080F0000, len = 960k
core1_ram (wx): org = 0x24000000, len = 128k
core2_ram (wx): org = 0x24020000, len = 128k
```

This means that an SDK project is compiled to be executed from

- **0x08000000** if CONFIG_TARGET_CORE=core1, CONFIG_TARGET_MEMORY=nvm
- **0x80F00000** if CONFIG_TARGET_CORE=core2, CONFIG_TARGET_MEMORY=nvm
- **0x24000000** if CONFIG_TARGET_CORE=core1, CONFIG_TARGET_MEMORY=ram
- **0x24020000** if CONFIG_TARGET_CORE=core2, CONFIG_TARGET_MEMORY=ram

If a customized linker script is required, it is enough to specify its path in the project makefile.

Figure 1 shows an example of memory layout in the case of SDK project compiled for NVM. The size of the Main Stack is specified by MAIN_STACK_SIZE, the size of the Process Stack is specified by PROCESS_STACK_SIZE and the size of the Heap is the available RAM (from the end of the Process Stack to the end of the RAM) for GCC, HighTec and GHS or is specified by HEAP_SIZE for IAR.

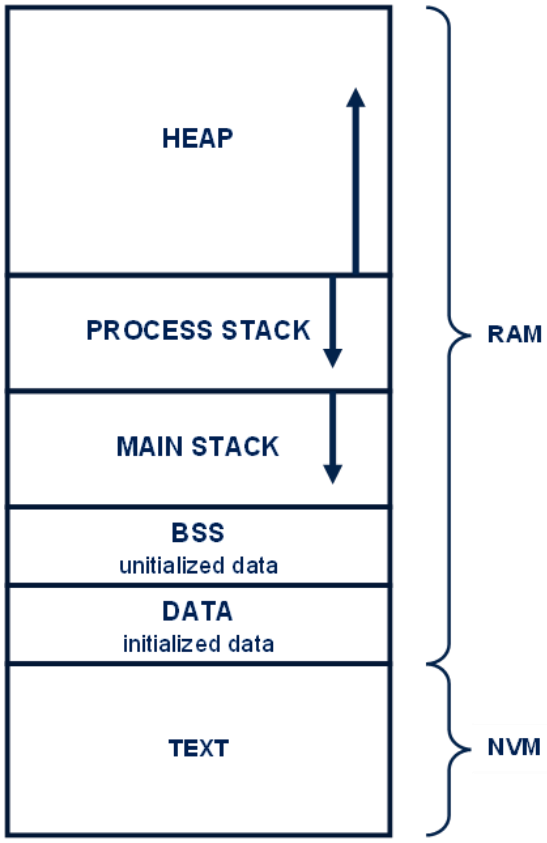

Figure 1: Memory layout

7 Stellar-E Tightly coupled RAMs

SR5E1 includes an Instruction and Data Tightly Coupled RAMs (ITCM and DTCM) directly connected through dedicated TCM buses to the Cortex-M7 cores. ITCM and DTCM are initialized by the Stellar-E SDK during the boot procedure. ITCM is accessed by its related Cortex-M7 core at CPU clock speed, with zero wait states. So, it can be used to speed the code execution. To execute a specific function/task from ITCM, it is enough to include it in the section ***itcm*** using the following attribute

```
__attribute__((section(".itcm")))
```

The function/task will be relocated in ITCM by the boot code of the Stellar-E SDK during the boot procedure and will be executed directly from ITCM.

At the same way, to speed data access, the data can be allocated in DTCM including it in the section ***dtcm*** with the following attribute

```
__attribute__((section(".dtcm")))
```

8 Stellar-E Boot procedure

In the current version of the Stellar-E SDK, when the code for core1 is compiled to be executed from flash (entry point = 0x08000000), the linker script places the vector table at address 0x0x08000000 and the reset hander at address 0x08000400 (addresses [0x08000000:0x080003FF] are reserved to the core1 vector table).

For devices with BAF (Boot Assist from Flash) v.1.0, if the BAF is enabled, a dedicated Device Configuration Format (DCF) record (named **Core1 User Application Address**) must be programmed with the start address of the user application (reset handler, 0x08000400).

For devices with BAF v.2.0 (with BAF already enabled in factory), the same dedicated DCF record (renamed **Core1 User VTOR Address**) must be programmed with the application vector table base address (VTOR, 0x08000000). For more details, refer to the RM.

9 Revision history

Date	Revision	Changes
28-Sep-2023	1.5.0	Updated to StellarE SDK revision 1.5.0
28-Jun-2023	1.4.0	Updated to StellarE SDK revision 1.4.0
05-Apr-2023	1.3.0	Updated to StellarE SDK revision 1.3.0
03-Mar-2023	1.2.0	Updated to StellarE SDK revision 1.2.0
22-Dic-2022	1.1.0	Updated to StellarE SDK revision 1.1.0
28-Sep-2022	1.0.0	Initial release