



SPC58EC80 c-startup example

Quick Guide

Table of Contents

1. Terms & Abbreviations	1
2. Introduction	2
2.1. Prerequisites	2
2.2. HW resources	2
3. Example overview	3
3.1. Application perspective	3
3.2. Component view	3
3.2.1. Application component	4
3.2.2. BSP component	4
3.3. Execution flow	5
4. Example Import & Build	6
5. Microcontroller default configuration	7
5.1. Microcontroller platform setup	7
5.2. BSP default settings	7
Appendix A: Document References	9
Appendix B: Document history	10

1. Terms & Abbreviations

BSP

Board Startup Package

crt0

'C' run-time environment initialization code

HAL

Hardware Abstraction Layer

EVB

Evaluation Board

PIT

Periodic Interrupt Timer

uC

Microcontroller

2. Introduction

A c-startup example is a small functional project designed to simplify an evaluation phase of the SPC58EC80 microcontroller. It comes with necessary low-level functions like a startup code, a minimalistic hardware abstraction or predefined memory partitioning. On top of this low-level implementation, it provides a reference application code running on all present cores.

2.1. Prerequisites

- SPC58EC80 device
- An evaluation board (SPC58EC-DISP Discovery Board)
- HighTec Development Suite, version: 4.9.3.0

2.2. HW resources

Hardware resources used by this example:

HW unit	channel / output pin	function
GPIO	PA [0]	Core [0] LED control
GPIO	PA [2]	Core [2] LED control
PIT 0	Channel [0]	Core [0] periodic interrupt
PIT 0	Channel [2]	Core [2] periodic interrupt

Tab. 1. HW resources

By default, the evaluation board does not have a connection between PA port and LEDs. To enable LED control, the user needs to connect wires between Port A header and User LEDs.

Jumper	PCB Legend	Description
JP44	LED_1	OPEN
JP6	PA0	connected to TP39 LED_1 (User_LED_1)
JP49	LED_3	OPEN
X9	C17	(PA2)connected to TP40 LED_3 (User_LED_3)

Tab. 2. SPC58EC-DISP Discovery Board configuration

3. Example overview

3.1. Application perspective

From the user perspective, the application functionality consists of a simple LED blinking, where each active core toggles its LED through a dedicated uC pin. The initialization code sets the toggling period in multiples of 250ms according to CoreId number read during the run-time.

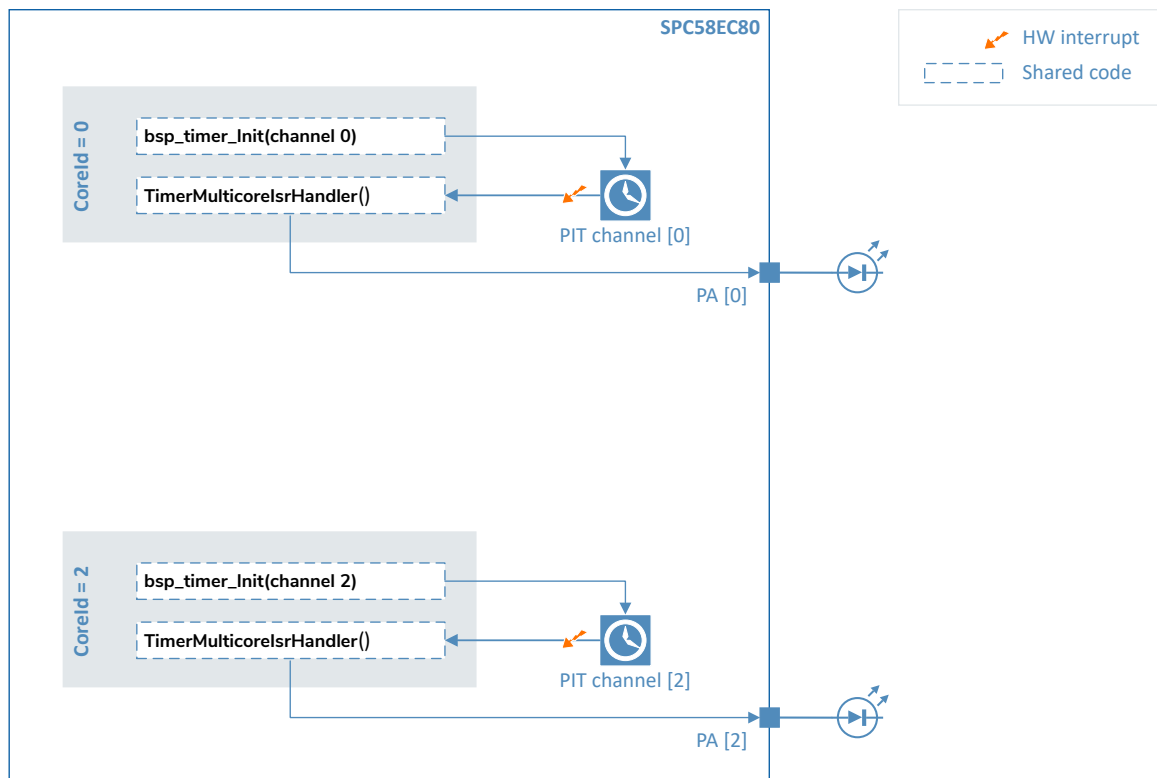


Fig. 1. Application view

3.2. Component view

The example consists of two components, Application (app), and BSP (bsp), each of them in its project folder.

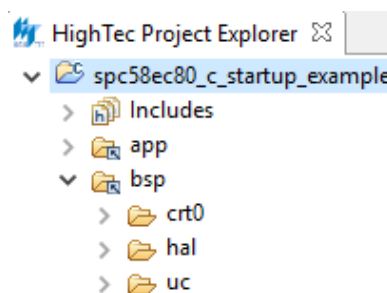


Fig. 2. Project folder view

3.2.1. Application component

A reference application implementing the functionality of the example, LED blinking by each core. The execution flow is the same on each core. Application folder represents a playground where the user can extend the example functionality.

3.2.2. BSP component

A BSP package represents toolchain and microcontroller dependent SW part that needs to be tailored to a particular microcontroller derivative, here to SPC58EC80. It assembles three related elements.

crt0

A mandatory 'C' runtime initialization (crt0), running immediately after the hardware reset up to the user entry point, here `shared_main()` function.

hal

A default hardware platform configuration and a user API for selected uC hardware modules. The BSP platform configuration prepares uC to its optimal execution performance. The application part calls the platform initialization function during crt0 PreInit callback.

uc

Microcontroller dependent low-level implementation of BSP API interface functions. Plus it contains pre-defined memory partitioning provided in a dedicated linker file.

Detail description of BSP elements is part of following technical notes [\[1\]](#), [\[2\]](#), [\[3\]](#).

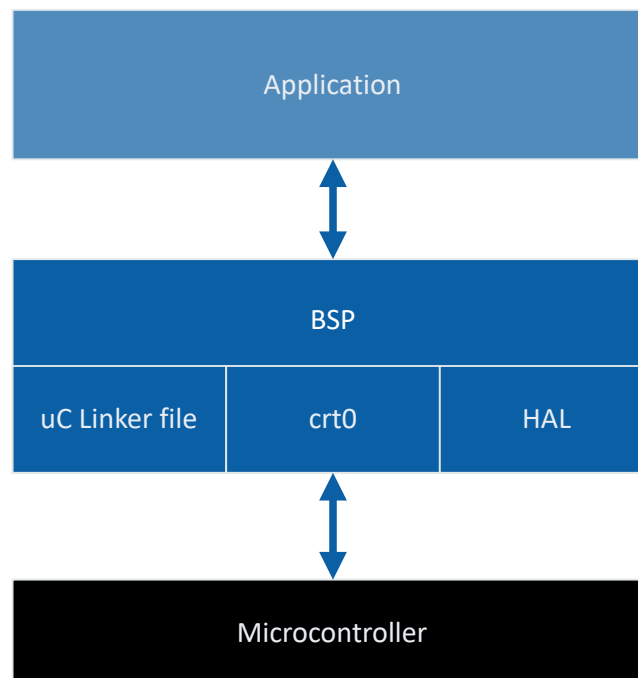


Fig. 3. Component view

3.3. Execution flow

The RESET core is responsible for the initialization of shared resources before enabling inactive cores. Once other core starts running, they can rely on stable hardware environment.

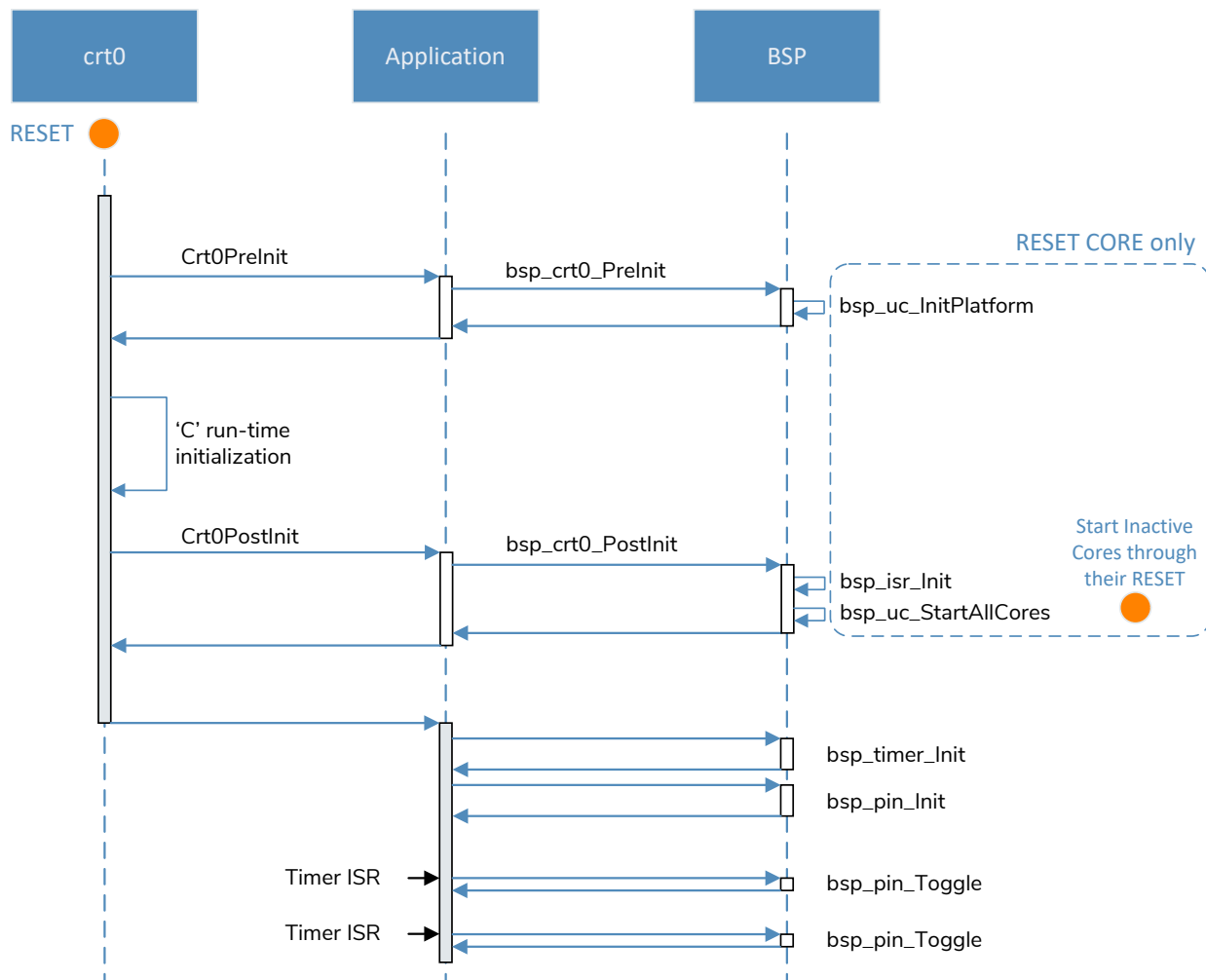


Fig. 4. Simplified individual core execution flow

All software parts of the example decide the right execution path according to CoreID read during the run-time. Such run-time read allows to get core specific behavior and still share the same code. Such shared approach simplifies microcontroller learning phase and first trials in case of a multi-core platform.

4. Example Import & Build

Follow these steps to import an example project to the HighTec IDE environment:

1. From menu **File** → **Import** → **General** choose an option Existing Projects into Workspace
2. Browse for your project location
3. Select project
4. Set checkbox **Copy project into workspace** option, leaving copy to the workspace option empty
5. Click Finish

Activate the project from menu **Project** → **Set Active Project**.

Build the project from the menu **Project** → **Build Project**.

The output binary file is located under the `_irom_build` folder.

5. Microcontroller default configuration

5.1. Microcontroller platform setup

c-startup example comes with a predefined SPC58EC80 hardware platform configuration that is part of `bsp\uc\uc_spc58ec80\uc_spc58ec80_conf.h` file.

There two clock parameters that the user can override outside the BSP package to adjust the clock system according to physical properties of the evaluation board.

Code 1. SPC58EC80 User controlled clock parameters

```
/* XTAL_CLOCK: External crystal clock */
#ifndef UC_XTAL_CLOCK
#define UC_XTAL_CLOCK          40
#endif

/* SYS_CLOCK: System clock for cores [MHz] */
#ifndef UC_SYSTEM_CLOCK
#define UC_SYSTEM_CLOCK        180
#endif
```

Based on these two clock parameters, the BSP adjusts microcontroller platform configuration to get optimal execution performance. In the same time, it keeps selected peripheral clocks setting invariant of user-controlled clock parameters.

Code 2. SPC58EC80 default values of selected peripheral clocks

```
/* Peripheral clocks */
/* AUX0 */
#define UC_SARADC_CLK 10
#define UC_EMIOS_CLK 100
/* AUX1 */
#define UC_LFAST_PLL 25
/* AUX2 */
#define UC_FRAY_CLK 40
/* AUX8 */
#define UC_CAN_CLK0 80
/* AUX11 */
#define UC_CAN_CLK1 80
/* AUX12 */
#define UC_DSPI_CLK0 100
#define UC_DSPI_CLK1 100
#define UC_LIN_CLK0 100
#define UC_LIN_CLK1 80
#define UC_PER_CLK0 100
```

5.2. BSP default settings

BSP package comes with a predefined setting to make a startup phase with the microcontroller an

easy task. The user can change this setting by assigning corresponding symbols a different value.

Code 3. SPC58EC80 BSP default setting

```
/* BSP HW Initialization control
 * 0 = OFF, no checks of correct UC configuration values
 * 1 = ON (default), uC spec checks group of parameters for their
 * correct HW setting */
#ifndef BSP_HW_INIT_CHECK
#define BSP_HW_INIT_CHECK      1
#endif

/* BSP_CORE_ENABLE: Inactive Core Enable control
 * 0 = OFF, inactive cores will not be enabled by bsp_crt0_PostInit()
 * 1 = ON (default), inactive cores will be enabled by bsp_crt0_PostInit() */
#ifndef BSP_CORE_ENABLE
#define BSP_CORE_ENABLE        1
#endif

/* BSP_ISR_TABLE: BSP implementation of ISR table in RAM
 * 0 = OFF, BSP will not implement its bsp_isr_table[1024] in RAM
 * user must provide his own ISR table (either ROM or RAM)
 * 1 = ON (default), BSP implements its bsp_isr_table[1024] in RAM */
#ifndef BSP_ISR_TABLE
#define BSP_ISR_TABLE          1
#endif

/* BSP_PIT_SUPPORT: BSP PIT timer control enable/disable
 * 0 = OFF, PIT timer will not be initialized and PIT interface disabled
 * 1 = ON (default), PIT timer under BSP control */
#ifndef BSP_TIMER_SUPPORT
#define BSP_TIMER_SUPPORT       1
#endif

/* SYSClk0 Output Enable Control
 * 0 = OFF, switch off clock generation setting
 * 1 = ON, BSP will enable clock generation output */
#ifndef BSP_SYSClk0_OUT
#define BSP_SYSClk0_OUT         1
#endif
```

Appendix A: Document References

- [1] "SPC5x c-startup - 'C' run-time initialization", HighTec EDV Systeme GmbH, 2018
- [2] "SPC5x c-startup - Linker file template", HighTec EDV Systeme GmbH, 2018
- [3] "SPC5x c-startup - Hardware Abstraction Layer", HighTec EDV Systeme GmbH, 2018

Appendix B: Document history

Version	Date	Changes to the previous version
1.0	March 2018	Initial version



HighTec EDV-Systeme GmbH
Feldmannstrasse 98, D-66119 Saarbrücken
info@hightec-rt.com
+49-681-92613-16
www.hightec-rt.com