



SPC58NN84 GTM PWM 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. System Description 3
 - 3.2. Application Flow 4
 - 3.3. Interrupt Handling 4
 - 3.4. GTM module configuration 5
- 4. Example Import & Build 7**
- Appendix A: Document References 8**
- Appendix B: Release Notes 9**

1. Terms & Abbreviations

PWM

Pulse Width Modulation.

GTM

Generic Timer Module

TOM

Timer Output Module

PIT

Periodic Interrupt Timer

IOP

I/O processor

BSP

Board Startup Package

GPIO

General Purpose Input Output

crt0

'C' run-time environment initialization code

HAL

Hardware Abstraction Layer

EVB

Evaluation Board

NC

Not Configured

uC

Microcontroller

ISR

Interrupt Service Routine

2. Introduction

This example is a small functional project designed to simplify an evaluation phase of the certain peripheral on the microcontroller. It is built on the c-startup example providing 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 the peripheral example code running on single core.

2.1. Prerequisites

- SPC58NN84 device
- An evaluation board (SPC57xxMB + SPC58xxADPT292S)
- HighTec Development Suite, version: 4.9.3.0

2.2. HW resources

Hardware resources used by this example:

| HW unit | channel / I/O pin | function |
|--|-------------------|-----------------------------|
| GPIO | PB[9] | Core [2] TOM GTM CH[0] |
| GPIO | PB[10] | Core [2] TOM GTM CH[1] |
| GTM TOM[0] | Channel [0] | Core [2] PWM |
| GTM TOM[0] | Channel [1] | Core [2] periodic interrupt |
| default resources from the c-startup example | | |
| GPIO | PA [0] | Core [0] LED control |
| GPIO | PA [1] | Core [1] LED control |
| GPIO | PA [2] | Core [2] LED control |
| PIT 0 | Channel [0] | Core [0] periodic interrupt |
| PIT 0 | Channel [1] | Core [1] periodic interrupt |
| PIT 0 | Channel [2] | Core [2] periodic interrupt |

Tab. 1. HW resources

3. Example overview

The peripheral generates the PWM signal with a synchronous update of the duty cycle within GTM module. The duty cycle of the generated PWM periodically sweeps in the range from 1% up to 99% and backward. The PWM signal is routed to the external pins.

3.1. System Description

The figure below shows the system view of the example:

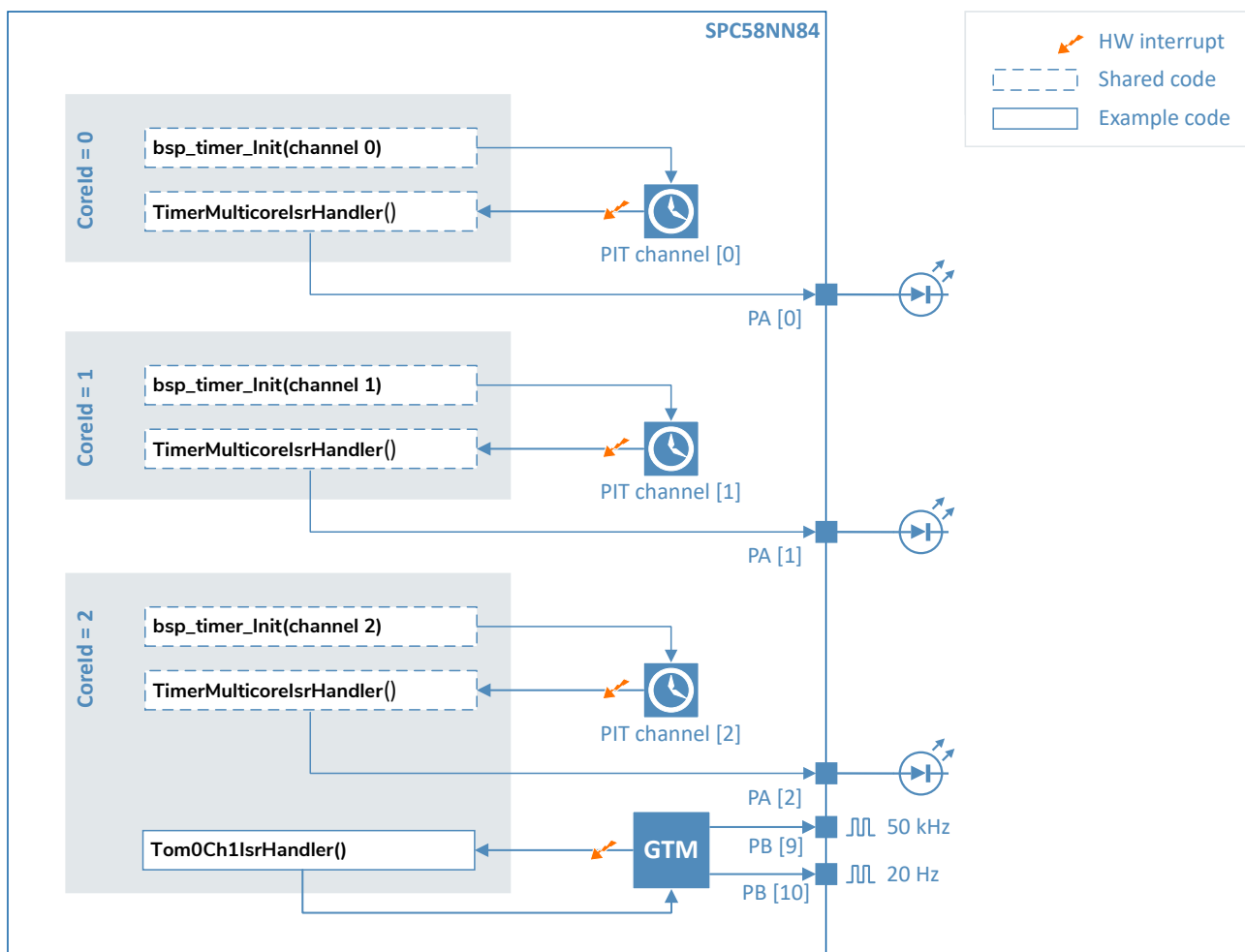


Fig. 1. System block diagram

The peripheral code runs on a single core. Other cores run the default functionality defined by SPC5xx_c_startup.

The example initializes two channels of the GTM TOM0 module.

- The channel 0 (CH0) generates the output PWM signal with the static frequency of 50kHz and with a dynamically changed duty cycle.
- The channel 1 (CH1) ensures a periodical update of the PWM duty cycle

The TOM0 CH1 timer generates an interrupt every 50ms in which PWM duty cycle is updated. One duty cycle increment means 1% and the sweep range is from 1% up to 99%.

The synchronous update of the duty cycle is ensured by writing a new value to shadow register SR1 without preceding disable of the update mechanism. The new duty cycle is applied in the period following the period where the update of register SR1 is done.

3.2. Application Flow

The UC_CORE_IOP core controls the GTM peripheral.

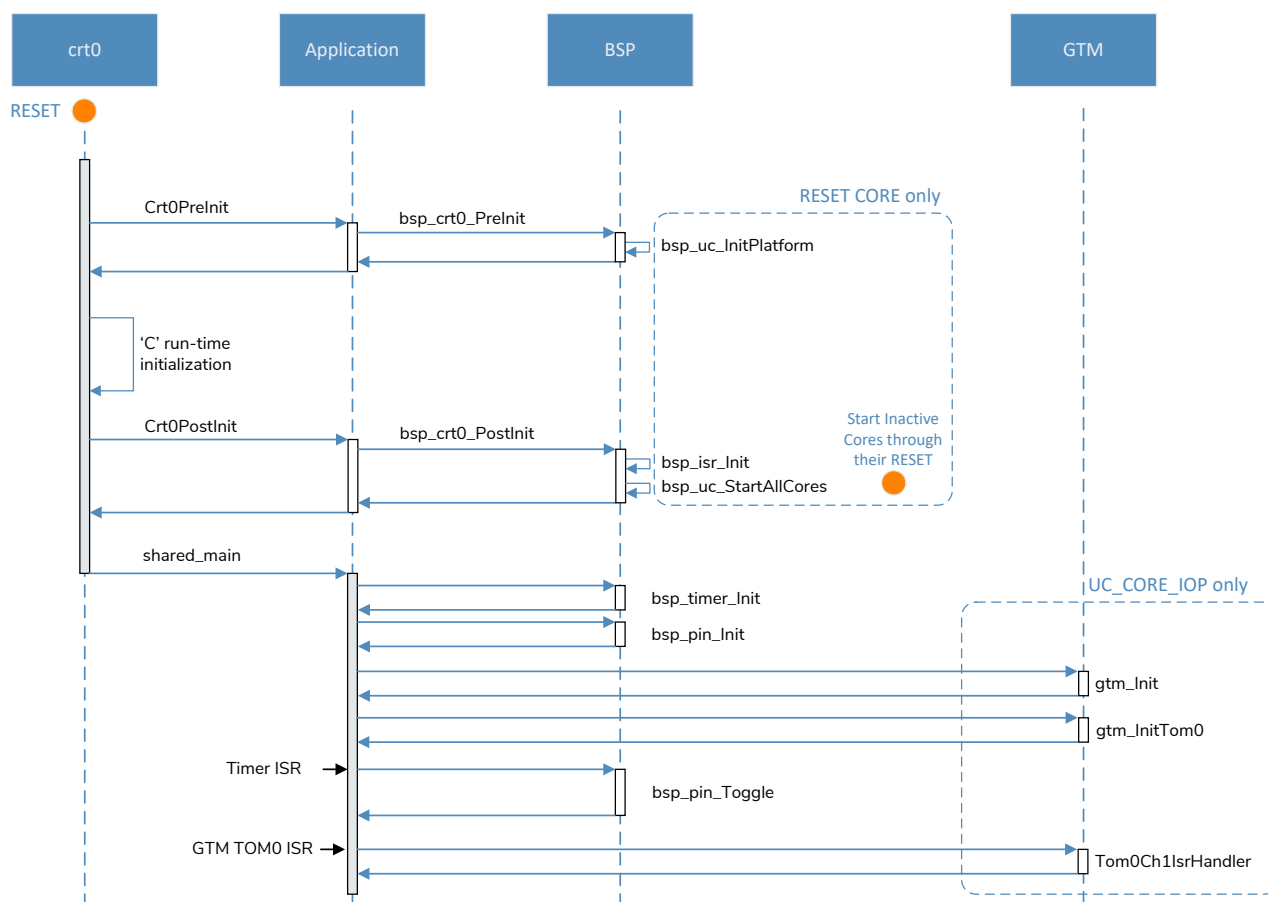


Fig. 2. Simplified individual core execution flow

3.3. Interrupt Handling

Implemented interrupt handling uses low-level BSP functions, for example to register each interrupt service routine (bsp_isr_RegisterHandler).

The GTM TOM0 initialization function assigns the "TOM0 channel 1 event" interrupt vector and sets the interrupt vector priority.

Code 1. /app/gtm.c

```
bsp_isr_RegisterHandler(UC_GTM_TOM0_CH1_ISR, 10, Tom0Ch1IsrHandler);
```

The example uses the SW interrupt mode.

3.4. GTM module configuration

The GTM module is configured with these parameters:

- GTM in normal mode
- AEI soft-reset control is disabled
- AEI timeout functionality is disabled
- $f_{\text{GTM}} = 200\text{MHz}$
- $f_{\text{CMU_GCLK_EN}} = 10\text{MHz}$
- $f_{\text{CMU_FXCLK0}} = f_{\text{CMU_GCLK_EN}} / 2^0 = 10\text{MHz}$
- $f_{\text{CMU_FXCLK1}} = f_{\text{CMU_GCLK_EN}} / 2^4 = 625\text{kHz}$

for more details see `gtm_Init()` and `gtm_InitTom0()` function below.

Code 2. /app/gtm.c

```
1 void gtm_Init(void)
2 {
3     /* Configure GTM */
4     GTMINT.MCR.R = 0; /* Enable GTM module, AEI soft-reset control is
5                        disabled, normal mode */
6     GTM.CTRL.U = 0; /* SW reset enabled, mode: Observe, AEI timeout
7                    functionality is disabled */
8     GTM.RST.U = 0x00000001; /* Initiate reset action for all submodules */
9
10    /* Init CMU
11     * setup global clock divider
12     * Tcm_gclk_en = (CMU_GCLK_NUM / CMU_GCLK_DEN) * Tsys_clk */
13
14    /* setup global clock divider - CMU_GCLK_EN = GTM_CLK / 2 */
15    GTM.CMU.GCLK_NUM.U = GTM_CMU_CFGU_GLOBAL_CLK_DIV;
16    GTM.CMU.GCLK_DEN.U = 0x1;
17
18    /* enable the clock pre-scalers */
19    GTM.CMU.FXCLK_CTRL.B.FXCLK_SEL = 0; /* CMU_GCLK_EN selected - Input clock
20                                         selection for EN_FXCLK line */
21    GTM.CMU.CLK_EN.U = 0x00800000; /* Enable FXCLK clock signal */
22
23    ...
24
25 }
```

Code 3. /app/gtm.c

```

1 void gtm_InitTom0 (void)
2 {
3     /* TOM global control configuration
4      * Enable update of register CM0, CM1 and CLK_SRC_STAT from SR0, SR1
5      * and CLK_SRC, no reset action after write by CPU, no trigger request */
6     GTM.TOM[0].TGC[0].GLB_CTRL.U = 0x000A0000;
7     GTM.TOM[0].TGC[0].ACT_TB.U = 0x00000000; /* TBU0_TS0 selected */
8
9     /* TOM0 channel 0 configuration
10      * SL=1, GTM_TOM0_CH0_CTRL_CLK_SRC_SR as a source clock, Reset CN0 to 0 on
11      * matching comparison CM0, TRIG_CCU0, One-shot mode disabled, SPE mode
12      * disabled, Gated Counter mode disabled */
13     GTM.TOM[0].CH[0].CTRL.U = \
14         (uint32_t)(0x01000800 | (GTM_TOM0_CH0_CTRL_CLK_SRC_SR << 12));
15
16     /* set period: (FXCLK / period) */
17     GTM.TOM[0].CH[0].SR0.U = (uint16_t)GTM_TOM0_CH0_PERIODE_TICKS;
18     /* set duty cycle: (period * duty) / 100 */
19     GTM.TOM[0].CH[0].SR1.U = (uint16_t)(GTM_TOM0_CH0_PERIODE_TICKS/2);
20     /* set period*/
21     GTM.TOM[0].CH[0].CM0.U = (uint16_t)GTM_TOM0_CH0_PERIODE_TICKS;
22     /* set duty cycle */
23     GTM.TOM[0].CH[0].CM1.U = (uint16_t)(GTM_TOM0_CH0_PERIODE_TICKS/2);
24     GTM.TOM[0].CH[0].CN0.U = (uint16_t)GTM_TOM0_CH0_PERIODE_TICKS;
25
26     /* TOM0 channel 1 configuration
27      * SL=1, GTM_TOM0_CH1_CTRL_CLK_SRC_SR as a source clock, Reset CN0 to 0 on
28      * matching comparison CM0, TRIG_CCU0, One-shot mode disabled, SPE mode
29      * disabled, Gated Counter mode disabled */
30     GTM.TOM[0].CH[1].CTRL.U = \
31         (uint32_t)(0x01000800 | (GTM_TOM0_CH1_CTRL_CLK_SRC_SR << 12));
32
33     /* set period - (FXCLK / period) */
34     GTM.TOM[0].CH[1].SR0.U = (uint16_t)GTM_TOM0_CH1_PERIODE_TICKS;
35     /* set duty cycle -(period * duty) / 100 */
36     GTM.TOM[0].CH[1].SR1.U = \
37         (uint16_t)((GTM_TOM0_CH1_PERIODE_TICKS * GTM_TOM0_CH1_DUTY)/100);
38
39     /* set period */
40     GTM.TOM[0].CH[1].CM0.U = (uint16_t)GTM_TOM0_CH1_PERIODE_TICKS;
41     /* set duty cycle */
42     GTM.TOM[0].CH[1].CM1.U = \
43         (uint16_t)((GTM_TOM0_CH1_PERIODE_TICKS * GTM_TOM0_CH1_DUTY)/100);
44     GTM.TOM[0].CH[1].CN0.U = GTM_TOM0_CH1_PERIODE_TICKS;
45
46     /* enable counter CCU0TC interrupt */
47     GTM.TOM[0].CH[1].IRQ_EN.B.CCU0TC_IRQ_EN = 1;
48
49     ...
50
51     /* enable TOM0 Ch0,1 (start counters) */
52     GTM.TOM[0].TGC[0].ENDIS_STAT.U = 0x0000000A;
53     /* enable output for TOM0 Ch0,1 */
54     GTM.TOM[0].TGC[0].OUTEN_STAT.U= 0x0000000A;
55 }

```


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. Leave the 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.

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: Release Notes

| Version | Date | Changes to the previous version |
|---------|---------------|--|
| 1.0 | December 2017 | Initial version |
| 2.0 | March 2018 | Update of BSP to the c-startup version 2.1 Example functionality alignment with other SPC derivatives Document alignment |



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