G4XS_KL25Z_XSG_IncrementalPWM

Processor Expert Example Project

Date: *21. 11. 2016*

Revision: 2.0

Overview

The purpose of this example project is to show how to use incremental PWM feature of Gen4 eXtremeSwitch. This feature allows to increment PWM duty of selected channels simultaneously by a given step. Utilization of this feature can be for example LED dimming or FAN speed regulation.

Requirements

Kinetis Design Studio 3.2.0 and newer

FRDM-KL25Z MCU board

FRDM-32XSG-EVB Gen4 eXtremeSwitch board

2 x channel load

USB cable

Setting up hardware

Target platform for this example is FRDM-KL25Z MCU board with FRDM-32XSG-EVB eXtremeSwitch board.

Chip select on FRDM-32XSG-EVB board has to be properly configured in such way that CSB0 pin is used. This can be achieved by switching **number 0 on SW2 switch** on the eSwitch board to **ON position**. The rest of CSB routes should remain in OFF position.
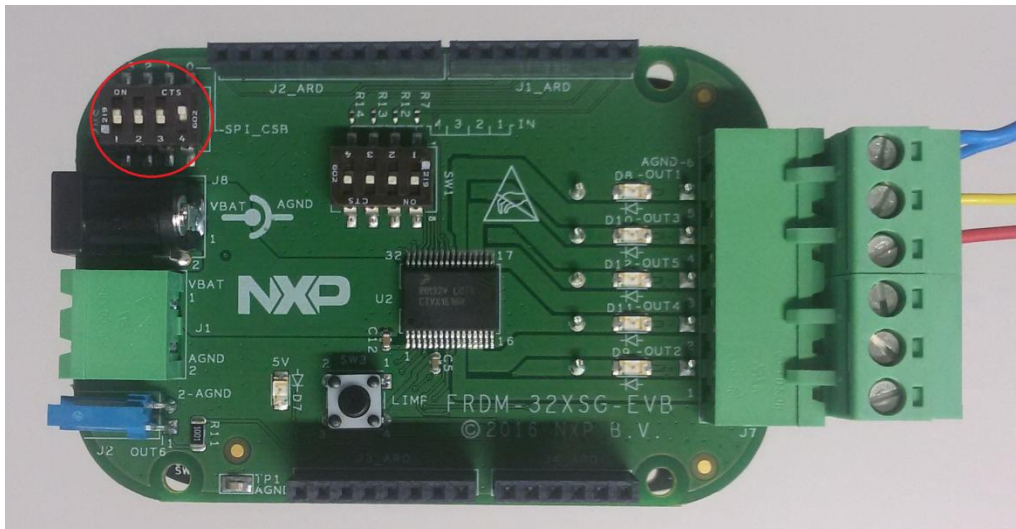

Figure 1 CSB Route Selection

In order to get the project successfully running you need to connect some load to channels 1 and 3 on eSwitch.
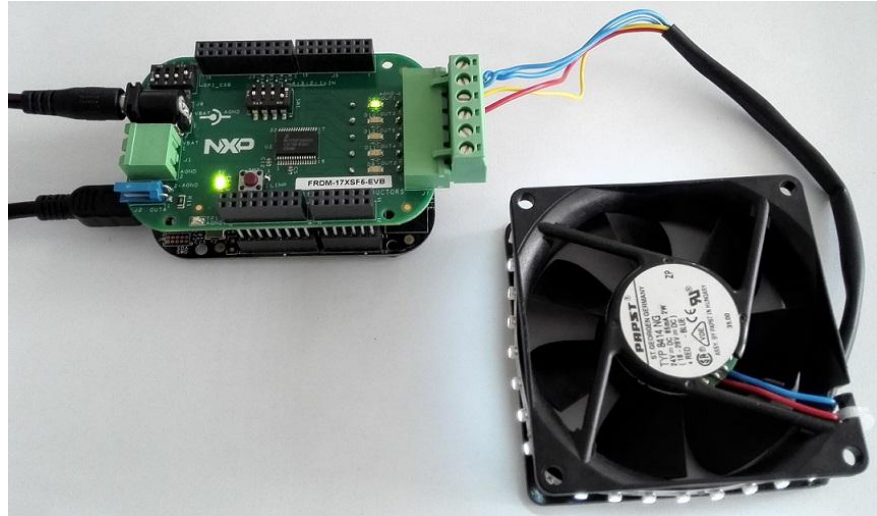
Figure 2 Channel Load

Set up OpenSDA connection between PC and MCU board with USB cable.

Figure 3 OpenSDA Connection

Setting up software

Make sure you have installed KDS 3.2.0 or newer which comes with Processor Expert support.

These components must be imported to the Processor Expert Component Library.

- **SPI_Device** component (encapsulates SPI communication)
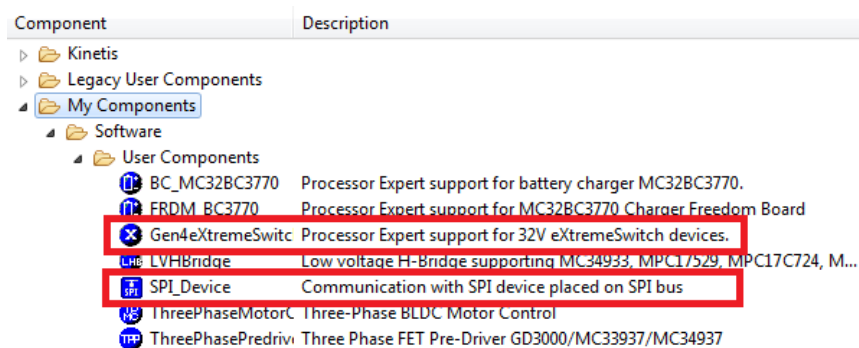- **Gen4eXtremeSwitch** component (encapsulates eSwitch configuration)

Figure 4 Processor Expert Component Library
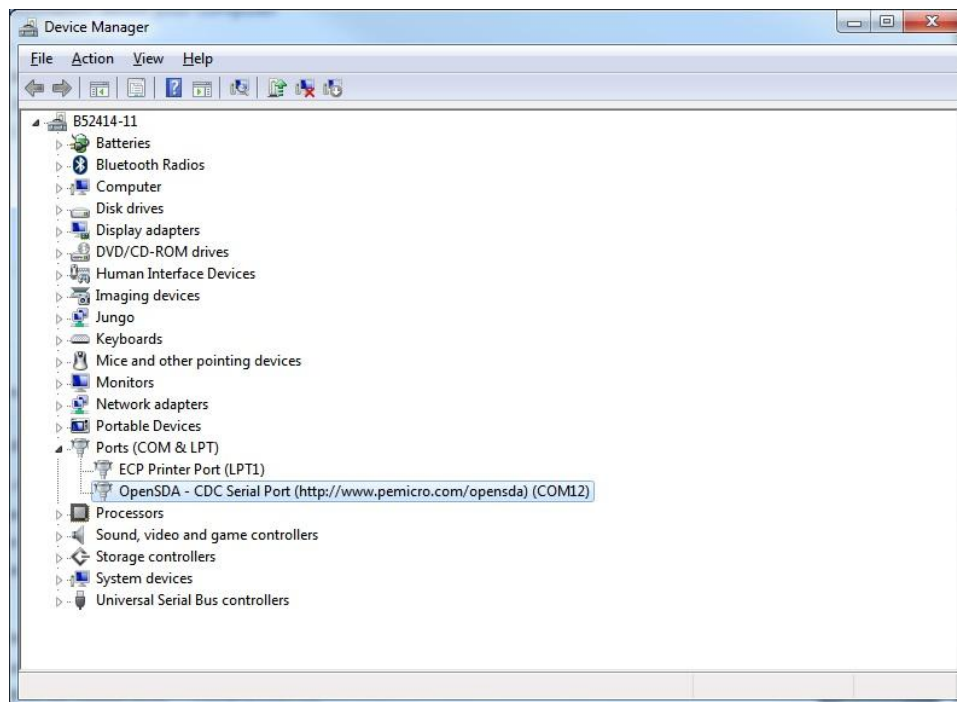
Check that your OpenSDA connection has been set up.

Figure 5 OpenSDA Virtual Port

Description

The example is preconfigured in the following way. FRDM-32XSG-EVB board is configured in **Gen4eXtremeSwitch** component, see Figure 7. SPI settings are configured separately in **SPIMaster_LDD** component, see Figure 8 and in inherited **CSpin1** component, see Figure 9. All components can be accessed in component tree, see Figure 6.
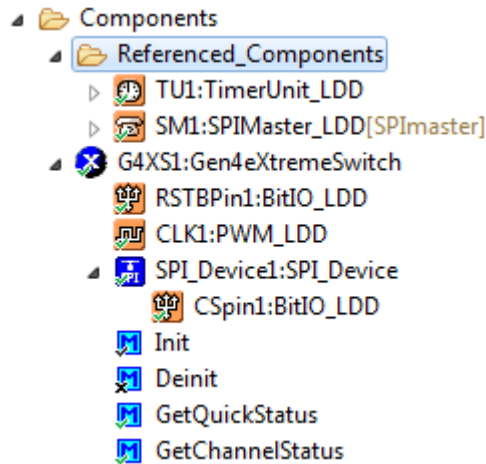
Figure 6 Component Tree

Channels 1 and 3 are enabled and configured in Gen4eXtremeSwitch component properties, see Figure 7. Both channels are enabled by default with PWM duty set to 32. This is dynamically changed later in the code. Watchdog timeout period is set to 128 ms. Make sure that if you modify the project, you still hold this watchdog period. All other features are set to default values except pin selection which follows **Error! Reference source not found.**.

Component Name	G4XS1	
SPI Configuration	Parallel SPI	
Global Configuration		
RSTB Pin	PTC12/TPM_CLKIN0	
External Clock Frequency	50000 [D]	
CLK Pin	PTC3/LLWU_P7/UART1_RX/TPM0_...	PTC3/LLWU_P7/UART1_RX/TPM0_...
Watchdog Timeout	128 ms	
Direct Input Control	Disabled	
Current, Voltage and Tempe	Disabled	
Devices	1	
⊿ **Device1**		
Device Model	MC17XSF500	
SOA Mode	Single read	
Overtemperature Warning	115 °C	
HID Selection	Disabled	
OCHI Type	Default	
Global PWM Duty Cycle	0 [D]	
⊿ **Channels**	6	
⊿ **Output1**	Enabled	Corresponds to channel with index...
⊿ **PWM Output Cont**		
Global PWM	Disabled	
Channel Duty C	32 [D]	
Phase Selection	0°	
Pulse Skipping	Disabled	
Slew Rate Presc	1	
Output Initial S	On	
Direct Input Contr	Disabled	
⊿ **Open Load**		
Open Load LED	Disabled	
OLON Deglitch	64 us	
⊿ **Overcurrent**		
OCLO Threshol	Low	
Advanced Curr	Disabled	
Short OCHI	Disabled	
No OCHI	Disabled	
Output2	Disabled	Channel disabled.
▷ **Output3**	Enabled	Corresponds to channel with index...
Output4	Disabled	Channel disabled.
Output5	Disabled	Channel disabled.
Output6	Disabled	Channel disabled.
Auto Initialization	yes	

Figure 7 G4XS Component Settings

SPI settings involve pin selection (MISO, MOSI, CLK and CSB). The first three pins are configured in **SPIMaster_LDD**, see Figure 8 and the last one is configured in **CSPin** component, see Figure 9, which is inherited by **SPI_Device** component.

	Device	SPI0
	Interrupt service/event	Enabled
◢	**Settings**	
◢	**Input pin**	Enabled
	Pin	PTD3/SPI0_MISO/UART2_TX/TPM0_CH3/SPI0_MOSI
◢	**Output pin**	Enabled
	Pin	PTD2/SPI0_MOSI/UART2_RX/TPM0_CH2/SPI0_MISO
◢	**Clock pin**	
	Pin	ADC0_SE5b/PTD1/SPI0_SCK/TPM0_CH1
	Chip select list	0
◢	**Attribute set list**	1
◢	**Attribute set 0**	
	Width	8 bits
	MSB first	yes
	Clock polarity	Low
	Clock phase	Change on leading edge
	Parity	None
	Chip select toggling	no
	Clock rate index	0
	Clock rate	1.497966 MHz
◢	**Initialization**	
	Auto initialization	yes

Figure 8 SPI Master Component Settings

Pin for I/O	PTD0/SPI0_PCS0/TPM0_CH0
Direction	Output
◢ Initialization	
Init. direction	Output
Init. value	1
Auto initialization	yes

Figure 9 Chip Select Pin Settings

Implementation itself is quite straightforward. In **main.c**, see Figure 10, is infinite main loop where all the logic of this example is situated.

1. At first duty value of both channels is incremented by 4 LSB until maximum duty value is reached.
2. After that duty value of both channels is decremented by 4 LSB until minimum duty value is reached.

```c
int main(void)
/*lint -restore Enable MISRA rule (6.3) checking. */
{
  /* Write your local variable definition here */
  G4XS1_TError Error;
  uint8_t LSBStep[2] = {ids4LSB, ids4LSB};

  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/

  for (;;) {
    while (1) { /* increments to maximum duty */
      Error = G4XS1_IncrementalPWMControl(0, idsPOSITIVE, LSBStep);
      if ((Error != ERR_OK) && (Error != ERR_MAX_DUTY)) {
        /* something went wrong */
      }
      for (uint8_t j = 0; j < 10; j++) { /* waits for 200 ms */
        WaitMS(20);
        G4XS1_FeedWatchdog();
      }
      if (Error == ERR_MAX_DUTY) {
        break;
      }
    }

    while (1) { /* decrements to minimum duty */
      Error = G4XS1_IncrementalPWMControl(0, idsNEGATIVE, LSBStep);
      if ((Error != ERR_OK) && (Error != ERR_MIN_DUTY)) {
        /* something went wrong */
      }
      for (uint8_t j = 0; j < 10; j++) { /* waits for 200 ms */
        WaitMS(20);
        G4XS1_FeedWatchdog();
      }
      if (Error == ERR_MIN_DUTY) {
        break;
      }
    }
  }
}
```

Figure 10 Content of main.c

Note that **FeedWatchdog()** has to be called periodically even when some delay is required.

Import the example project

Pre-requisites such as needed components are assumed to be imported already. If that is correct, you can approach to importing of the example project.

1. In KDS click on the *File / Import*.
2. Choose *General / Existing Projects into Workspace*.
3. Click *Browse to select root directory* with your downloaded example projects.
4. *Select project* named **G4XS_KL25Z_XSG_IncrementalPWM** and click *Finish* to complete process.
5. Now the example project should be copied to your workspace and ready to run.

Build and Debug

In order to build and run the project you need to *generate code* at first. Then you can *build* the project usual way. If the build is successful, *debug and run* the project.