Freescale Semiconductor, Inc.
Application Notes

Document Number: AN5209
Rev. 0, 10/2015

Generating PWM by Using FlexIO

1 Introduction

This application note illustrates how to use the FlexIO module to generate the pulse-width modulation (PWM) waveform.

FlexIO is a new on-chip peripheral available on some of the Kinetis series microcontrollers. It is highly configurable and capable of emulating a wide range of communication protocols, such as UART, I^{2}C, SPI, I^{2}S, and so on. You can use FlexIO to generate PWM waveform and build your own peripherals directly.

The standalone peripheral module FlexIO is used as an additional peripheral module of the microcontroller and is not a replacement of the PWM generator like TPM and FlexTimer peripherals. A key feature of FlexIO is that it enables you to build your own peripheral directly in the microcontroller.

This use case creates a simple software demo based on the SDK FlexIO HAL driver for you to easily use FlexIO as PWM with frequency and duty configurations.

Contents

2 Overview of the FlexIO module

The hardware resources in the FlexIO module are:

- Shifter
- Timer
- Pin

The amount of these resources for a given microcontroller is read from the FLEXIO_PARAM register. For example, there are four shifters, four timers, and eight pins in KL27Z64.

Figure 1 shows a high-level overview of the FlexIO module.

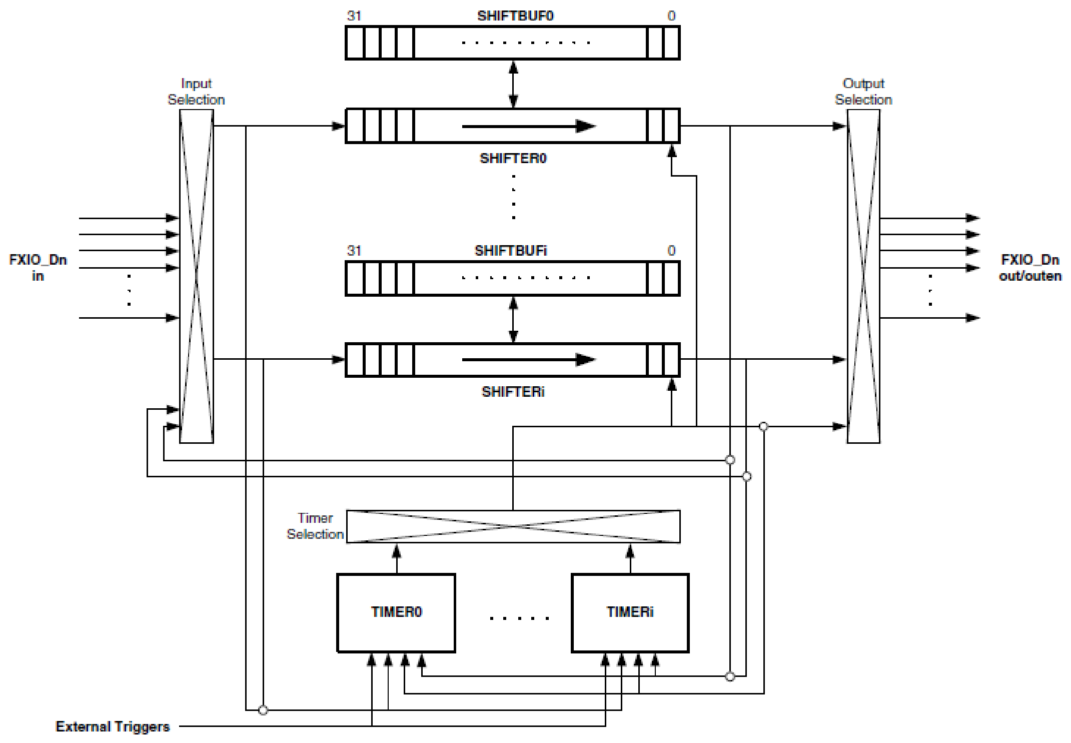

Figure 1. FlexIO block diagram

The key features are:

- 32-bit shifters with transmit, receive, and data match modes.
- Double buffered shifter operation.
- 16-bit timers with high flexibility support for a variety of internal or external triggers, and Reset/ Enable/Disable/ Decrement conditions.
- Automatic start/stop bit generation/check.
- Interrupt, DMA, or polling mode operation.
- Shifters, timers, pins, and triggers can be flexibly combined.

Transmit and receive are two basic modes of the shifters. If one shifter is configured to transmit mode, it loads data from its buffer register and shifts data out to its assigned pin bit by bit. If one shifter is configured to receive mode, it shifts data from its assigned pin and stores data in its buffer register. The load, store, and shift operations are all controlled by the shifter's assigned timer.

You can also configure the timers as different operation modes, including dual 8-bit counters baud/bit mode, dual 8-bit counters PWM mode, and single 16-bit counter mode.

3 Generating PWM using FlexIO

This section describes how to generate the PWM by FlexIO. For this application, the Freescale Freedom development board FRDM-KL27Z, in Figure 2 has been used.

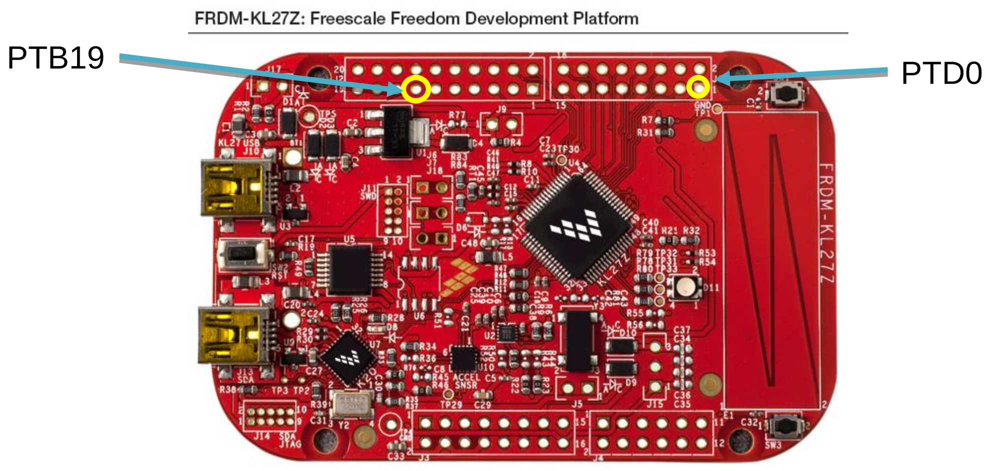

Figure 2. Freescale Freedom development board FRDM-KL27Z

In the application, FlexIO generates a PWM waveform on the FXIO0_D0 (PTD0) pin. Connect the PTD0 to PTB19 which drivers the GREEN LED (J1-1 connected with J2-13). Use PWM to lighten the GREEN LED with different duty to show the brightness change under different duty configurations.

3.1 General description

Generating PWM uses the following resources:

- One timer — configured as an 8bit PWM mode to control the related pin output.
- One pin — controlled by timer to toggle pin out and generate PWM.

Figure 3 shows the resource assignment.

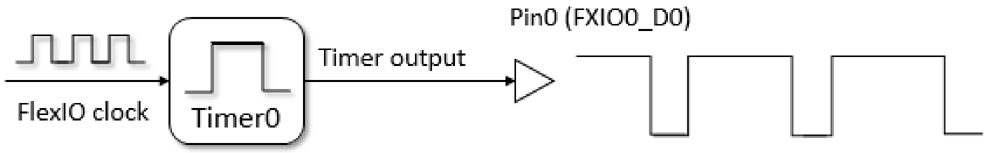

Figure 3. Resource Assignment of FlexIO to generate PWM

The detailed configurations and usage information is provided in the following sections.

3.2 Configurations of the timer

This section provides detailed configurations of the timer.

To understand these configurations, see the following descriptions and the reference manual.

Configurations for Timer 0

Timer 0 generates the PWM output to pin 0. It has the following initial configurations.

Table 1. Initial configuration of Timer 0

Items	Configurations
Timer mode	Dual 8-bit PWM mode
Trigger selection	N/A
Trigger polarity	N/A
Trigger source	Internal
Pin selection	Pin 0
Pin configuration	Output enabled
Pin polarity	Active high
Timer initial output	Output logic 1 when enabled, not affect by reset
Timer decrement source	Decrement on FlexIO clock, Shift on timer output
Timer enable condition	Always enabled
Timer disable condition	Never disabled
Timer reset condition	Never reset
Start bit	Disabled
Stop bit	Disabled
Timer compare value	(((FLEXIO_CLK / freq) * (100 - duty) / 100 - 1) << 8) | ((FLEXIO_CLK / freq) * duty / 100 - 1) (((FLEXIO_CLK*Low_Period)/2-1)<<8)|((FLEXIO_CLK*High_Period)/2-1) or ((((FLEXIO_CLK/freq)*(100-duty)/100)/2-1)<<8)|(((FLEXIO_CLK/freq)*duty/100)/2-1)

The following tips are the key points of using FlexIO to generate PWM.

- **Timer mode configuration (TIMCTL[TIMOD] and TIMCTL[TIMDEC])**
  - Select the dual 8-bit PWM mode. In this mode, the lower 8-bits of the counter only decrease when the timer output pin is high, and upper 8-bits only decrease when the timer output pin is low. When the lower 8-bits of the counter decreases to 0, the timer output would toggle and

lead the upper 8-bits decrease. The lower 8-bits control the PWM high pulse width and the upper 8-bits control the low pulse width.
  - Configure the FlexIO clock (FLEXIO_CLK[1]) as the timer counter decrease source. Therefore, the counter decreases on every FlexIO clock if decrease condition met.
- **Timer compare value (TIMCMP[CMP])**:
  - The timer compare value is loaded into the timer counter when the timer is first enabled, when the timer is reset or decreased to 0. Enter the dual 8-bits value into this compare register to load the value to ensure every cycle of the PWM has the timer counter reloaded from CMP. This compare value controls the PWM frequency and duty.
  - Timer compare value = (((FLEXIO_CLK / freq) * (100 - duty) / 100 – 1) << 8) | ((FLEXIO_CLK / freq[2]) * duty[3] / 100 - 1)
- **Timer output configuration (TIMCFG[TIMOUT])**
  - The timer output is set to logic one (high on pin) when enabled the timer is not affected by reset. In 8-bits PWM mode, you must set the timer output to 1 when the timer enabled, otherwise the lower 8-bits value of the counter would not decrease as mentioned above.

[1] FLEXIO_CLK is the clock from clock modules like SCG or MCG, used to control the FlexIO timing.

[2] The frequency of the PWM generated.

[3] The duty of the PWM waveform.

Figure 34 shows the timing and signal examples. For example, TIMCP=0x58.

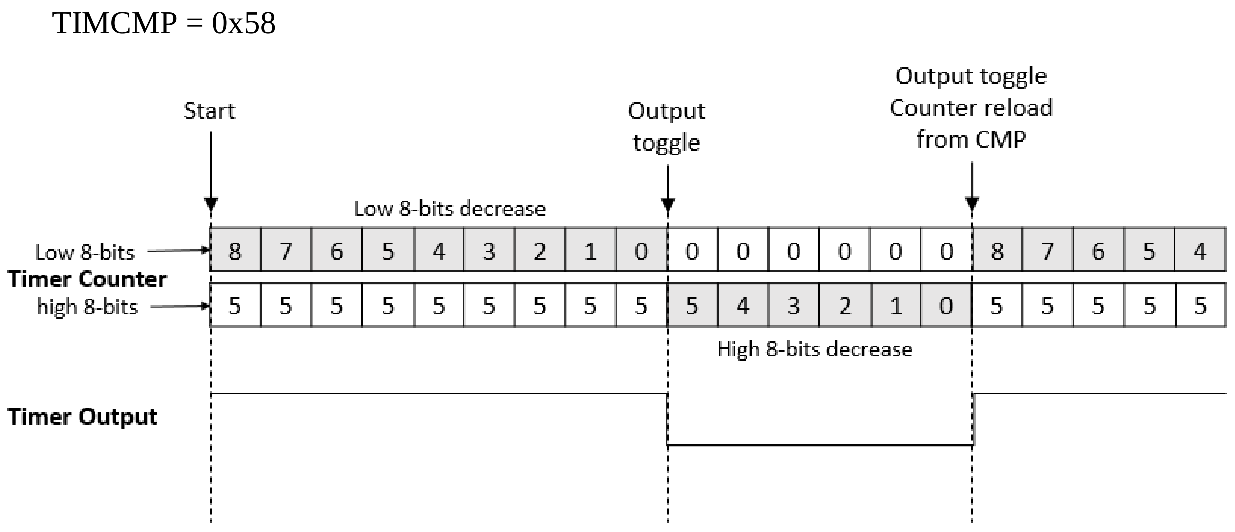

Figure 4. Timing of the configurations

3.3 Software implementation overview

This section describes the software implementation. Several driver functions have been implemented in this application, which are based on the HAL (Hardware Abstraction Layer) of the Freescale KSDK (Kinetis Software Development Kit).

A software package is provided along with this application note. The package contains a pruned KSDK based on V1.2, required by the application.

The demo software includes only one source file, main.c. This source file provides the following functions to configure the FlexIO as a PWM generator, all of them can be directly used by user in their own codes with minor changes:

- flexio_init(void)
- flexio_pwm_init(uint32_t freq, uint8_t duty)
- flexio_pwm_start(void)
- flexio_pwm_stop(void)

3.3.1 flexio_init()

flexio_init() function enables the clock gating of the FlexIO IP module and selects the proper peripheral clock source for FlexIO, for example, MCGIRC CLK. The FLEXIO_CLK defined in the source file is exactly the frequency of the peripheral clock source. It resets the FlexIO IP module by SW and re-enables it. This is a general FlexIO IP module initialization function, called before using its shifters and timers.

3.3.2 flexio_pwm_init()

flexio_pwm_init() function configures the timer0 as a 8-bits PWM mode with pin0 output to generate the PWM waveform. The timer detail configurations can be found in section 3.2. The ‘freq’ parameter of this function is the specified PWM frequency you want to generate. The parameter must be within the range of [MIN_FREQ, MAX_FREQ] macros defined in the source file. The ‘duty’ parameter is the specified duty in unit of %, with a range of [1, 99].

3.3.3 flexio_pwm_start()

flexio_pwm_start() function enables the timer0 by setting TIMOD to 8-bits PWM and start generating the PWM.

3.3.4 flexio_pwm_stop()

flexio_pwm_stop() function disables the timer0 by setting TIMOD and disable generating the PWM.

To use the FlexIO as PWM generator, you can call these three functions in sequence like in the main() function. It configures the PWM frequency to 8 KHz, and duty from 99 to 1, to change the brightness of the GREEN LED by PTD0.

4 Running the demos

You can download a program image to the microcontroller with Open-SDA. The PC host obtains a serial port after a USB cable is connected between the PC host and the Open-SDA USB socket (J13) on FRDM-KL27Z.

The project and workspace files of the demo are located in:

/examples/frdmkl27z/demo_apps/flexio_pwm/iar

The source file is located in:

/examples/frdmkl27z/demo_apps/flexio_pwm/src

Open the workspace file *.eww*, and then build the demo project. Download and run the demo. The user can look at the GREEN LED with brightness changing from min to max. The PWM signal can also be captured with an oscilloscope on the PTD0 pin, this shows the exact frequency and duty.

5 Conclusion

FlexIO is a new peripheral on some of the Kinetis microcontrollers. Thanks to the high flexibility of the shifters and timers, FlexIO has the capability to emulate a wide range of protocols and generate PWM.

This application note describes how to generate the PWM using FlexIO. The application is based on KSDK HAL. Although the demo runs on FRDM-KL27Z, the user can readily port them to other parts of Kinetis with FlexIO.

However, there are also some limitations when using FlexIO to generate PWM. For example, the maximum and minimum frequency range which depends highly on the FlexIO clock source. Also it can only support Edge-Aligned PWM; Center-Aligned PWM is not supported.

6 Additional information

Currently, the FlexIO module can be found on the following Kinetis series MCUs: KL17, K27, KL33, and KL43. In addition, the FlexIO modules on those platforms have same hardware resources and capabilities.

The next version of FlexIO is in development and will be launched with new Kinetis parts. The next version has more abundant hardware resources and more powerful capabilities. Some of the key features of the new version are listed below.

- Up to 8 shifters.
- Up to 8 timers.
- Up to 32 pins.
- Parallel data transmission is supported. This enables the emulations of camera interface, Motorola 68K and Intel 8080 bus, and so on.
- Programmable logic blocks allowing external digital logic to be integrated on-chip.
- Programmable state machine for offloading basic system control functions from CPU.

7 References

1. FRDM-KL27Z: Freescale Freedom Development Board for Kinetis KL17 and KL27 MCUs
   http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=FRDM-KL27Z
2. KINETIS_SDK: Software Development Kit for Kinetis MCUs
   http://www.freescale.com/ksdk

8 Revision history

Table 2. Revision history

Revision Number	Date	Substantive changes
0	10/2015	Initial release

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AN5209
Rev. 0
10/2015