AN13459

Generating PWM by Using FlexIO on KE17Z

Rev. 0 — 18 November 2021 Application Note

Contents

1 Introduction

This application note illustrates how to use the FlexIO module to generate the Pulse-Width Modulation (PWM) waveform.

FlexIO is an on-chip peripheral available on NXP Kinetis series microcontrollers. It is highly configurable and capable of emulating a wide range of communication protocols, such as UART, I^2C, SPI, I^2S. You can use FlexIO to generate PWM waveform and build your own peripheral 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, such as, FlexTimer (FTM) peripheral. A key feature of FlexIO is that it enables you to build your own peripheral directly in the microcontroller.

This application note takes the `flexio_pwm` project in KE17Z SDK 2.10.1 as an example to explain how to implement the function of generating PWM by using FlexIO. The download link of KE17Z SDK is https://mcuxpresso.nxp.com/en/select.

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 KE17Z.

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

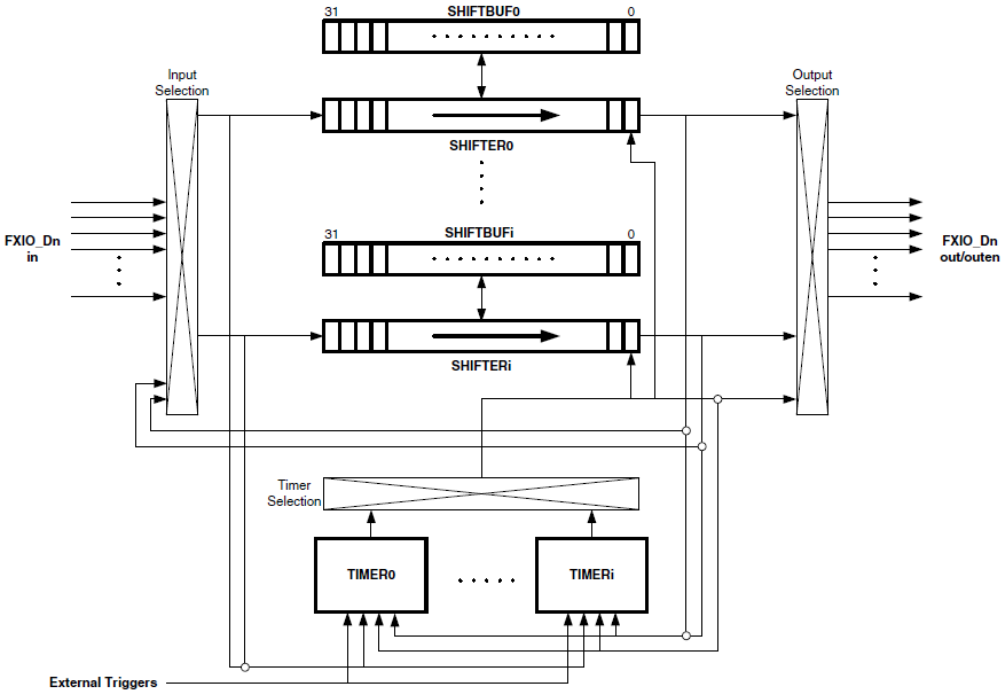

Figure 1. FlexIO block diagram

The key features are:

- Array of 32-bit shift registers with transmit, receive, and data match modes.
- Double buffered shifter operation for continuous data transfer.
- Shifter concatenation to support large transfer sizes.
- Automatic start/stop bit generation.
- Interrupt, DMA, or polled transmit/receive operation.
- Programmable baud rates independent of bus clock frequency, with support for asynchronous operation during stop modes.
- Highly flexible 16-bit timers with support for various internal or external triggers, reset, enable, and disable conditions.

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 shifters assigned timer controls the load, store, and shift operations.

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 NXP Freedom development board FRDM-KE17Z, as shown in Figure 2, is used.

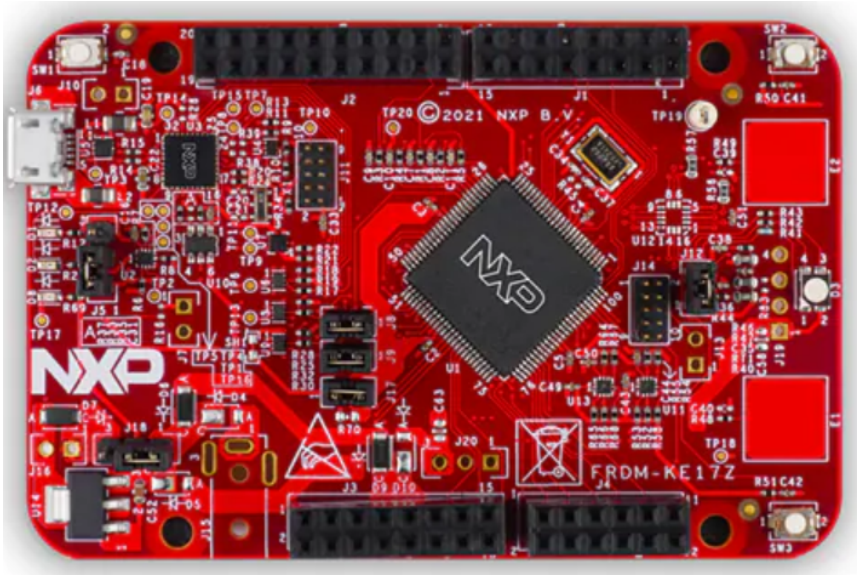

Figure 2. FRDM-KE17Z board

In the application, FlexIO generates a PWM waveform on the `FXIOO_D3` (PTD5) pin. Use an oscilloscope to capture the waveform of PTD5 (J3-1), and a PWM wave with a frequency of 48 K and a continuously changing duty cycle is generated.

3.1 General description

To generate PWM, use the following resources:

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

Figure 3 shows the resource assignment.

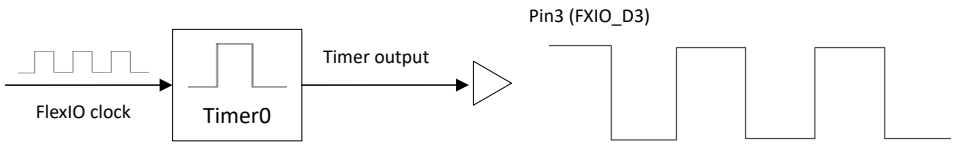

Figure 3. Resource assignment of FlexIO to generate PWM

The following sections describe the detailed configurations and usages.

3.2 Timer configurations

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 3. 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 3
Pin configuration	Output enabled
Pin polarity	Active high
Timer initial output	Output logic 1 when enabled, not affected 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 comparison value	sum = ((FLEXIO_CLK * 2)/freq + 1) /2;
	lowerValue = (sum * duty/50 +1)/2;
	upperValue = sum – lowerValue;
	Timer comparison value = (upperValue -1) << 8 | (lowerValue -1);

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

- **Timer mode configuration (TIMCTL[TIMOD] and TIMCFG[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 decrease to 0, the timer output toggles and leads 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 comparison value (TIMCMP[CMP])**

  The timer comparison 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-bit values into this comparison register. Then, every cycle of the PWM has the timer counter reloaded from CMP. This comparison value controls the PWM frequency and duty.

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

sum = ((FLEXIO_CLK * 2)/freq + 1) /2;

lowerValue = (sum * duty/50 +1)/2;

upperValue = sum – lowerValue;

Timer comparison value = (upperValue -1) << 8 | (lowerValue -1);

- **Timer output configuration (TIMCFG[TIMOUT])**
  Set the timer output to the logic one (high on pin) when the timer is enabled and not affected by timer reset. In the 8-bit PWM mode, set the timer output to **1** when the timer is enabled. Otherwise, the lower 8-bit values of the counter do not decrease as mentioned above.

Figure 4 shows the timing and signal examples. For example, TIMCP = 0x58.

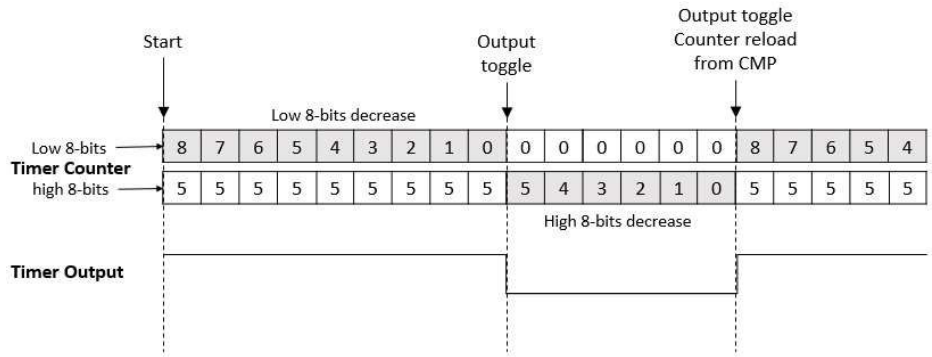

Figure 4. Timing of the configurations

3.3 Software implement overview

This section describes the software implementation. FRDM-KE17Z SDK 2.10.1 provides a `flexio_pwm` example. This example includes only one source file, `flexion_pwm.c`. This source file provides the following functions to configure the FlexIO as a PWM generator. Users can use all the functions directly in their own code with minor changes.

- `FLEXIO_Init` (`void`)
- `flexio_pwm_init` (`uint32_t freq`, `uint8_t duty`)
- `flexio_pwm_start`(`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. `FLEXIO_Init()` 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 **timer0** as an 8-bit PWM mode with pin3 output to generate the PWM waveform. For detailed timer configurations, see Timer configurations. The `freq` parameter of this function is the specified PWM frequency to generate. The parameter must be within the range of [`FLEXIO_MIN_FREQUENCY`, `FLEXIO_MAX_FREQUENCY`] 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 **timer0** by setting **TIMOD** to 8-bit PWM and starting to generate the PWM.

3.4 Frequency range of PWM simulated by FlexIO

The frequency of FlexIO module determines the frequency range of the PWM simulated by FlexIO. The calculation of the highest frequency and the lowest frequency is as below.

```
#define FLEXIO_MAX_FREQUENCY (DEMO_FLEXIO_CLOCK_FREQUENCY / 2U)
#define FLEXIO_MIN_FREQUENCY (DEMO_FLEXIO_CLOCK_FREQUENCY / 512U)
```

The macro `DEMO_FLEXIO_CLOCK_FRENQUENCY` is the clock of the FlexIO module. The clock source of the FlexIO module can be `FLLDIV2_CLK`, `SIRCDIV2_CLK`, `FIRCDIV2_CLK`, and `SOSCDIV2_CLK`. Among them, **FLLDIV2_CLK** is at most 72 MHz, so the highest PWM frequency simulated by FlexIO is 72 M/2 = 36 M, `SOSCDIV2_CLK` is at least 500 Hz (32000/64), and the lowest PWM frequency simulated by FlexIO is 500/512 = 0.97 Hz. For different KE17Z boards, the lowest frequency of the PWM generated by FlexIO depends on the frequency of the external crystal oscillator.

On the FRDM-KZ17 board, the frequency of the external crystal oscillator is 8 MHz, the minimum clock of `SOSCDIV2_CLK` is 125000 Hz (8 M/64), and the lowest PWM frequency simulated by FlexIO is 125000/512 = 244 Hz.

3.5 Characteristics of PWM simulated by FlexIO

Compared with the PWM generated by the FTM module, each PWM simulated by FlexIO has an independent frequency and duty cycle. Multiple channels PWMs generated by the FTM module share the frequency.

The PWM simulated by FlexIO also has some shortcomings. For example, the complete FTM module supports deadtime insertion and fault control functions, but the FlexIO simulated PWM does not support these functions and the center-aligned mode.

4 Running the demo

Download a program image to the microcontroller with Open-SDA. The PC host recognizes a virtual serial port after a USB cable is connected between the PC host and the Open-SDA USB socket, **J6**, on FRDM-KE17Z.

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

```
/boards/frdmke17z/driver_examples/flexion/pwm/iar
```

The source file is located in:

```
/boards/frdmke17z/driver_examples/flexion/pwm/
```

Open the workspace file, *.eww*, build the demo project, download the demo, and run it. The PWM signal can be captured with an oscilloscope on the **PTD5** pin. The exact frequency and duty are as shown in Figure 5 and Figure 6.

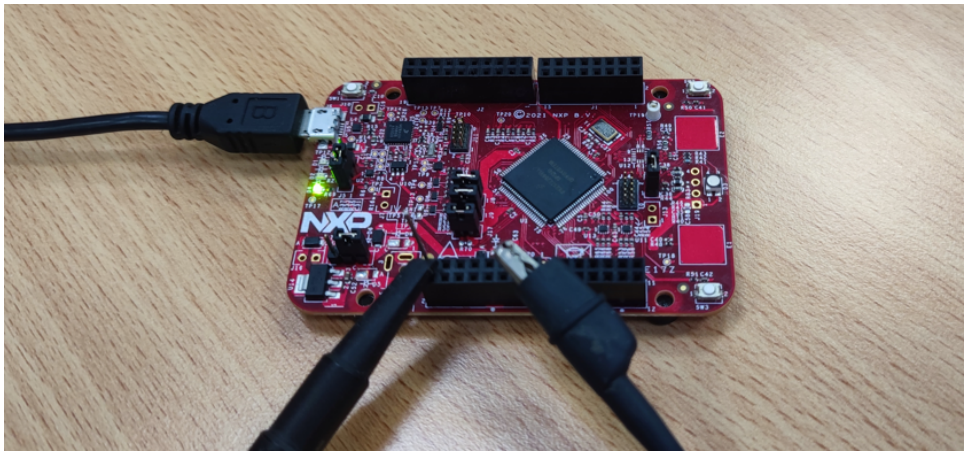

Figure 5. FRDM-KE17Z board

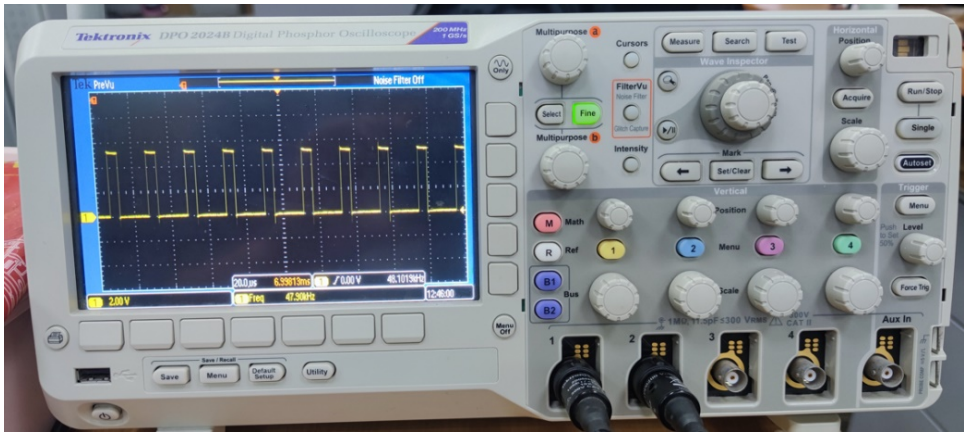

Figure 6. 48 K PWM wave generated by FlexIO module

5 Conclusion

With the high flexibility of the shifters and timers, FlexIO can 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 NXP SDK. Although the demo runs on FRDM-KE17Z, the user can quickly port them to other parts of Kinetis with FlexIO.

6 References

1. *Generating PWM by Using FlexIO* (document AN5209)
2. *Generating PWM and PFM by Using FlexIO* (document AN12713)
3. *Kinetis KE17Z/13Z/12Z with up to 256 KB Flash Reference Manual* (document KE1xZP100M72SF1RM)

7 Revision history

Rev.	Date	Description
0	18 November 2021	Initial release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Right to make changes - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 18 November 2021

Document identifier: AN13459

arm