NXP Semiconductors
Application Note

Document Number: AN12313
Rev. 0, 12/2018

KE16Z PDB Trigger ADC Conversion in Back-to-Back Mode

Contents

1. Introduction

Programmable Delay Block (PDB) provides controllable delays from either an internal or an external trigger, or a programmable interval tick, to the hardware trigger inputs of ADCs, so that the precise timing between ADC conversions can be achieved. PDB module supports one configurable channel for ADC hardware trigger, which is associated with one ADC module. For each PDB channel, there is one trigger output for ADC hardware trigger to start an analog-to-digital conversion, and up to four pre-trigger outputs to ADC module to select different ADC channels prior to ADC conversion.

PDB back-to-back operation enables the ADC conversions complete signal (the COCO flag) to trigger the next PDB channel pre-trigger and trigger output automatically without any additional external trigger input to PDB module. In some real-time applications, like motor control, it's necessary to sample two ADC channels almost at the same time to get the correct analog value. For some MCUs with more than one ADC modules, it's not a problem, since each ADC module can sample one ADC input, the two ADC modules sample the two analog inputs in parallel. While KE16Z only has one ADC module, it has to implement analog-to-digital conversion one by one. As a result, the interval between the two ADC conversions becomes critical. In this case, PDB back-to-back mode is a

good solution, as it can start another ADC conversion immediately right after the previous conversion complete.
This application note explains the inter-connections of PDB and ADC, provides an example to help user understand the configurations of PDB and ADC for back-to-back mode. The implementation of the example is based on *IAR embedded Workbench 8.30.1* development environment, *SDK 2.4.0* software, KE16Z board.

2. Features Usage

2.1 Overview

The PDB module is a timer that provides controllable delays to the hardware trigger inputs of ADCs, in order to the precise timing between ADC conversion. The PDB uses either an internal or external trigger input in order to start counting. In this use case, the PDB trigger source selection is implemented through the TRGMUX module. FTM channel match trigger or init trigger is used as the trigger input source of PDB and one PDB channel is associated with one ADC module. The PDB channel supports 4 pre-triggers, which can be used to select different ADC channels. The modules overview is shown in Figure 1.

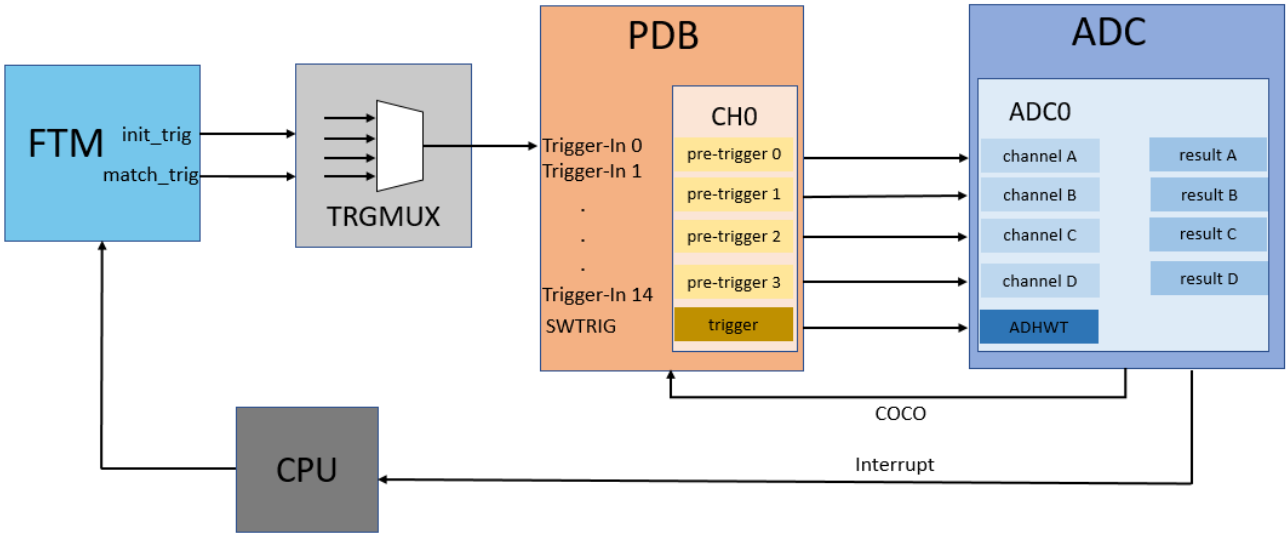

Figure 1. **The modules overview**

2.2 Back-to-back acknowledgement connections

The PDB back-to-back mode acknowledgement connections are implemented inside PDB unit as a ring. The back-to-back mode needs to receive an ADC conversion complete flag to trigger the next PDB pre-trigger output. Once PDB get a trigger input signal, the first pre-trigger output is generated after a predefined delay time. The following PDB back-to-back operation acknowledgment connections are implemented based on *PDB0_CH0C1_BB* bits setting.

When *BB*[0]=1, PDB0 channel 0 pre-trigger 0 acknowledgement input: *ADC0SC1D_COCO*; When *BB*[1]=1, PDB0 channel 0 pre-trigger 1 acknowledgement input: *ADC0SC1A_COCO*; When *BB*[2]=1, PDB0 channel 0 pre-trigger 2 acknowledgement input: *ADC0SC1B_COCO*; When *BB*[3]=1, PDB0 channel 0 pre-trigger 3 acknowledgement input: *ADC0SC1C_COCO*.

In the use case, FTM0 generates a periodic interrupt trigger signal to start PDB0,so *PDB0_CH0C1_BB* is written with 0xE. The mapping between back-to-back enable bits and pre-triggers is shown in Table 1. The inter-connectivity diagram in back-to-back mode is shown in Figure 2.

Table 1. **Mapping between enable bits and pre-triggers**

Back-to-Back Enable bit	PDB Channel Pre-Trigger
BB[0]	pre-trigger 0
BB[1]	pre-trigger 1
BB[2]	pre-trigger 2
BB[3]	pre-trigger 3

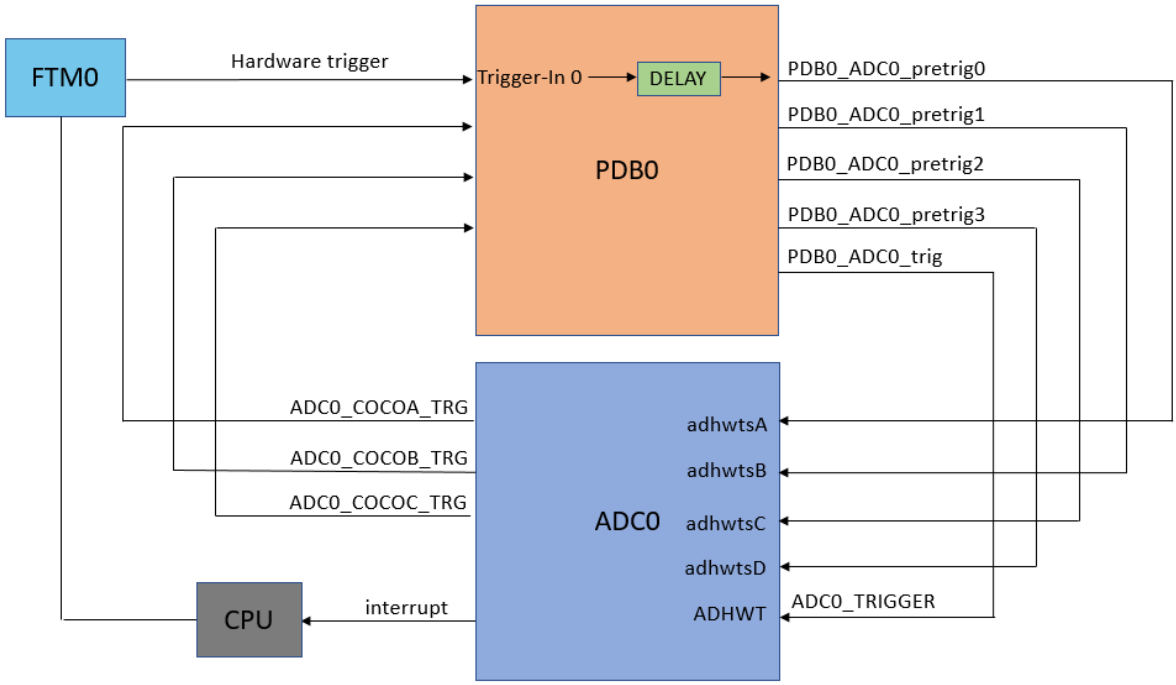

Figure 2. **Inter-connectivity diagram in back-to-back mode**

2.3 Timing of PDB pre-trigger and trigger outputs

KE16Z MCU has one PDB module and one ADC module only. PDB0 channel 0 is associated with ADC0 block, and the PDB0 channel 0 has four pre-trigger outputs. Each pre-trigger output and trigger output are connected to the ADC hardware inputs. The pre-triggers are used to precondition the ADC block before the actual trigger occurs. When the ADC receives the rising edge of the trigger, the ADC will start the conversion according to the precondition determined by the pre-triggers.

The ADC module has 4 sets of configuration and result registers, for example, *ADC_SC1A* and *ADC_RA*. So that the ADC can alternate conversions between four different analog sources. The PDB pre-trigger outputs are used to specify which signal will be sampled next. When a pre-trigger *m* is asserted, the ADC conversion is triggered with set *m* of the configuration and result registers.

The timing diagram in Figure 3 shows the pre-trigger and trigger outputs of PDB0 to ADC0 trigger by setting *PDB0_CH0C1_BB* to 0xE.

The following steps occur in the process of PDB0 trigger ADC0 conversion in back-to-back mode.

1. FTM0 generates periodic channel match trigger to start PDB0.
2. When external trigger input to PDB0, it will generate the pre-trigger 0 after a defined delay, which depends on *PDB0_CH0DLY0_DLY*. The pre-trigger 0 output can trigger one ADC0 channel conversion.
3. *BB*[1] is set to 1, ADC completed flag *COCOA* will trigger PDB0 pre-trigger 1. The pre-trigger 1 output can trigger one ADC0 channel conversion.
4. *BB*[2] is set to 1, ADC completed flag *COCOB* will trigger PDB0 pre-trigger 2. The pre-trigger 2 output can trigger one ADC0 channel conversion.
5. *BB*[3] is set to 1, ADC completed flag *COCOC* will trigger PDB0 pre-trigger 3. The pre-trigger 3 output can trigger one ADC0 channel conversion.
6. After the period of FTM0, FTM0 generates another trigger input to PDB0, to start another ADC0 conversion.

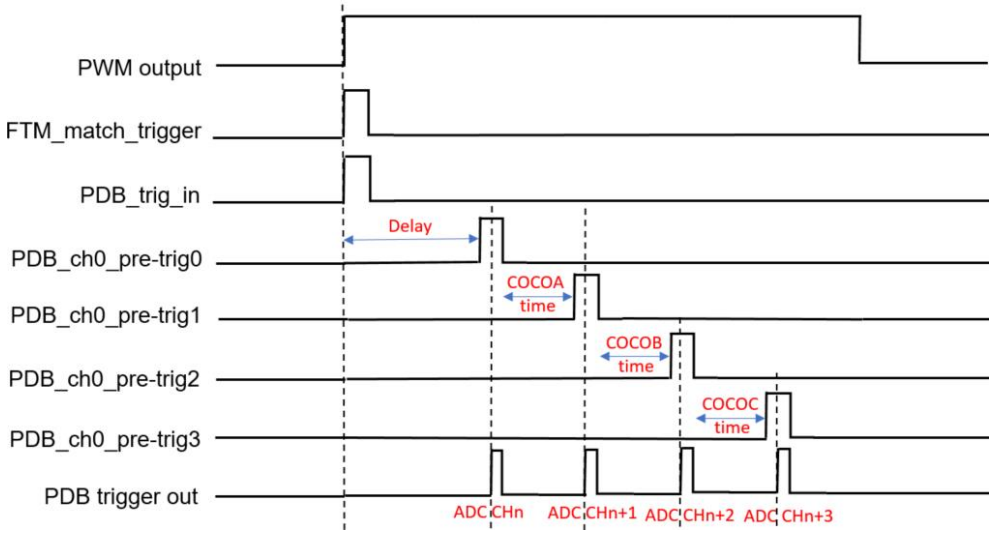

Figure 3. **Timing of PDB trigger and pre-trigger outputs to ADC**

3 Implementation Details

There is a use case for implementation details of PDB trigger ADC conversion in back-to-back mode. In the use case, FTM0, PDB0, ADC0 three peripherals must be configured. FTM0 generates an periodic external trigger to start PDB0 channel 0 trigger. Then PDB0 trigger four ADC channel alternate conversions in back-to-back mode.

3.1 FTM configuration

FTM0 module is set up to generate channel match trigger. KE16Z has a TRGMUX that provides a module interconnectivity scheme. The FTM0 module triggers are external triggers and connect to one input trigger of TRGMUX in the chip design. It is necessary to configure TRGMUX before FTM0 configuration.

```
/* select FTM0 hardware trigger as PDB0 trigger input */
TRGMUX_SetTriggerSource(TRGMUX0, kTRGMUX_Pdb0, kTRGMUX_TriggerInput0, kTRGMUX_SourceFtm0);
```

FTM0 hardware trigger is selected as the PDB input trigger by configuring the TRGMUX. The trigger output signal of FTM0 is connect to Trigger-In 0 of PDB0.

FTM0 is initialized to output edge-aligned PWM signal, the frequency of PWM is configured as 100 Hz. Select FTM0 channel 0 to generate the external trigger by setting *FTM0_EXTTRIG_CH0TRIG* to 1.

The initialization code for reference is as follows:

```
/* Initialize FTM module. */
FTM_GetDefaultConfig(&ftmConfigStruct);
ftmConfigStruct.extTriggers = kFTM_Chnl0Trigger; /* Enable to output the trigger. */
FTM_Init(DEMO_FTM_BASE, &ftmConfigStruct);


/* Configure ftm params with frequency 100Hz */
pwmParam.chnlNumber = kFTM_Chnl_0;
pwmParam.level = pwmLevel;
pwmParam.dutyCyclePercent = 50U; /* Percent: 0 - 100. */
pwmParam.firstEdgeDelayPercent = 0U;
FTM_SetupPwm(DEMO_FTM_BASE, &pwmParam, 1U, kFTM_EdgeAlignedPwm, DEMO_FTM_PWM_HZ,
 DEMO_FTM_COUNTER_CLOCK_HZ);
```

3.2 ADC configuration

PDB0 trigger four ADC0 channel alternate conversions in back-to-back mode. In addition to the ADC0 clock select, arbitration, and some basic initialization configurations, there are a few points need to pay attention in the use case.

Set *ADC0_SC2_ADTRG* to 1 to enable PDB0 hardware trigger for ADC0. ADC0 contains four sets of configuration and result registers: A, B, C, D, set *ADC0_SC1m_ADCH* to channel number to select the external channels as ADC0 input and *ADC0_SC1D_AIEN* to 1 to enable conversion complete interrupt.

```
/* Enable hardware trigger mode. */
ADC12_EnableHardwareTrigger(DEMO_ADC_BASE, true);


/* Configure the ADC12 conversion channels and interrupt. */
adc12ChannelConfigStruct.enableInterruptOnConversionCompleted = false;
adc12ChannelConfigStruct.channelNumber = DEMO_ADC_USER_CHANNEL;
ADC12_SetChannelConfig(DEMO_ADC_BASE, DEMO_ADC_CHANNEL_GROUP, &adc12ChannelConfigStruct);

adc12ChannelConfigStruct.enableInterruptOnConversionCompleted = false;
adc12ChannelConfigStruct.channelNumber = DEMO_ADC_USER_CHANNEL+26;
ADC12_SetChannelConfig(DEMO_ADC_BASE, DEMO_ADC_CHANNEL_GROUP+27, &adc12ChannelConfigStruct);

adc12ChannelConfigStruct.enableInterruptOnConversionCompleted = false;
adc12ChannelConfigStruct.channelNumber = DEMO_ADC_USER_CHANNEL+29;
ADC12_SetChannelConfig(DEMO_ADC_BASE, DEMO_ADC_CHANNEL_GROUP+2, &adc12ChannelConfigStruct);

adc12ChannelConfigStruct.enableInterruptOnConversionCompleted = true;
adc12ChannelConfigStruct.channelNumber = DEMO_ADC_USER_CHANNEL+30;
ADC12_SetChannelConfig(DEMO_ADC_BASE, DEMO_ADC_CHANNEL_GROUP+3, &adc12ChannelConfigStruct);
```

3.3 PDB configuration

The PDB0 generates pre-triggers and trigger output to the ADC0. In order to make PDB0 work properly in back-to-back mode, there are a few points need to note when configuring it. The configuration details about PDB0 will be given in the following content:

1. Clock select

The only clock source of PDB module is system clock. Make sure the system clock is enable for PDB0 by setting *PCC_PDB0_CGC* to 1.

```
/* Enable PDB0 clock. */
CLOCK_EnableClock(kCLOCK_Pdb0);
```

2. Enable and select the trigger source for PDB0

Some features of PDB0 are configured by *PDB0_SC* register. The PDB0 internal registers are loaded with values from their buffers when the trigger input event is detected after *PDB0_SC_LDOK* is set to 1. So this bit can be written in the last to update the *MOD*, *IDLY* and other registers with the values previously written to their internal buffer.

Set the *PDB0_SC_PDBEN* to 1 to enable the PDB function. PDB0 contains a counter whose output is compared to several different values, the counter will reset and start when PDB0 is enabled.

Set *PDB0_SC_LDMOD* to 0, it can load values from buffers into the internal registers immediately after 1 is written to *PDB0_SC_LDOK*.

Set *PDB0 _SC_PRESCALER* and *PDB0_SC_MUTL* to appropriate values to divider for the counter clock, they are used to calculate the delay time.

PDB0 channel 0 has up to 15 trigger input sources, set *PDB0_SC_TRGSEL* to 0 to select Trigger-In 0 as the trigger input source.

```
/* Initialize PDB module. */
PDB_GetDefaultConfig(&pdbConfigStruct);
/* The trigger would be selected by TRGMUX. */
pdbConfigStruct.triggerInputSource = kPDB_TriggerInput0;
PDB_Init(DEMO_PDB_BASE, &pdbConfigStruct);
```

3. PDB0 modulus and channel delay time configuration

The mod of PDB is not used but must be set up to an appropriate value. It should be more than delay value. The delay value determines the delay time of pre-trigger. Set the mod (*PDB0_MOD_MOD*) to 2000 and set channel 0 pre-trigger 0 delay value (*PDB0_CH0DLY0_DLY*) to 500 in the use case. It does not need to configure delay value for other pre-triggers in back-to-back mode. The channel delay time for pre-trigger is determined by clock, prescale, multiplication factor and delay value.

```
/* Configure the PDB mod. */
PDB_SetModulusValue(DEMO_PDB_BASE, DEMO_PDB_MODULO_VALUE);
/* Configure the PDB pre-Trigger 0 delay time. */
PDB_SetADCPreTriggerDelayValue(DEMO_PDB_BASE, DEMO_PDB_TRIGGER_CHANNEL,
DEMO_PDB_PRETRIGGER_CHANNEL, DEMO_PDB_PRETRIGGER_DELAY_VALUE);
```

4. Back-to-back mode configuration

Set *PDB0_CH0C1_EN* to 0xF to enable PDB0 channel 0 four pre-triggers.

Set *PDB0_CH0C1_TOS*[0] to 1, pre-trigger 0 asserts when the counter reaches the channel delay register plus one peripheral clock cycle after a rising edge is detected on selected trigger input source. Other three pre-triggers are in bypassed mode.

Set *PDB0_CH0C1_BB* to 0xE, the pre-trigger 1, pre-trigger 2 and pre-trigger 3 are enabled in back-to-back mode. When the first ADC conversion complete flag *COCOA* is set up, it can trigger next pre-trigger and the trigger output can trigger the next ADC conversion.

```
/* Configure the ADC Pre-Trigger. */
  PDB_SetADCPreTriggerDelayValue(DEMO_PDB_BASE, DEMO_PDB_TRIGGER_CHANNEL,
DEMO_PDB_PRETRIGGER_CHANNEL, DEMO_PDB_PRETRIGGER_DELAY_VALUE);


  /* PDB Channel Back-to-back Enable / PDB Channel Pre-Trigger Enable / PDB Channel Pre-Trigger Output Select */
  pdbAdcPreTriggerConfigStruct.enablePreTriggerMask = (1U << DEMO_PDB_PRETRIGGER_CHANNEL)|(1U <<
DEMO_PDB_PRETRIGGER_CHANNEL+1)|(1U << DEMO_PDB_PRETRIGGER_CHANNEL+2)|(1U <<
DEMO_PDB_PRETRIGGER_CHANNEL+3);
  pdbAdcPreTriggerConfigStruct.enableOutputMask = (1U << DEMO_PDB_PRETRIGGER_CHANNEL);
  pdbAdcPreTriggerConfigStruct.enableBackToBackOperationMask = (1U << DEMO_PDB_PRETRIGGER_CHANNEL+1)
|(1U << DEMO_PDB_PRETRIGGER_CHANNEL+2)|(1U << DEMO_PDB_PRETRIGGER_CHANNEL+3);
  PDB_SetADCPreTriggerConfig(DEMO_PDB_BASE, DEMO_PDB_TRIGGER_CHANNEL,
&pdbAdcPreTriggerConfigStruct);
```

5. Load OK

Modulus register, channel 0 delay 0 register and others are internal buffered, any values written to the register are written to its internal buffer instead. The values in this register's internal buffers are loaded into this register only after "1" is written to the *PDB0_SC_LDOK* bit.

```
 /* Load PDB counter register. */
  PDB_DoLoadValues(DEMO_PDB_BASE);
```

4 Conclusion

This application note explains the inter-connections of FTM, PDB and ADC, introduces how to use PDB back-to-back mode to implement periodic trigger four ADC channel conversions. Users can easily implement the function by reference to details of configuration.

5 Reference

Following references are available on NXP website:

1. KE16Z Reference Manual (Document: KE1xZP48M48SF0RM)
2. Using FTM, PDB, and ADC on KE1xF to Drive Dual PMSM FOC and PFC (AN5380)
3. Synchronize Analog Modules and Timers with PDB Modules in MC9S08MP16 (AN4424)
4. Tips and Tricks Using PDB in Motor Control Applications on Kinetis (AN4822)
5. PDB Driver for the MC9S08GW64 (AN4163)

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: AN12313
Rev. 0
12/2018