Freescale Semiconductor
Application Note

Document Number: AN4822
Rev. 0, 12/2013

Tips and Tricks Using PDB in Motor Control Applications on Kinetis

by *Ivan Lovas*
Freescale Semiconductor, Inc.

Contents

1 Introduction

The 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 the ADCs. This functionality is required for applications when precise timing of ADC conversions are required, or when the measured signal must be filtered using precise sampling.

The PDB is necessary in some motor control specific applications which require advanced ADC to PWM synchronization. This application note describes the set-up of the PDB, routing of the signal onto a GPIO pin, and sequence error handling. A typical application is the BLDC six-step method. This application note focuses on the Kinetis K60 family, the FlexTimer, and 16-bit ADCs.

2 Understanding PDB terminology

This section describes the important terms used in the Kinetis K60 reference manual that relate to this application.

Figure 1 illustrates the arrangement of the FlexTimer to the ADCs in association to the terms used.

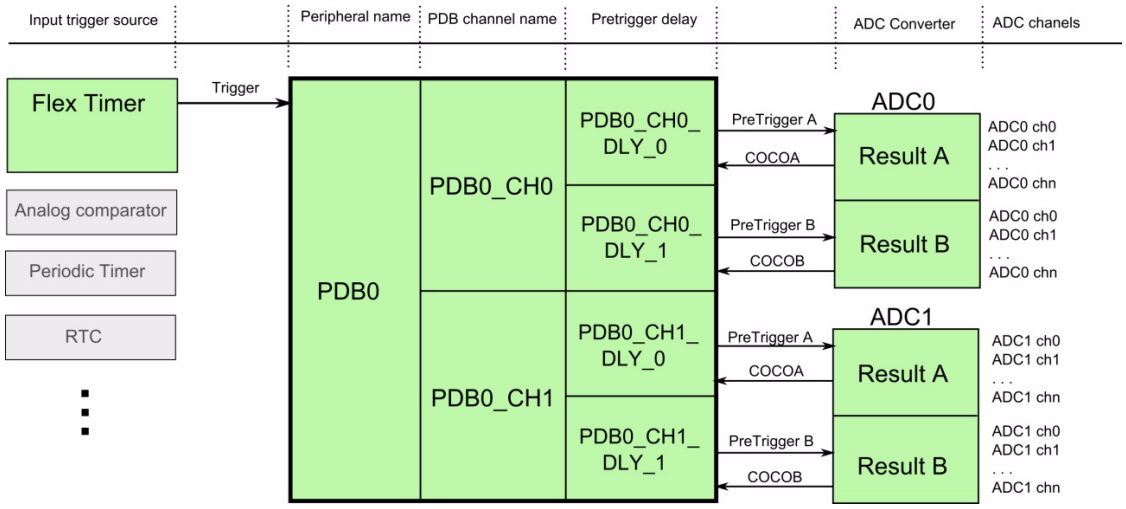


Figure 1. Internal arrangement of PDB

Input trigger source—One possible input source which starts the PDB counter for all PDB channels. All possible input sources can be found in the chip reference manual, in the section "PDB Input Trigger Connections." The input trigger source number must be entered into register PDBx_SC, field TRGSEL.

PDB channel—Each PDB channel is associated with one ADC. This means that ADC0 is associated with PDB channel 0 and ADC1 is associated with PDB channel 1.

Pre-trigger delay—Each pre-trigger is generated after a predefined delay, which is measured from the input trigger event. After this delay, an ADC convertor begins to measure and the result of conversion is stored in the result register corresponding to the respective pre-trigger delay. Relationships between pre-triggers and result registers are described in the Table 1.

Table 1. Relation between pre-triggers and result registers

Respective Pre-Trigger Delay	Register with Conversion Result
PDB0_CH0_DLY0	ADC0_RA
PDB0_CH0_DLY1	ADC0_RB
PDB0_CH1_DLY0	ADC1_RA
PDB0_CH1_DLY1	ADC1_RB

3 PDB Set-up

To accurately synchronize between the FlexTimer and ADC, three peripherals must be configured. Only the settings necessary for synchronization will be described. The settings for PWM generation and ADC module are application specific and therefore cannot be described in detail here.

Example 1. PDB settings

```
PDBinit(void)
{
SIM_SCGC6 |= SIM_SCGC6_PDB_MASK;    // enable clock for PDB module
PDB0_MOD = MODULO;                  // modulo for PDB counter should be same as FTM modulo
                           //enable required pre-triggers, enable back-to-back operation
PDB0_CH1C1 = PDB_C1_EN (0x03) | PDB_C1_TOS(0x03) | PDB_C1_BB (0x02);
PDB0_CH0C1 = PDB_C1_EN (0x03) | PDB_C1_TOS(0x03) | PDB_C1_BB (0x02);
PDB0_CH0DLY0 = 150;                 // delay for first pre-trigger of PDB channel 0
PDB0_CH1DLY0 = 200;                 // delay for first pre-trigger of PDB channel 1
PDB0_SC = PDB_SC_PDBEN_MASK
                  | PDB_SC_PRESCALER(0x0)
                  | PDB_SC_TRGSEL(0x8)  // 0x8=FTM0
                  | PDB_SC_MULT(0)
                  | PDB_SC_LDOK_MASK
                  | PDB_SC_PDBEIE_MASK;
}
```

FlexTimer *init trigger* signal needs to be routed out of the FlexTimer by the FlexTimer settings command.

Example 2. FlexTimer settings

```
FTM0_EXTTRIG |= FTM_EXTTRIG_INITTRIGEN_MASK;
```

Example 3. ADC settings

```
ADC1_SC2 |= ADC_SC2_ADTRG_MASK;     // Enable H/W trigger for ADC1
ADC0_SC2 |= ADC_SC2_ADTRG_MASK;     // Enable H/W trigger for ADC0
SIM_SOPT7 = 0;                      // Select H/W trigger source for the ADCs
ADC1_SC1A = ADC_SC1_ADCH(11);       // BEMF voltage measurement (ADC1_RA)
ADC1_SC1B = ADC_SC1_AIEN_MASK | ADC_SC1_ADCH(12);    // DC-Bus current (ADC1_RB)
ADC0_SC1A = ADC_SC1_ADCH(19);                        // DC voltage (ADC0_RA)
ADC0_SC1B = ADC_SC1_AIEN_MASK | ADC_SC1_ADCH(14);    // Pressure sensor (ADC0_RB)
```

4 Routing the PDB pre-trigger onto a GPIO pin

To check whether the synchronization is working correctly, it is good to route the pre-trigger signal to a GPIO pin. For this purpose, you can use an unused channel of the FlexTimer.

The following settings are necessary to prepare the trigger diagnostic pin. This example selects the FlexTimer channel 7, however it is possible to select any free FlexTimer channel.

```
FTM0_C7SC |= FTM_CnSC_MSA_MASK;   // Set the output compare mode to FTM channel 7

FTM0_C7SC |= FTM_CnSC_ELSA_MASK; // Enable the output toggle on match

PORTD_PCR7 |= PORT_PCR_MUX(4);    // Set the pin alternative for PORTD 7 - Route the FTM signal
to a pin
```

Next, fill the FlexTimer value register with the value of the required delay. If the PDB delay is changing periodically, the FlexTimer value register requires periodic updates.

```
FTM0_C7V = PDB0_CH1DLY0;   // Enter the value of the delay from the pre-trigger delay register
```

5 PDB sequence error handling

The ADC*n* block can be triggered for a conversion by one pre-trigger from PDB channel *n*. When one conversion is in progress that is triggered by one of the pre-triggers from PDB channel *n*, then a new trigger from the PDB channel's corresponding pre-trigger *m* cannot be accepted by ADC*n*, and ERR[m] is set.

For example, in the Sensorless BLDC six-step control, ADC channels are periodically switched according to rotor position. This principle may cause a PDB sequence error. To avoid this type of error, it is necessary to switch to the channel only when no conversion is active. It is recommended to only allow switching after the ADC conversion complete interrupt is received.

Changing the ADC channel can be also tricky, because writing SC1A while SC1A is actively controlling a conversion, aborts the current conversion therefore; the conversion complete interrupt is not called, the result register is not read, the PDB flag is not cleared, and a PDB sequence error occurs. In Software Trigger mode, when SC2[ADTRG]=0, writes to SC1A it subsequently initiates a new conversion if SC1[ADCH] contains a value other than all 1s.

When the PDB is not managed correctly, PDB sequence errors can occur. These situations occur given several scenarios. For example:

- When delay0 and delay1 of one PDB channel are too close and the first conversion is not complete before the start of a new conversion.
- When the asynchronous event that causes the conversion complete flag is not cleared on time and a new conversion is requested, such as:
  - an interrupt with a high priority causes the COCO flag not to be cleared on time.
  - using break points during application debugging.
- When the ADC channel is changed during an active ADC conversion, such as:
  - a BLDC motor with Hall sensors controlled by interrupts.
  - a slow control loop is asynchronous to a fast control loop.

After a PDB sequence error occurs the PDB remains inactive until the errors are cleared. Before a PDB error is cleared, the PDB module must be disabled and re-enabled after errors are cleared.

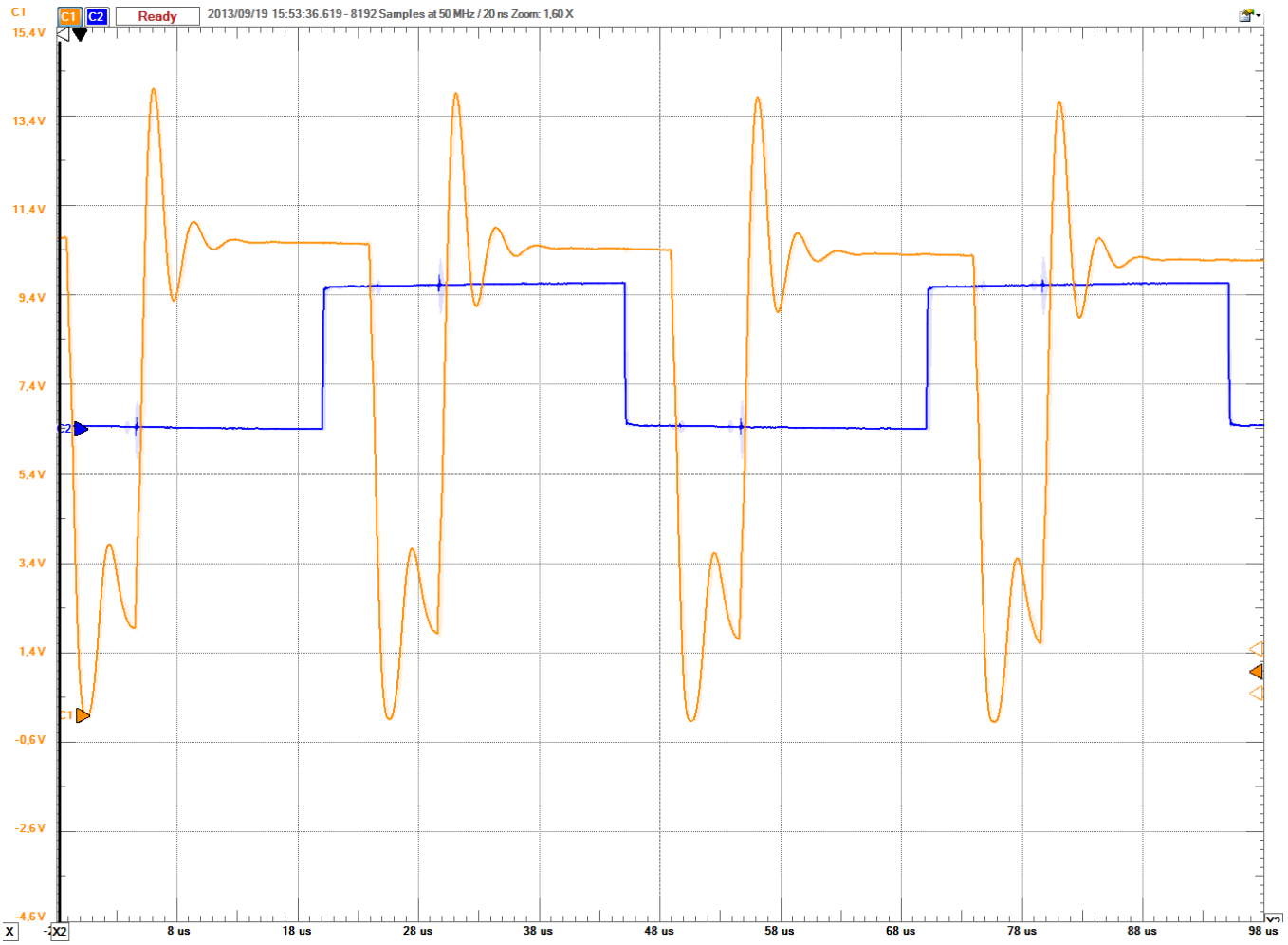
Example 4. An example of a PDB sequence error interrupt routine

```
void PDB_error_isr(void)
{
  PDB0_SC &= ~PDB_SC_PDBEN_MASK;  // disable PDB
  PDB0_CH0S &= ~PDB_S_ERR_MASK;   // reset error CH0
  PDB0_CH1S &= ~PDB_S_ERR_MASK;   // reset error CH1
  PDB0_SC |= PDB_SC_PDBEN_MASK;    // Enable PDB
}
```

This function is called when a PDB error occurs. Bit PDBx_SC_PDBEIE must be set to enable the sequence error interrupt. It is not required to set bit PDBx_SC_PDBIE to enable the error interrupt.

6 Application example

Figure 2 illustrates a practical usage of PDB synchronization when a noisy signal is measured. As shown, the measured signal (orange signal) is effectively filtered by proper sampling. Samples are taken after a transient event finishes, so only clean signal is measured. Points where the samples are taken are indicated by the blue diagnostic signal.



Legend:
Orange waveform: Noisy signal to be measured.
Blue waveform: Diagnostic signal (toggled on the ADC trigger).

Figure 2. Example of proper setting of PDB trigger, for noisy signal filtering

7 Reference documentation

K60 Sub-Family Reference Manual for 100 MHz devices in 100 pin packages (document K60P100M100SF2V2RM)

K60 Sub-Family Reference Manual for 100 MHz devices in 121 pin packages (document K60P121M100SF2V2RM)

K60 Sub-Family Reference Manual for 100 MHz devices in 144 pin packages (document K60P144M100SF2V2RM)

K60 Reference Manual (document number K60P120M100SF2RM)

K60 Sub-Family Reference Manual for 100 MHz devices in 100 pin packages (document K60P100M100SF2RM)

K60 Sub-Family Reference Manual for 100 MHz devices in 121 pin packages (document K60P121M100SF2RM)

K60 Sub-Family Reference Manual for 100 MHz devices in 144 pin packages (document K60P144M100SF2RM)

8 Revision history

Revision 0 is the initial release of the document.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AN4822
Rev. 0
12/2013