





## Agenda

1. Module Overview
2. On-chip Interconnections and Inter-module Dependencies
3. Software Configuration
4. Example Use Case
5. PIT Frequently Asked Questions (FAQ)

 External Use | 1 

In this presentation we'll cover:

- An overview of the PIT module itself
- The on-chip interconnections and inter-module dependencies
- Software configurations
- An example use case
- And some frequently asked questions



First, let's start with an overview of the module.



## PIT Features and Application Benefits

### Features

- Ability to generate DMA trigger pulses
- Ability to generate interrupts
- Maskable interrupts
- Independent timeout periods for each timer

### Application Benefits

- DMA and interrupt triggering capability

 External Use | 3 

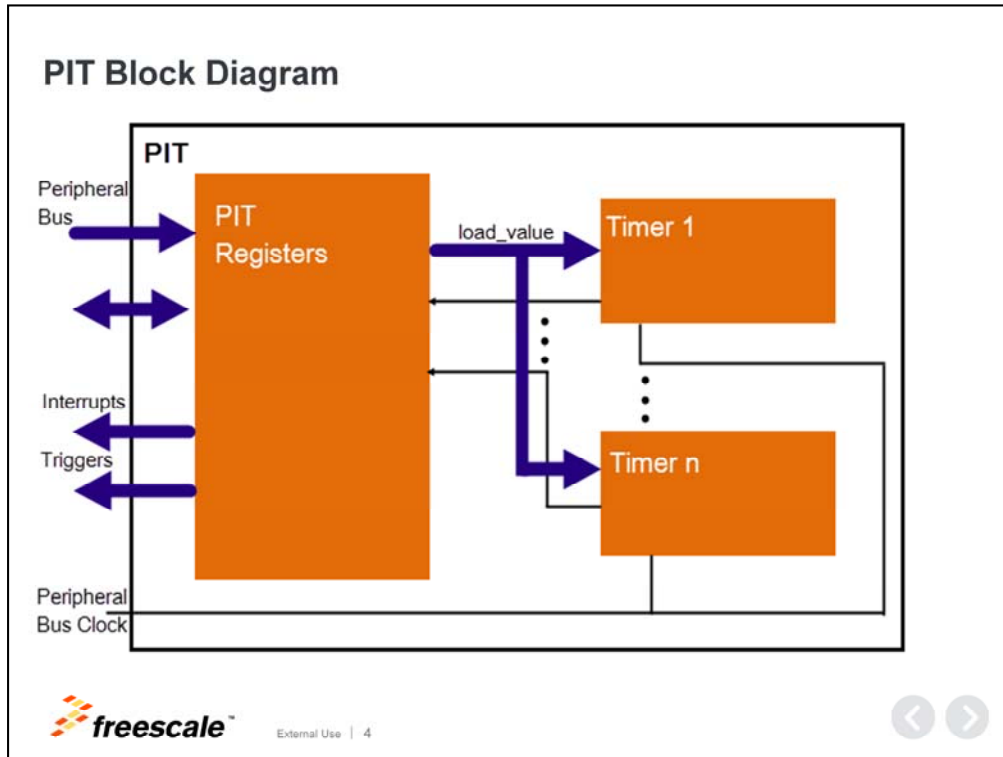
## PIT Features and Application Benefits

The PIT module **features** include:

- The ability to generate DMA trigger pulses
- The ability to generate interrupts
- Maskable interrupts and,
- Independent timeout periods for each timer

Application **benefits** include:

- DMA and interrupt triggering capability for reduced CPU intervention

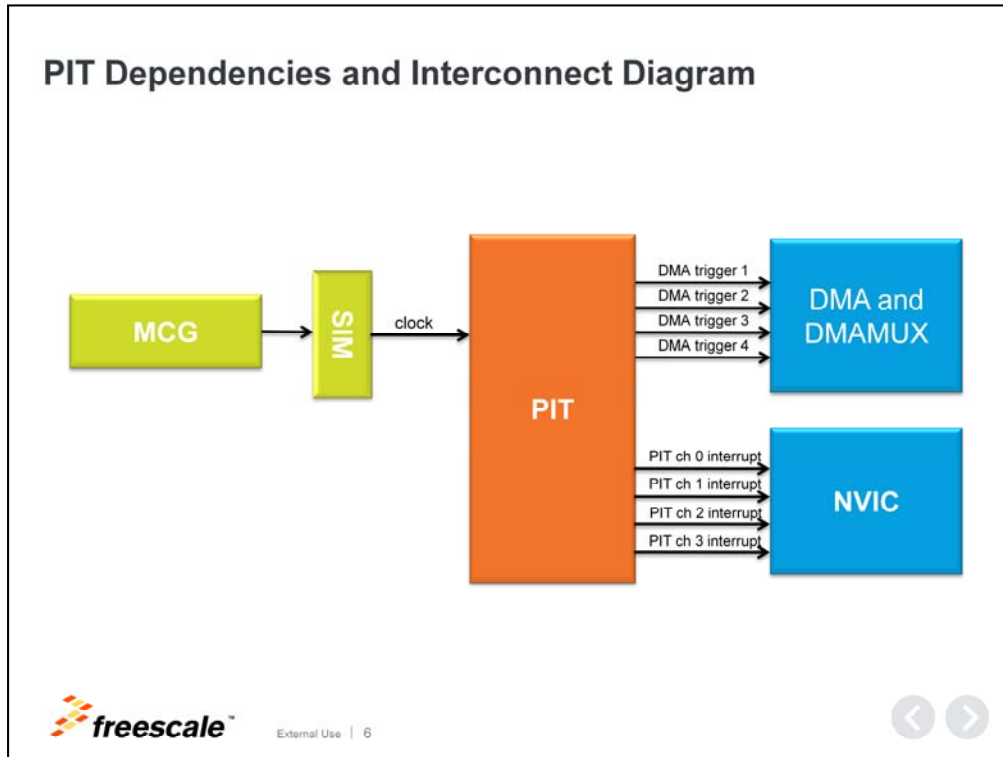


### PIT Block Diagram

The PIT module generates precise interrupts at regular intervals with minimal processor intervention. It can be used to perform many routines, from periodic signals to scheduling functions. This module is internal and has no off-chip signals.

## 2. On-chip Interconnections and Inter-module Dependencies

Now, let's talk about on-chip interconnection and inter-module dependencies.



## PIT Dependencies and Interconnect Diagram

The clock gating for PIT is controlled by the System Integration Module, or SIM.

The SIM should be initialized to enable the clock for the PIT. The PIT module registers are not accessible until the clock is enabled.

The PIT is inclusive of an array of timers that can be used to raise interrupts and trigger DMA channels.

The PIT module can trigger a DMA transfer on the first four DMA channels.

The PIT generates periodic trigger events to the DMA Mux as DMA channels 0-3 for PIT channels 0-3, respectively.

Each timer can be used to generate trigger pulses and interrupts. Each interrupt is available on a separate interrupt line.

Also, the PIT module can be used as an input trigger for the PDB module.



In this next section we'll discuss software configuration.



### Kinetis SDK PIT Driver

```

#define BOARD_PIT_INSTANCE 0
volatile bool pitIsrFlag[2] = {false};
int main(void)
{
    pit_user_config_t chn0Conf = {
        .isInterruptEnabled = true,
        .periodUs = 1000000u
    };

    pit_user_config_t chn1Conf = {
        .isInterruptEnabled = true,
        .periodUs = 2000000u
    };
    hardware_init();
    LED1_EN;
    LED2_EN;
    PIT_DRV_Init(BOARD_PIT_INSTANCE, false);


    PIT_DRV_InitChannel(BOARD_PIT_INSTANCE, 0, &chn0Conf);
    PIT_DRV_InitChannel(BOARD_PIT_INSTANCE, 1, &chn1Conf);

    PRINTF("\n\nStarting channel No.0 ...");
    PIT_DRV_StartTimer(BOARD_PIT_INSTANCE, 0);

    PRINTF("\n\nStarting channel No.1 ...");
    PIT_DRV_StartTimer(BOARD_PIT_INSTANCE, 1);

    while (true)
    {
        if (true == pitIsrFlag[0])
        {
            PRINTF("\n\n Channel No.0 interrupt is
occured !");
            LED1_TOGGLE;
            pitIsrFlag[0] = false;
        }
        if (true == pitIsrFlag[1])
        {
            PRINTF("\n\n Channel No.1 interrupt is
occured !");
            LED2_TOGGLE;
            pitIsrFlag[1] = false;
        }
    }
}

```


External Use | 8
⏪ ⏩

## Kinetis SDK PIT Driver

This example shows how to use the Kinetis SDK PIT peripheral driver to initialize and configure the PIT timer.

To initialize the PIT module, call the PIT\_DRV\_Init function. This function enables the PIT module and clock automatically. One of the parameters passed in the PIT\_DRV\_Init function configures the timers to run or stop in debug mode, the other parameter is the instance number of the PIT.


To initialize a timer channel, call the PIT\_DRV\_InitChannel function. The user must define the structure PitConfig which is used as a parameter to the initialization function. This structure allows the user to enable interrupts and set the timer period in microseconds.


Timers do not start counting by default, the function PIT\_DRV\_StartTimer must be called to start counting. After calling this function, the timer loads the period value, counts down to 0, and then loads the respective start value again. Each time a timer reaches 0, it can generate a trigger pulse and sets the timeout interrupt flag.

### Kinetis SDK PIT Driver

```
void PIT_DRV_StopTimer ( uint32_t instance,  
                        uint32_t channel )
```

```
pit_status_t PIT_DRV_Deinit ( uint32_t instance )
```

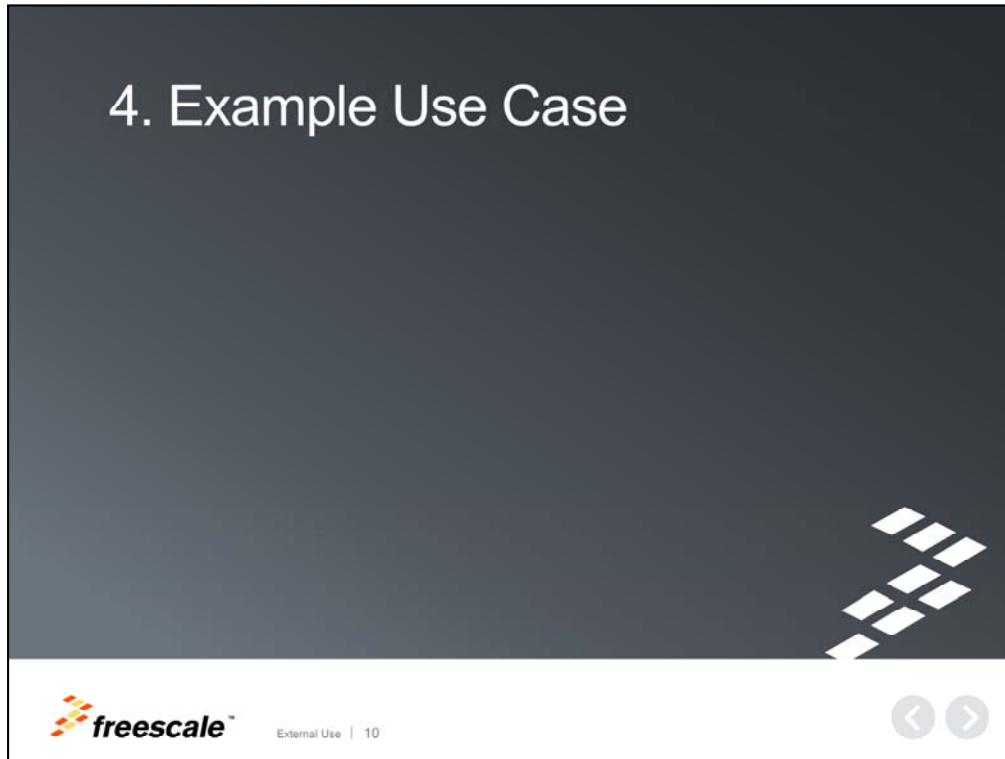


 External Use | 9

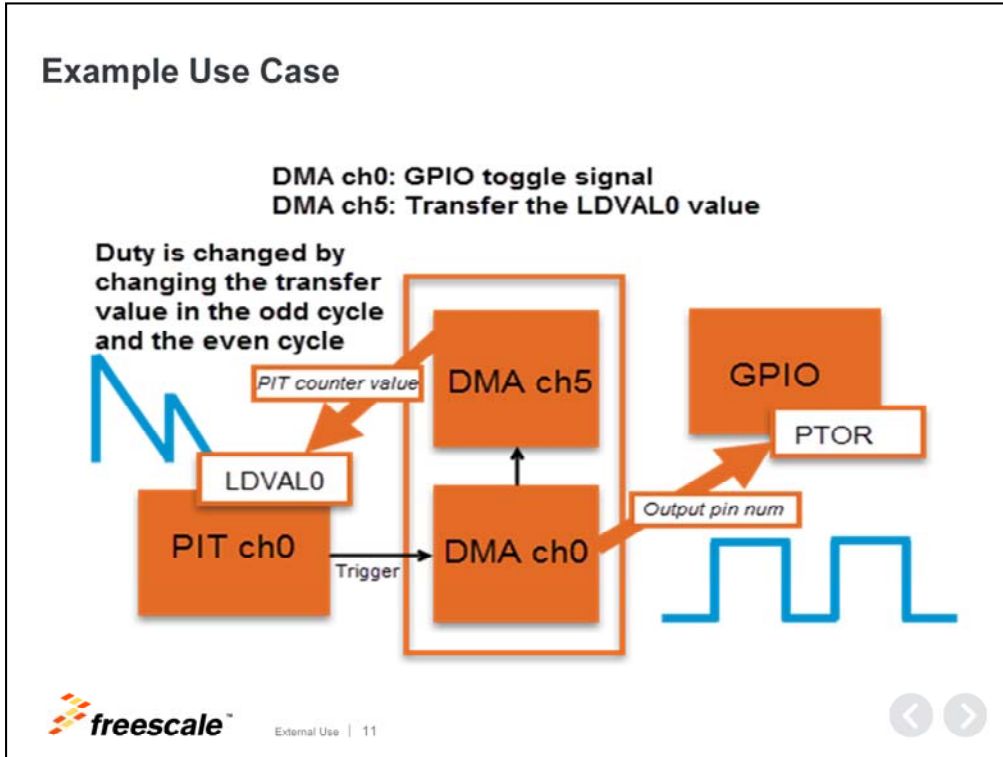
### Kinetis SDK PIT Driver

Call the PIT\_DRV\_StopTimer function to stop the timer at any time. This function stops the count of every timer. Timers reload their periods when the PIT\_DRV\_StartTimer function is called again.

To close the PIT module entirely, call the PIT\_DRV\_Deinit function. This function disables all PIT interrupts and the PIT clock. It then gates the PIT clock control.



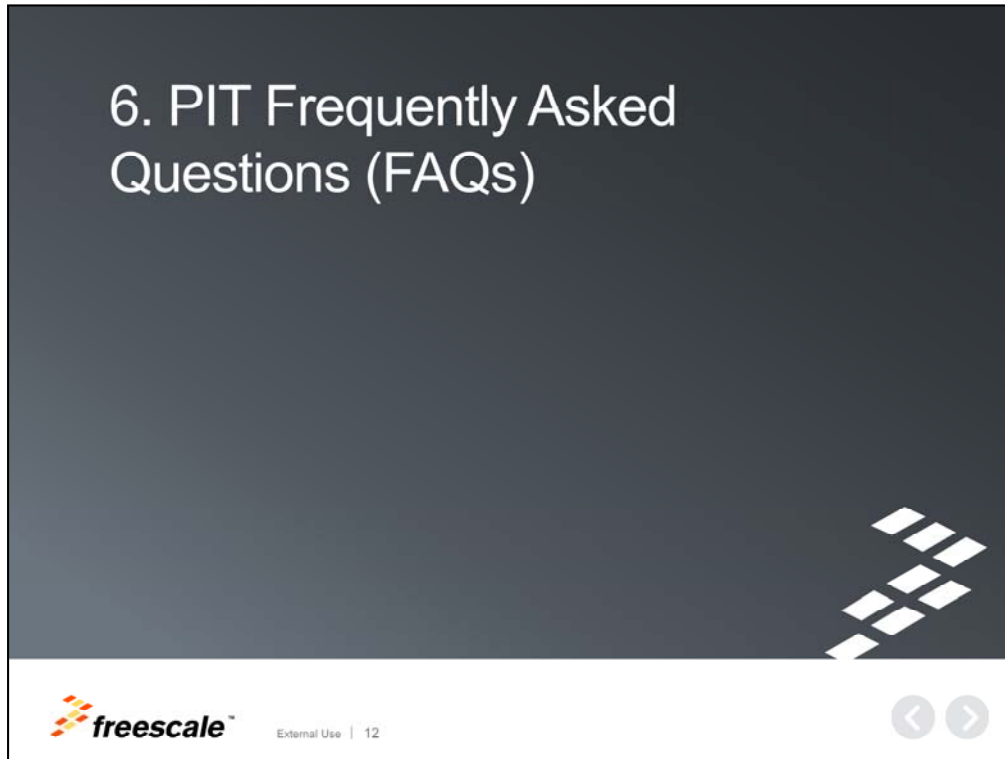
Now, let's review an example use case.



### Example Use Case

This figure demonstrates the PIT timer being used to generate a waveform of varying duty cycle with a GPIO.

The PIT load value register (LDVAL) is loaded with an initial value and counts down to zero. When it reaches zero, it generates a DMA request to DMA channel 0, which then drives the GPIO and generates a DMA request to DMA channel 5 to reload the load value registers (LDVAL) of the PIT. The duty cycle of the GPIO waveform can be modified by changing the timeout value of the PIT.



Finally, let's discuss some frequently asked questions.

## PIT FAQs

### Q: Does the timer count down while in debug mode?

A: In debug mode, the timers will be frozen based on the MCR[FRZ] bit. This is intended to aid software development, allowing the developer to halt the processor, investigate the current state of the system such as the timer values, and then continue the operation.

### Q: How is the value calculated for the LDVAL register trigger?

A: Use the following formula

$$\text{LDVAL trigger} = (\text{period} / \text{clock period}) - 1$$

The clock associated to the PIT module is the bus clock.



## PIT FAQs

### Question – Does the timer count down while in debug mode?

Answer – In Debug mode, the timers will be frozen based on the MCR[FRZ] bit. This is intended to aid software development, allowing the developer to halt the processor, investigate the current state of the system such as the timer values, and then continue the operation.

### Question – How is the value calculated for the LDVAL register trigger?

Answer – Please use the following formula:

$$\text{LDVAL trigger} = (\text{period} / \text{clock period}) - 1$$

The clock associated to the PIT module is the bus clock.

## References

- **Application Notes:**
  - [AN4419](#): Using DMA and GPIO to emulate timer functionality on Kinetis Family devices
- **Website:** [Freescale.com/Kinetis](http://Freescale.com/Kinetis)
- **Community:** [community.freescale.com/community/Kinetis](http://community.freescale.com/community/Kinetis)



External Use | 14



## References

This concludes our presentation on the PIT module for Kinetis K Series MCUs.

For more information on the PIT module, please visit the app note listed here.

We also invite you to visit us on the web [Freescale.com/Kinetis](http://Freescale.com/Kinetis) and check out our Kinetis community page.



[www.Freescale.com](http://www.Freescale.com)

© 2015 Freescale Semiconductor, Inc. | *External Use*