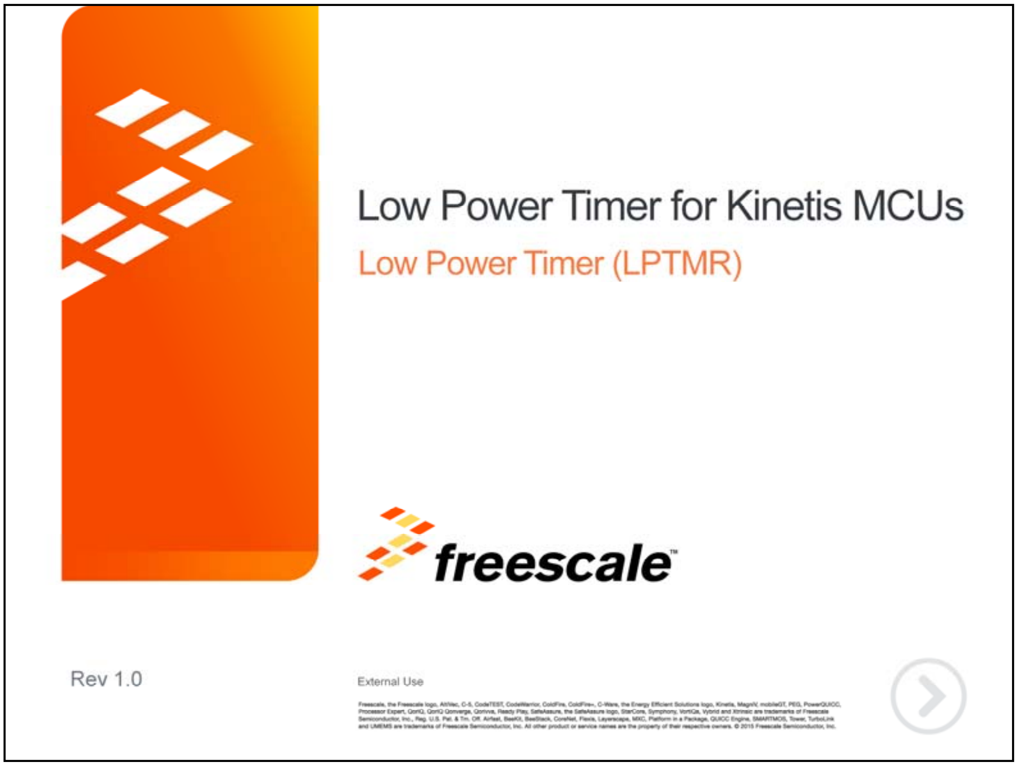


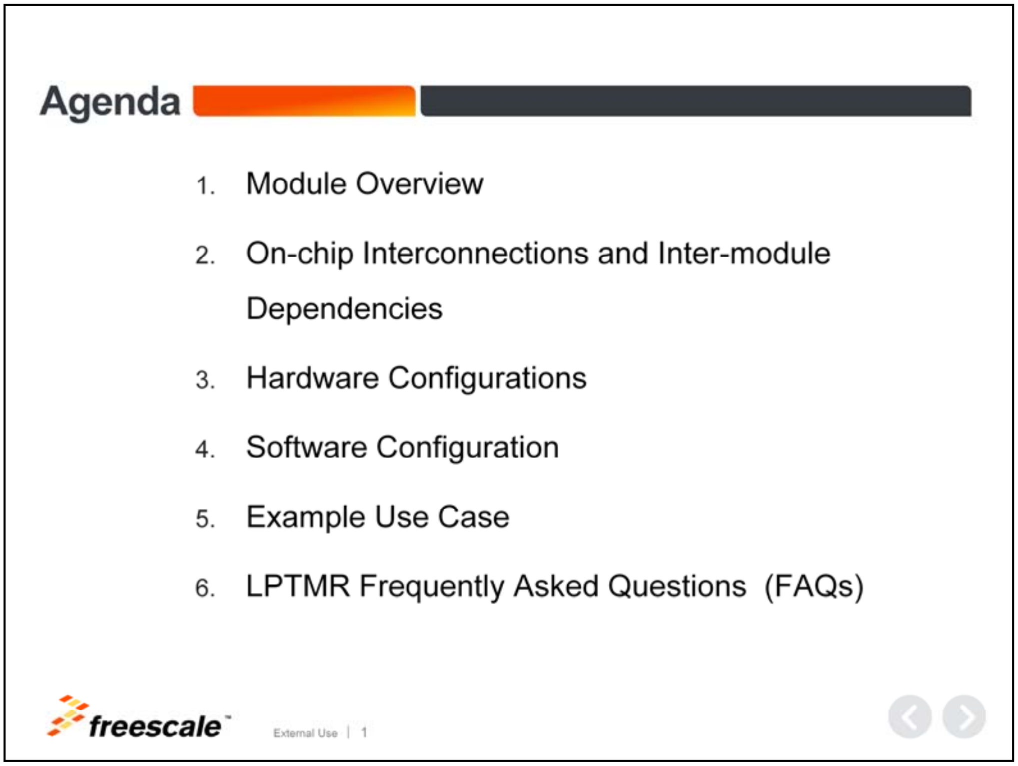
Low Power Timer for Kinetis MCUs

Low Power Timer (LPTMR)

Rev 1.0

External Use

Hello, welcome to this presentation of the low power timer, or LPTMR, module for Kinetis MCUs. In this session you’ll learn about the LPTMR, it’s main features and the application benefits of leveraging this function.



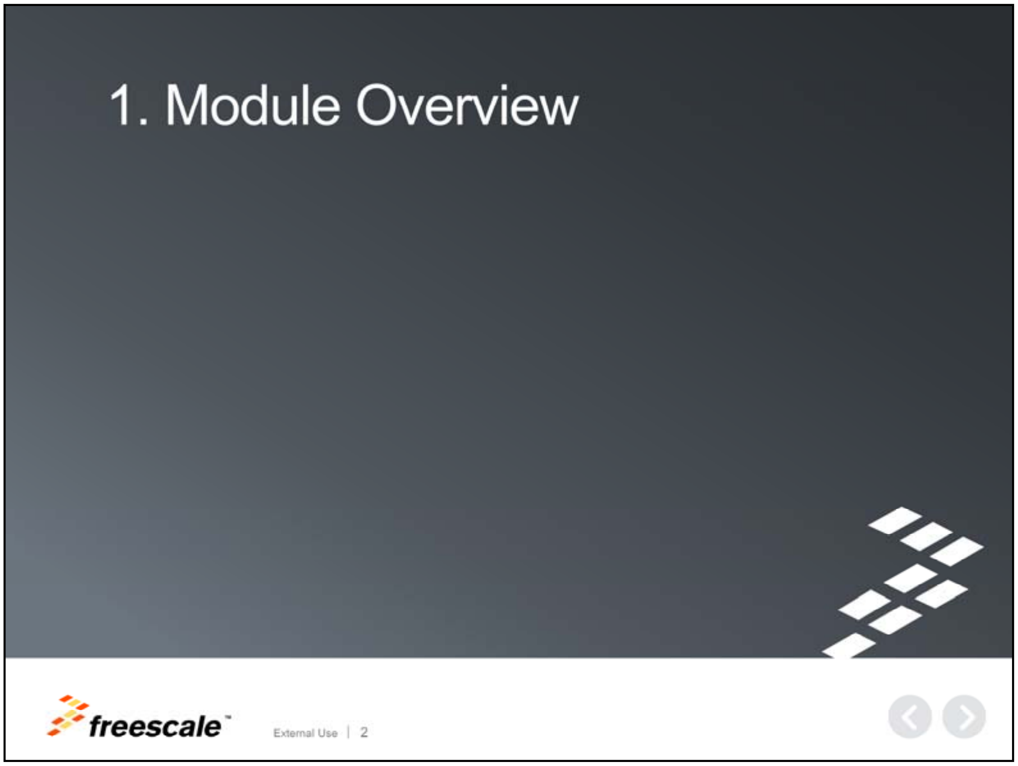
Agenda

1. Module Overview
2. On-chip Interconnections and Inter-module Dependencies
3. Hardware Configurations
4. Software Configuration
5. Example Use Case
6. LPTMR Frequently Asked Questions (FAQs)

freescale External Use | 1

In this presentation, we’ll cover:

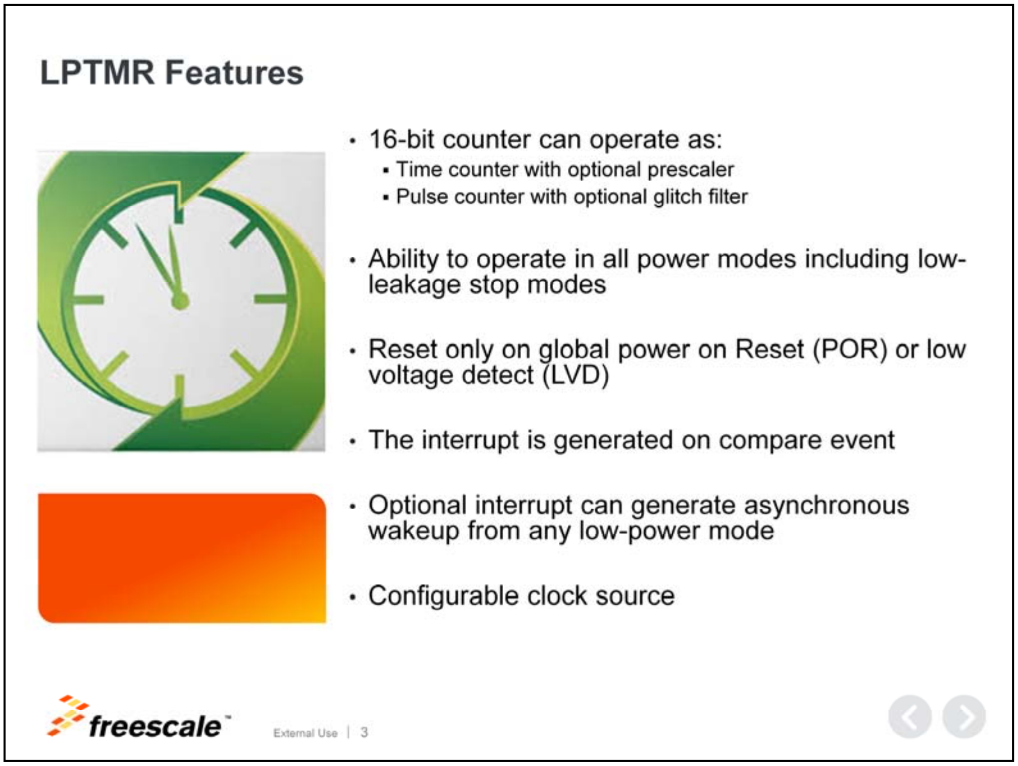
- An overview of the module
- On-chip interconnections and inter-module dependencies
- Hardware and software configuration
- An example use case
- And a few frequently asked questions



1. Module Overview

freescale External Use | 2

Let’s first begin with an overview of the module.



LPTMR Features

- 16-bit counter can operate as:
  - Time counter with optional prescaler
  - Pulse counter with optional glitch filter
- Ability to operate in all power modes including low-leakage stop modes
- Reset only on global power on Reset (POR) or low voltage detect (LVD)
- The interrupt is generated on compare event
- Optional interrupt can generate asynchronous wakeup from any low-power mode
- Configurable clock source

External Use | 3

LPTMR Features

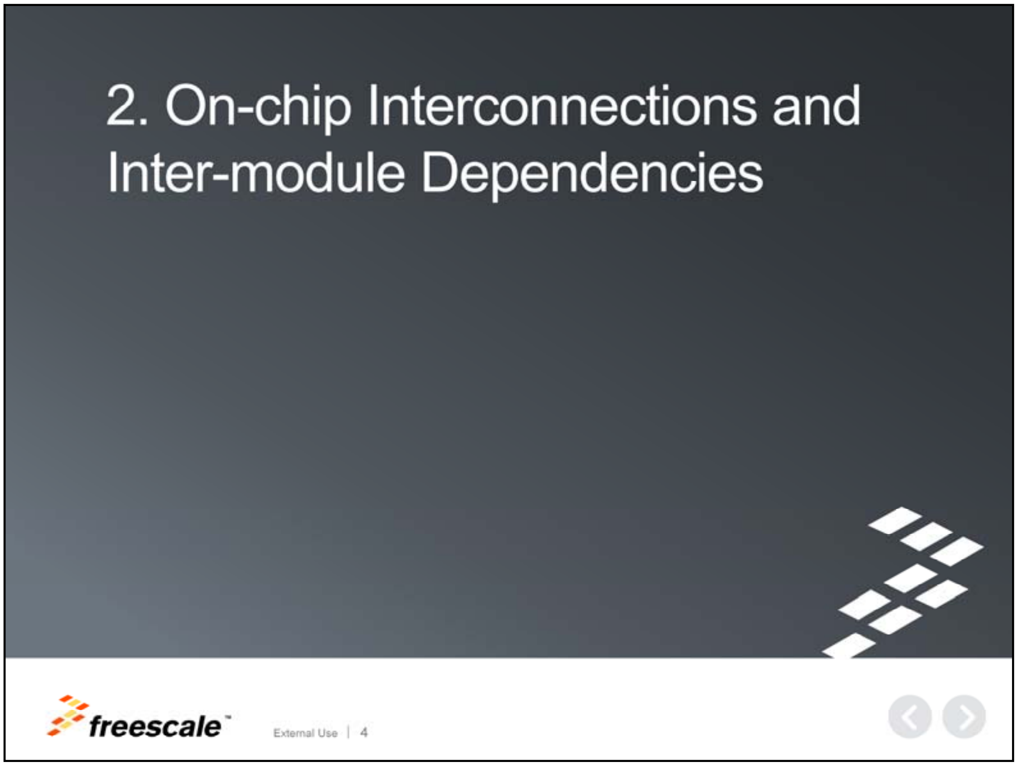
Features of the low power timer, include:

•A 16-bit counter: This block can be configured to operate as a time counter with optional prescaler, or as a pulse counter with optional glitch filter across all power modes, including the low-leakage modes.

•The LPTMR is reset only on global power on reset (POR) or low voltage detect (LVD), allowing it to be used as a time of day counter.

•The interrupt is generated asynchronously to the system clock and can be used to generate a wakeup from any low-power mode, including the low-leakage modes.

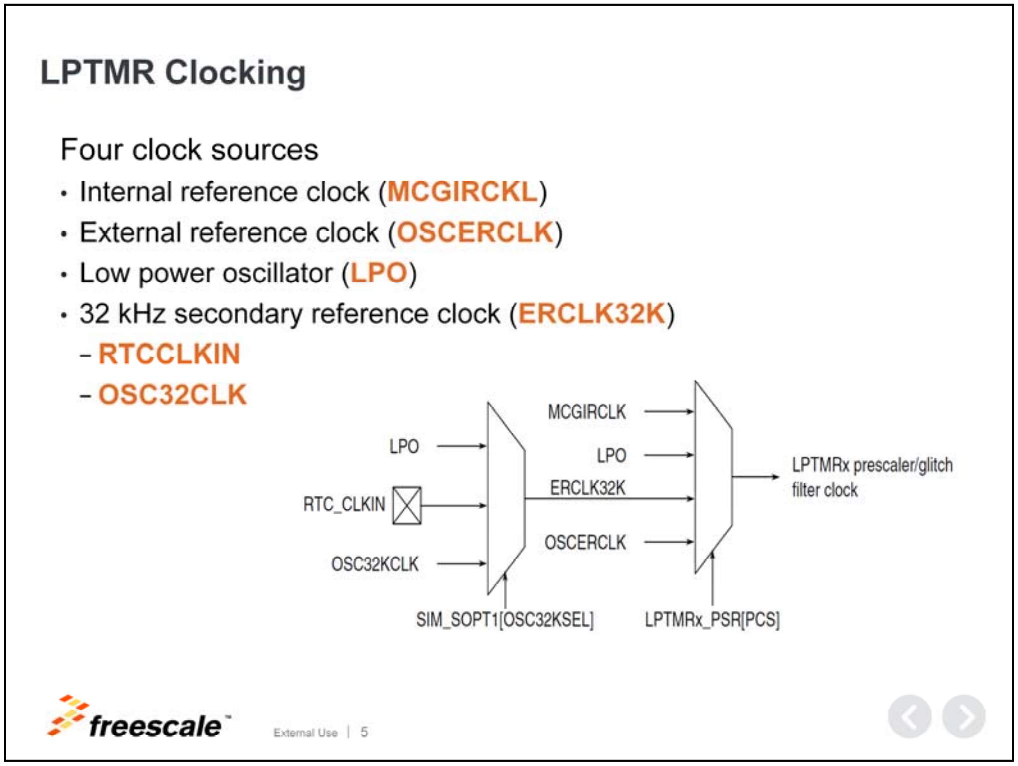
•The LPTMR has a configurable clock source.



2. On-chip Interconnections and Inter-module Dependencies

External Use | 4

Now, let’s discuss on-chip interconnections and inter-module dependencies.



LPTMR Clocking

Four clock sources

- Internal reference clock (**MCGIRCKL**)
- External reference clock (**OSCERCLK**)
- Low power oscillator (**LPO**)
- 32 kHz secondary reference clock (**ERCLK32K**)
  - **RTCCLKIN**
  - **OSC32CLK**

freescale External Use | 5

LPTMR Clocking

There are four clock sources by which the LPTMR is clocked:

1. The internal reference clock. This reference is not available in VLPS, LLS or VLLS modes.

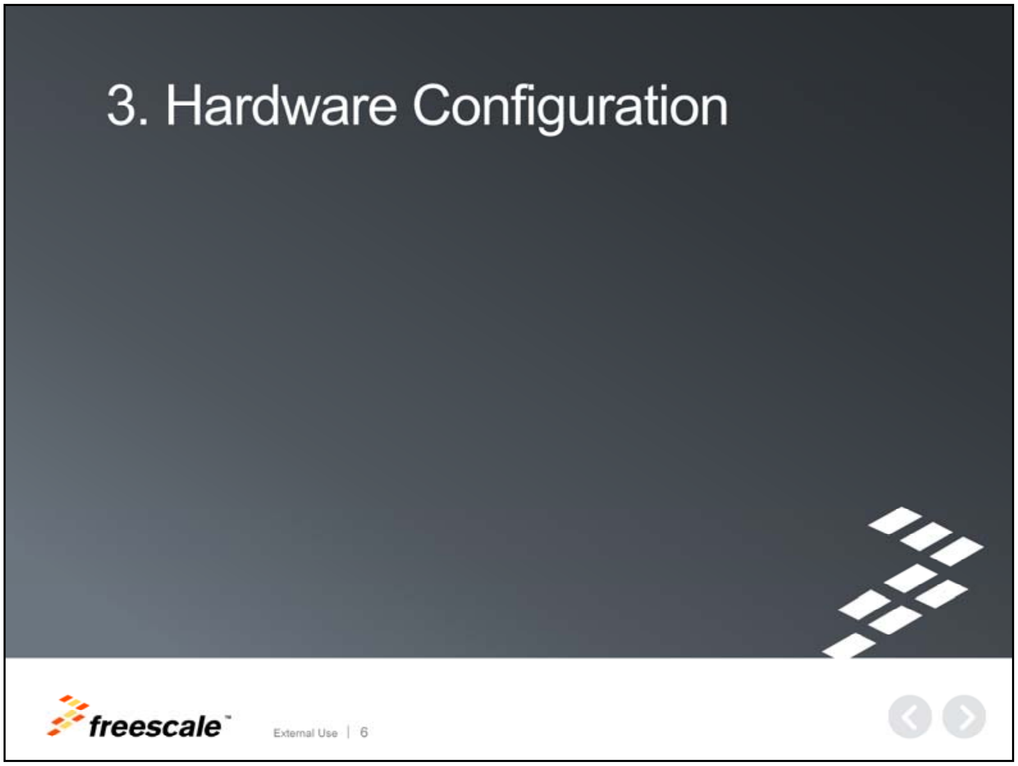
2. The external reference clock, which is not available in VLLS0 mode.

3. The low power oscillator. This is a 1kHz clock and is also not available in VLLS0 mode.

4. The 32 kHz secondary external reference clock, which is available in all low power modes.

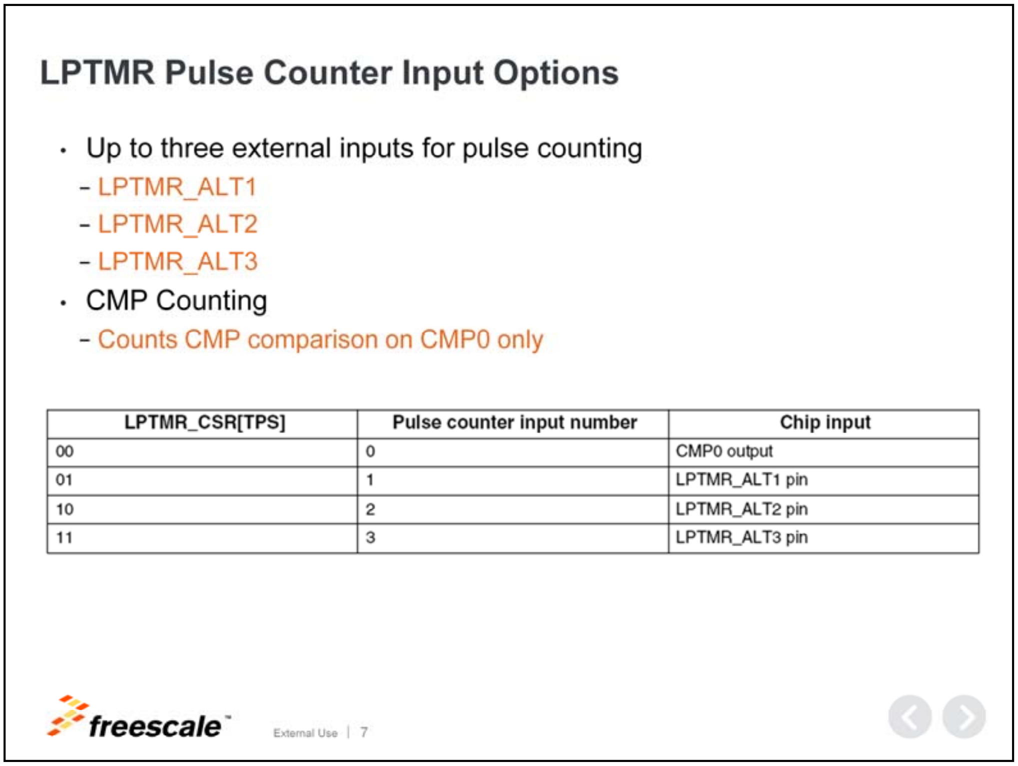
The LPTMR operates in all power modes as long as a clock is available.

The LPTMR prescale register configures the input source to be used in pulse counter mode with the prescaler clock select bits. These bits should only be altered when the LPTMR is disabled.



3. Hardware Configuration

Now, let’s discuss hardware configuration.



LPTMR Pulse Counter Input Options

- Up to three external inputs for pulse counting
  - LPTMR_ALT1
  - LPTMR_ALT2
  - LPTMR_ALT3
- CMP Counting
  - Counts CMP comparison on CMP0 only

LPTMR_CSR[TPS]	Pulse counter input number	Chip input
00	0	CMP0 output
01	1	LPTMR_ALT1 pin
10	2	LPTMR_ALT2 pin
11	3	LPTMR_ALT3 pin

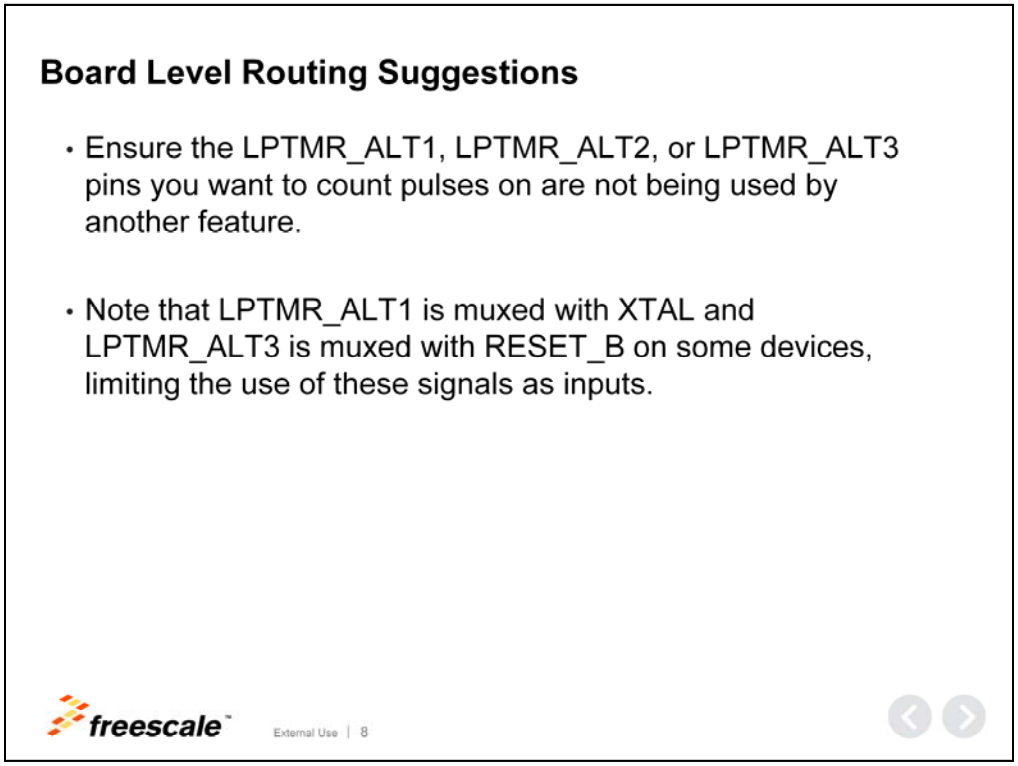
External Use | 7

LPTMR Pulse Counter Input Options

In pulse counter mode, the selected input source directly clocks the counter and no other clock source is required.

The external inputs for pulse counting are LPTMR_ALT1, LPTMR_ALT2, and when available LPTMR_ALT3.

The LPTMR can also count CMP comparisons on CMP0.



Board Level Routing Suggestions

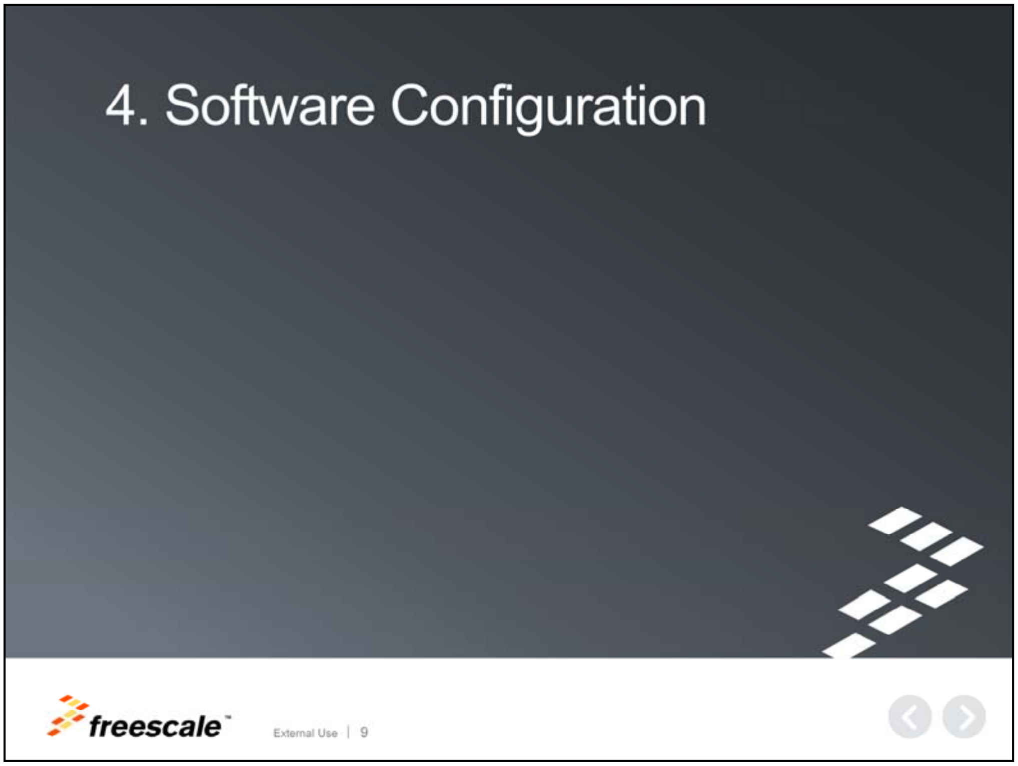
- Ensure the LPTMR_ALT1, LPTMR_ALT2, or LPTMR_ALT3 pins you want to count pulses on are not being used by another feature.
- Note that LPTMR_ALT1 is muxed with XTAL and LPTMR_ALT3 is muxed with RESET_B on some devices, limiting the use of these signals as inputs.

External Use | 8

Board Level Routing Suggestions

Ensure the LPTMR_ALT1, LPTMR_ALT2, or LPTMR_ALT3 pins you want to count pulses on are not being used by another feature.

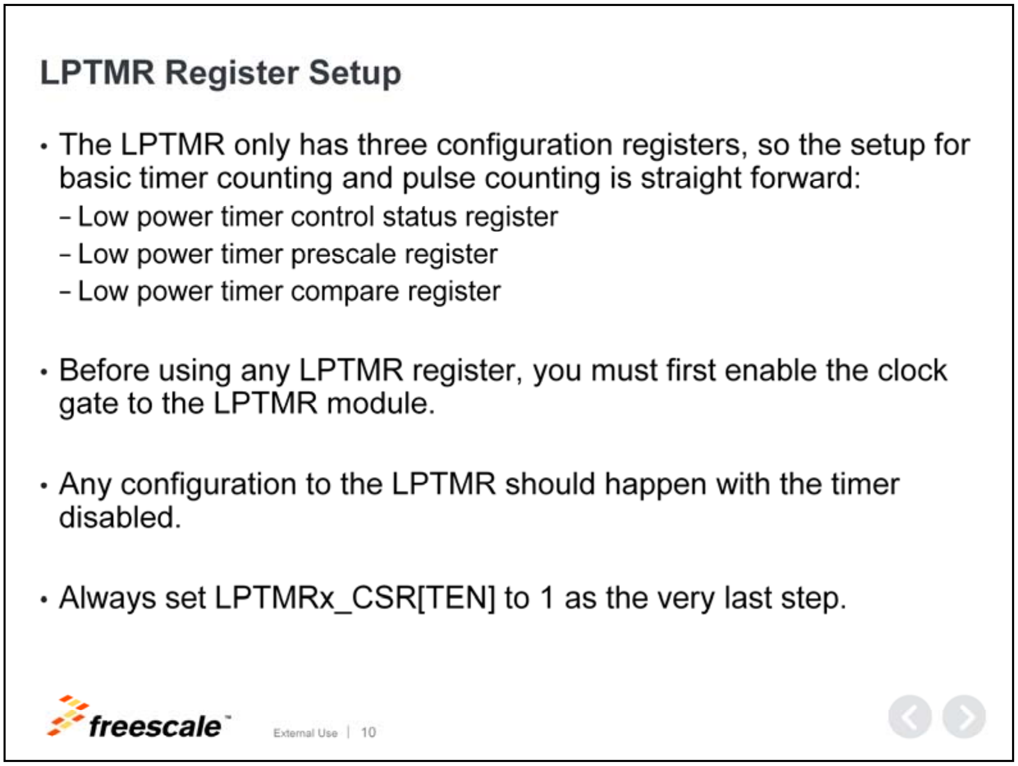
Note that LPTMR_ALT1 is muxed with XTAL, and LPTMR_ALT3 is muxed with RESET_B on some devices, limiting the use of these signals as inputs.



4. Software Configuration

External Use | 9

Next, let’s move on and discuss software configuration.



LPTMR Register Setup

- The LPTMR only has three configuration registers, so the setup for basic timer counting and pulse counting is straight forward:
  - Low power timer control status register
  - Low power timer prescale register
  - Low power timer compare register
- Before using any LPTMR register, you must first enable the clock gate to the LPTMR module.
- Any configuration to the LPTMR should happen with the timer disabled.
- Always set LPTMRx_CSR[TEN] to 1 as the very last step.

External Use | 10

LPTMR Register Setup

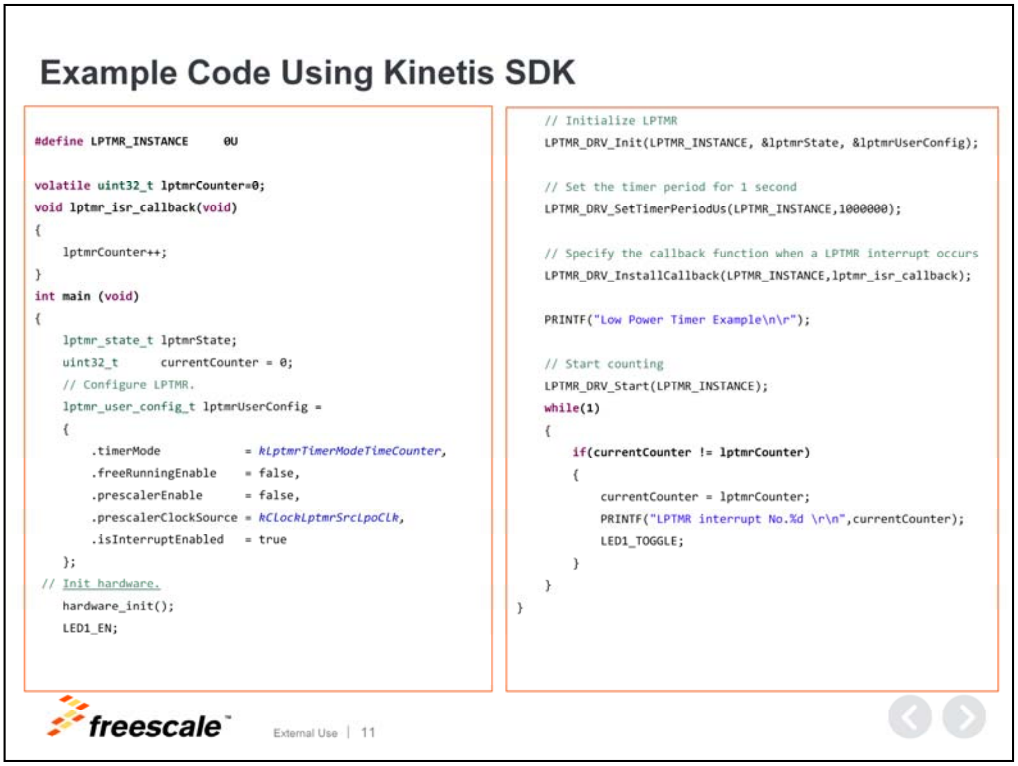
The LPTMR only has three configuration registers, so the setup for basic timer counting and pulse counting is straight forward.

The three configuration registers are:

1. The low power timer control status register
2. The low power timer prescale register
3. The low power timer compare register

Before using any of these registers, it is important to enable the clock gate to the LPTMR. In addition, any configuration to the LPTMR should happen with the timer disabled, with LPTMRx_CSR[TEN]=0 and

always set LPTMRx_CSR[TEN] to 1 as the very last step



Example Code Using Kinetis SDK

```
#define LPTMR_INSTANCE     0U

volatile uint32_t lptmrCounter=0;
void lptmr_isr_callback(void)
{
    lptmrCounter++;
}
int main (void)
{
    lptmr_state_t lptmrState;
    uint32_t       currentCounter = 0;
    // Configure LPTMR.
    lptmr_user_config_t lptmrUserConfig =
    {
        .timerMode             = kLptmrTimerModeTimeCounter,
        .freeRunningEnable     = false,
        .prescalerEnable       = false,
        .prescalerClockSource  = kClockLptmrSrcLpoClk,
        .isInterruptEnabled    = true
    };
 // Init hardware.
    hardware_init();
    LED1_EN;
    // Initialize LPTMR
    LPTMR_DRV_Init(LPTMR_INSTANCE, &lptmrState, &lptmrUserConfig);

    // Set the timer period for 1 second
    LPTMR_DRV_SetTimerPeriodUs(LPTMR_INSTANCE,1000000);

    // Specify the callback function when a LPTMR interrupt occurs
    LPTMR_DRV_InstallCallback(LPTMR_INSTANCE,lptmr_isr_callback);

    PRINTF("Low Power Timer Example\n\r");

    // Start counting
    LPTMR_DRV_Start(LPTMR_INSTANCE);
    while(1)
    {
        if(currentCounter != lptmrCounter)
        {
            currentCounter = lptmrCounter;
            PRINTF("LPTMR interrupt No.%d \r\n",currentCounter);
            LED1_TOGGLE;
        }
    }
}
```

freescale External Use | 11

Example Code Using Kinetis SDK

This is an example of the LPTMR configured to generate an interrupt every second. When the interrupt is triggered, LED1 changes status and information is printed to a terminal.

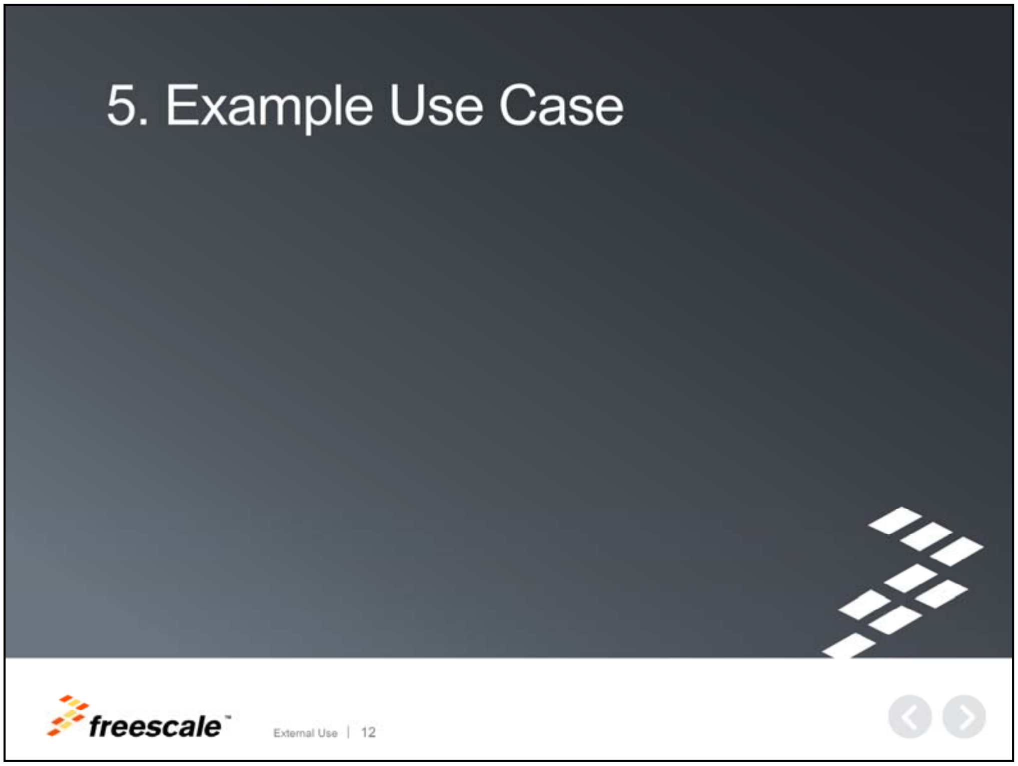
To initialize the LPTMR module, call the LPTMR_DRV_Init() function.

In order to use this function, it is necessary to define data structure lptmrUserConfig. This contains the LPTMR user configuration options.

In order to set the period, call the function `LPTMR_DRV_SetTimerPeriodUs (microseconds). This function` configures the LPTMR time period in microseconds, while the LPTMR is working as a time counter. After the time period, the callback function is called. This function cannot be called while the LPTMR is working as a pulse counter. The value in microseconds (us) should be an integer multiple of the clock source time slice.

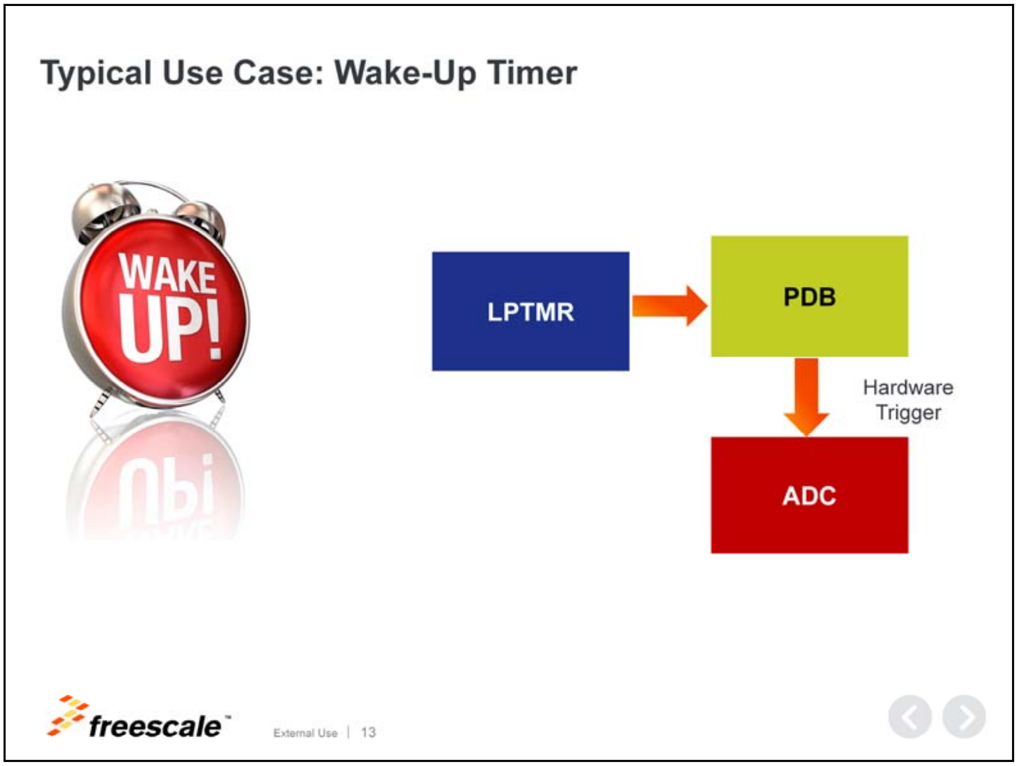
If the interrupt is enabled, then it is necessary to use the `LPTMR_DRV_InstallCallback function.` This function installs the user-defined callback in the LPTMR module. When an LPTMR interrupt request is served, the callback is executed inside the ISR.

Now we are ready to start the module. The LPTRM_DRV_Start function starts the



5. Example Use Case

Let’s review an example use case.



Typical Use Case: Wake-Up Timer

LPTMR → PDB

Hardware Trigger

ADC

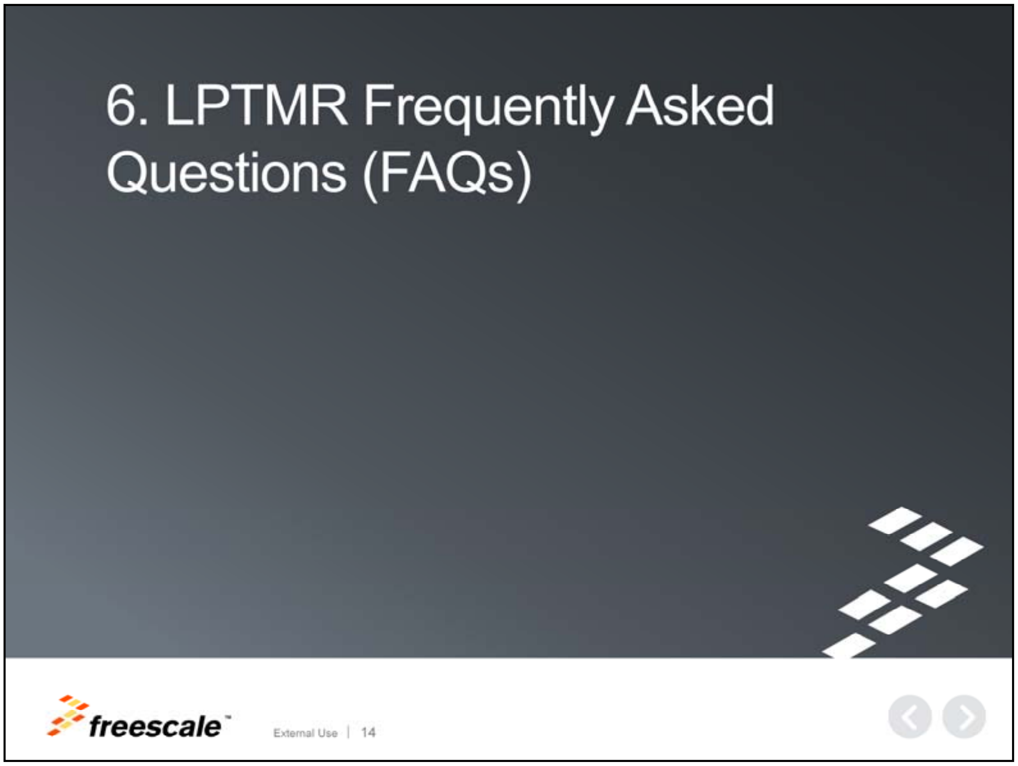
External Use | 13

Typical Use Case: Wake-Up Timer

The LPTMR is useful in low power applications. In this example, you'll see it is used as a wake-up timer to wake the part out of sleep modes after a certain amount of time.

If a clock is not available, the LPTMR can be used in pulse counter mode with the glitch filter enabled to wake-up the part based on pulse counts.

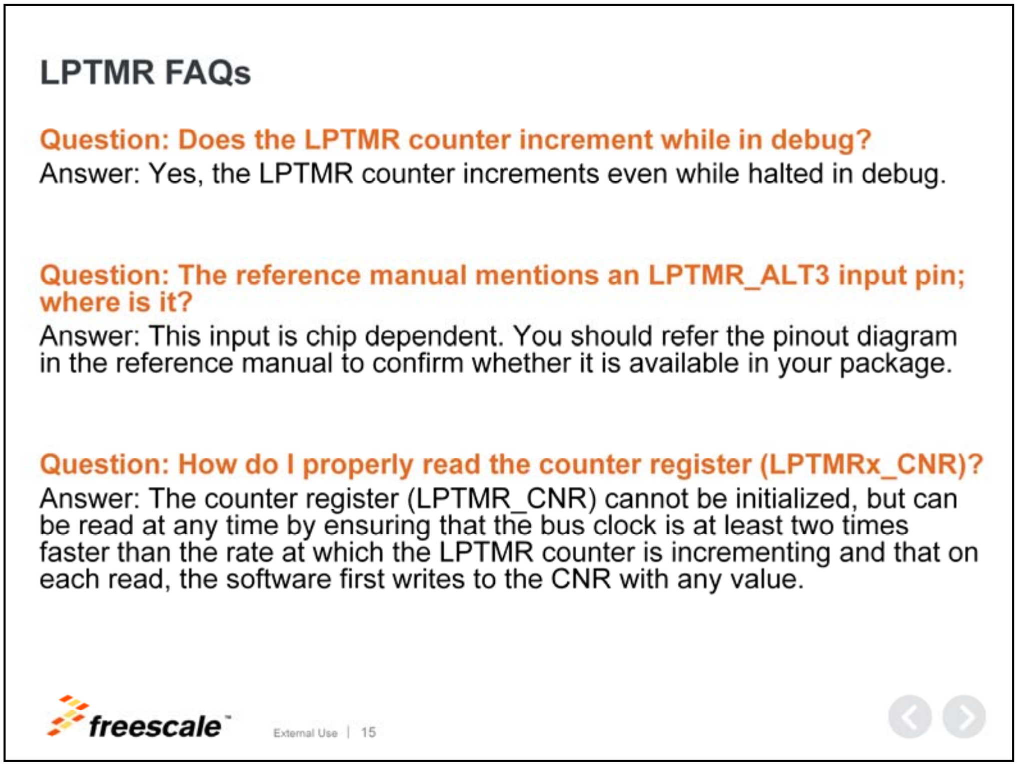
In addition the LPTMR can trigger events like an ADC conversion.



6. LPTMR Frequently Asked Questions (FAQs)

External Use | 14

Now, let’s review some frequently asked questions.



LPTMR FAQs

Question: Does the LPTMR counter increment while in debug?
Answer: Yes, the LPTMR counter increments even while halted in debug.

Question: The reference manual mentions an LPTMR_ALT3 input pin; where is it?
Answer: This input is chip dependent. You should refer the pinout diagram in the reference manual to confirm whether it is available in your package.

Question: How do I properly read the counter register (LPTMRx_CNR)?
Answer: The counter register (LPTMR_CNR) cannot be initialized, but can be read at any time by ensuring that the bus clock is at least two times faster than the rate at which the LPTMR counter is incrementing and that on each read, the software first writes to the CNR with any value.

External Use | 15

LPTMR FAQs

Question: Does the LPTMR counter increment while in debug?

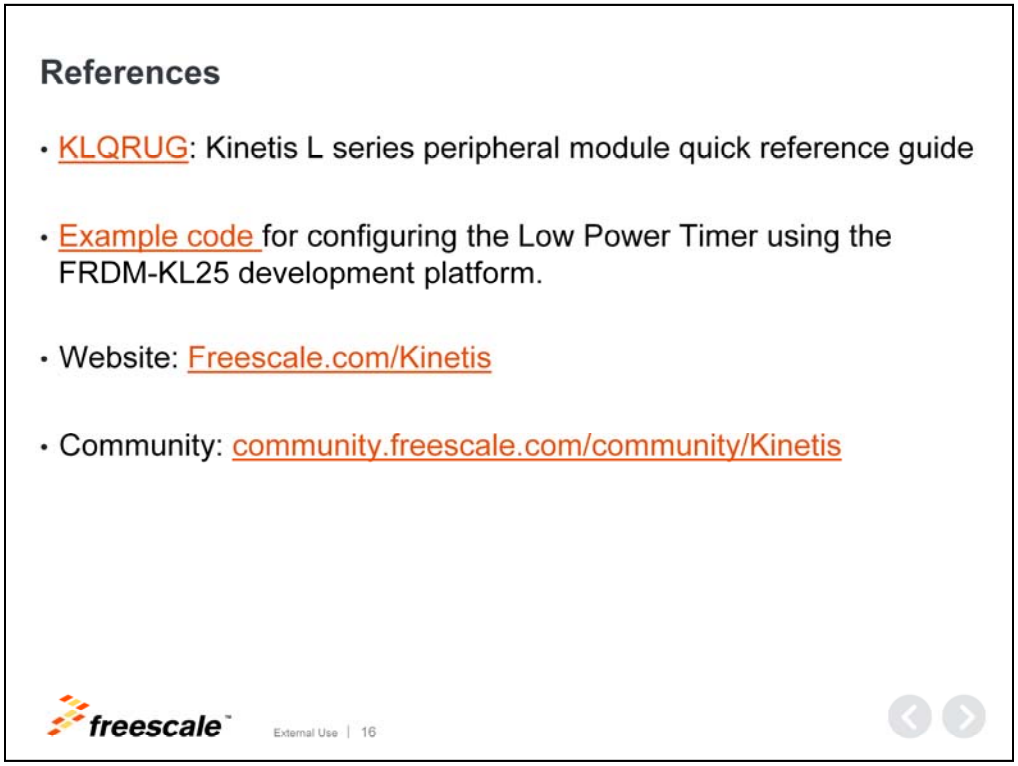
Answer: Yes, the LPTMR counter increments even while halted in debug.

Question: The reference manual mentions an LPTMR_ALT3 input pin; where is it?

Answer: This input is chip dependent. Please refer to the pinout diagram in the reference manual to confirm whether it is available in your package.

Question: How do I properly read the counter register (LPTMRx_CNR)?

Answer: The counter register (LPTMR_CNR) cannot be initialized, but can be read at any time by ensuring that the bus clock is at least two times faster than the rate at which the LPTMR counter is incrementing and that on each read, the software first writes to the CNR with any value.



References

- KLQRUG: Kinetis L series peripheral module quick reference guide
- Example code for configuring the Low Power Timer using the FRDM-KL25 development platform.
- Website: Freescale.com/Kinetis
- Community: community.freescale.com/community/Kinetis

External Use | 16

This concludes our presentation on the LPTMR module.

For more information on the LPTMR for Kinetis MCUs, please visit the references listed here.

We also invite you to visit us on the web at Freescale.com/Kinetis and check out our Kinetis community page.

freescale™

www.Freescale.com