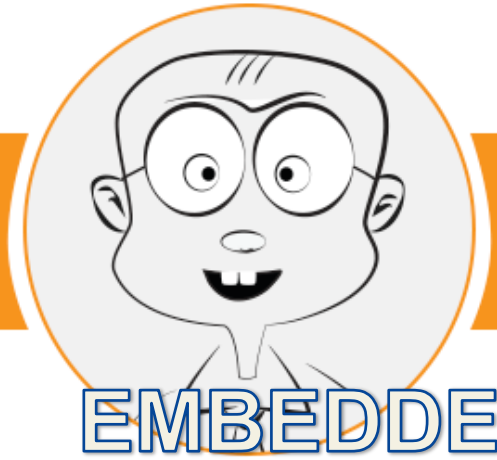


EMBEDDED SYSTEM COURSE

LECTURE 7: PERIPHERALS PIT TIMER

Learning Goals

- Understanding basis concepts about timer, specifically in KE16Z SoC.
- Understanding on how to configure the PIT module in KE16Z.

Table of contents

Table of contents

Overview on KE16Z Timer modules

- The KE16Z supports following timer modules:
  - **Timer/PWM Module (PWT)**: four channels timer which supports input capture, output compare, and the generation of PWM signals to control electric motor and power management applications. The counter, compare and capture registers are clocked by an asynchronous clock that can remain enabled in low power modes.
  - **Periodic Interrupt Timer (PIT)**
  - **Low power timer (LPTMR)**: 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. It can also continue operating through most system reset events, allowing it to be used as a time of day counter.
  - **Real Time Clock (RTC)**
- This lecture only focuses on PIT, the remaining modules are advance topics and target on student's self-study.

Table of contents

Periodic Interrupt Timer (PIT)

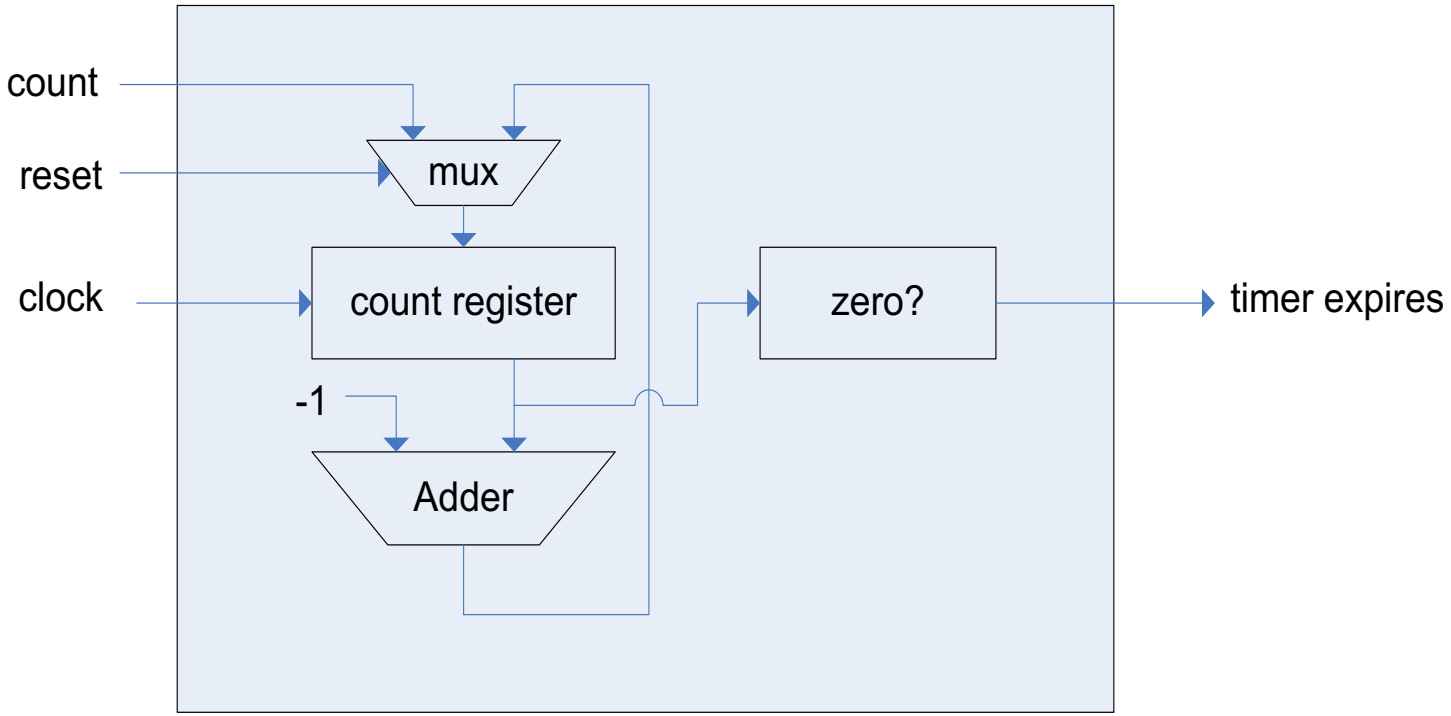
- PIT is an counter that generates an output signal when it reaches a programmed count. The output signal is often used to trigger an interrupt
- PIT may be one-shot or periodic.
  - One-shot timers will signal only once and then stop counting.
  - Periodic timers signal every time they reach a specific value and then restart, thus producing a signal at periodic intervals. Periodic timers are typically used to invoke activities that must be performed at regular intervals

Periodic Interrupt Timer (PIT) – cont.

- Typical applications:
  - To implement a clock
  - To check user input periodically
  - To monitor environment changes
  - To switch between programs
  - Count how many cycles/times the MCU has been elapsed since last restart.

Periodic Interrupt Timer (PIT) – cont.

- **Simple explanation**:
  - A timer is basically a counter of clock cycles.



Periodic Interrupt Timer (PIT) – cont.

- **Time Period**

  = (count + 1) × clock cycle time

  = (count + 1) / clock frequency

Periodic Interrupt Timer (PIT) – cont.

- **How to program a timer**?
  - Set up count value
  - Check if the timer expires
  - Configure interrupt, if interrupt is to be used
  - Read current value (if supported)

Periodic Interrupt Timer (PIT) – cont.

- **The KE16Z PIT register descriptions**

LPIT memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_7000	Version ID Register (LPIT0_VERID)	32	R	0100_0000h	35.4.1/758
4003_7004	Parameter Register (LPIT0_PARAM)	32	R	See section	35.4.2/759
4003_7008	Module Control Register (LPIT0_MCR)	32	R/W	0000_0000h	35.4.3/759
4003_700C	Module Status Register (LPIT0_MSR)	32	w1c	0000_0000h	35.4.4/760
4003_7010	Module Interrupt Enable Register (LPIT0_MIER)	32	R/W	0000_0000h	35.4.5/761
4003_7014	Set Timer Enable Register (LPIT0_SETTEN)	32	R/W	0000_0000h	35.4.6/763
4003_7018	Clear Timer Enable Register (LPIT0_CLRTEN)	32	W (always reads 0)	0000_0000h	35.4.7/764
4003_7020	Timer Value Register (LPIT0_TVAL0)	32	R/W	0000_0000h	35.4.8/765
4003_7024	Current Timer Value (LPIT0_CVAL0)	32	R	FFFF_FFFFh	35.4.9/766
4003_7028	Timer Control Register (LPIT0_TCTRL0)	32	R/W	0000_0000h	35.4.10/767

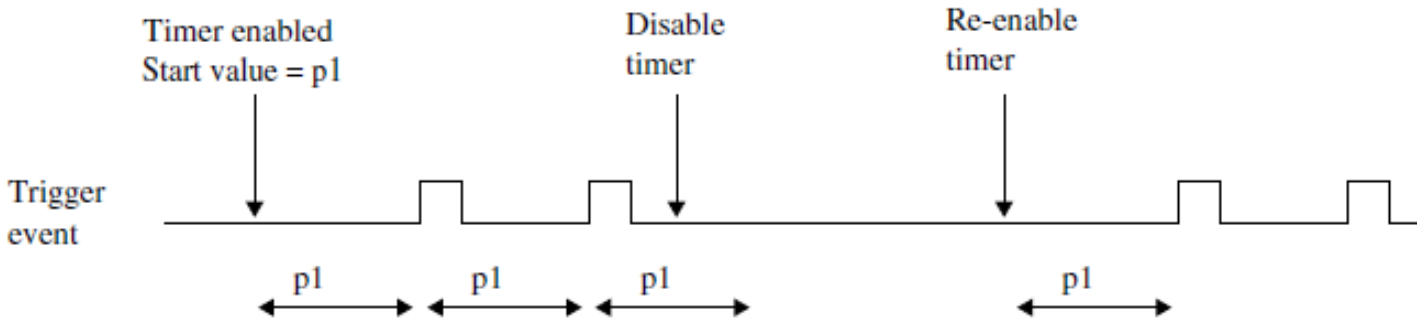
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_7030	Timer Value Register (LPIT0_TVAL1)	32	R/W	0000_0000h	35.4.8/765
4003_7034	Current Timer Value (LPIT0_CVAL1)	32	R	FFFF_FFFFh	35.4.9/766
4003_7038	Timer Control Register (LPIT0_TCTRL1)	32	R/W	0000_0000h	35.4.10/767

Periodic Interrupt Timer (PIT) – cont.

- **The KE16Z PIT register descriptions**
  - **PIT Module Control Register (PIT_MCR)** : enable/disable the PIT timer clock and control the timer when PIT enters debug mode
  - **Timer Value Register (PIT_TVALn)**: set timeout counter number
  - **Current Timer Value Register (PIT_CVALn)**: indicate the current timer position
  - **Timer Control Register (PIT_TCTRLn)**: configure the specific timer enable/disable, setup the chain mode, timer interrupt enable
  - **Module Status Register (PIT_MSR)**: holds that status of the timer interrupt flag.
  - **Timer Control Register (PIT_TCTRLn):** control timer channel register
  - **Module Interrupt Enable Register(PIT_MIER):** interrupt control timer

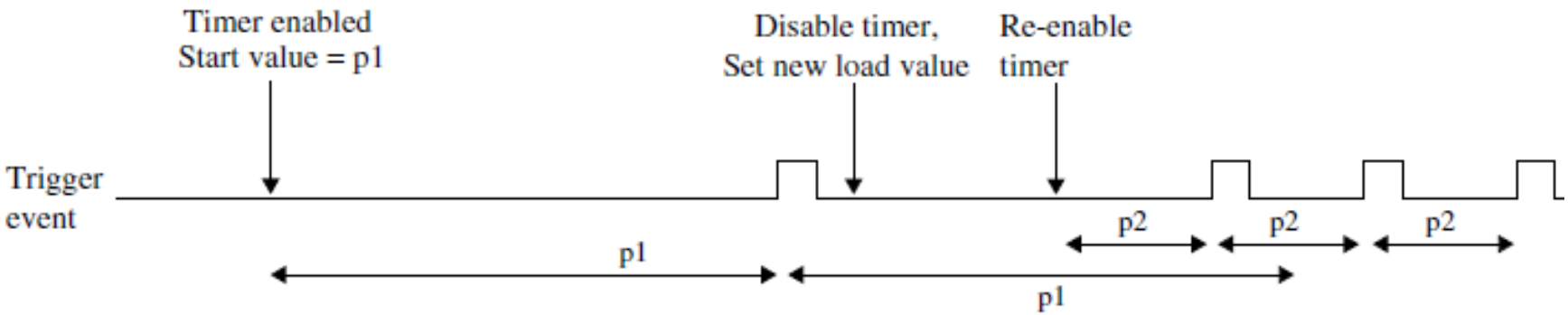
Periodic Interrupt Timer (PIT) – cont.

- **Some scenarios on setting the timer counter**
  - The counter period can be restarted, by first disabling, and then enabling the timer with TCTRLn[TEN].



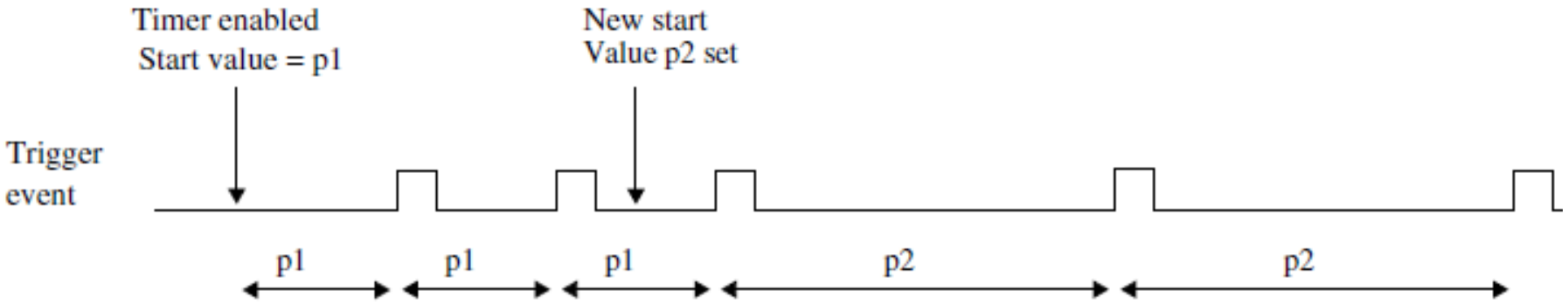
Periodic Interrupt Timer (PIT) – cont.

- **Some scenarios on setting the timer counter**
  - The counter period of a running timer can be modified, by first disabling the timer, setting a new load value, and then enabling the timer again



Periodic Interrupt Timer (PIT) – cont.

- **Some scenarios on setting the timer counter**
  - It is also possible to change the counter period without restarting the timer by writing TVAL with the new load value. This value will then be loaded after the next trigger event



Periodic Interrupt Timer (PIT) – cont.

- **Some scenarios on setting the timer counter**
  - If we want to setup a timer with timeout period > maximum timer counter:

$$\texttt{Expect timeout} > 2^{32}/f_{PIT}$$

  - In this case we can consider to use the **chain timer** (timer2 counter will be started to count when timer1 counter reached to zero)

Periodic Interrupt Timer (PIT) – cont.

❖**Some examples**

– please refer to 35.6 Usage guide section at reference manual

Copyright

- This course including **Lecture Presentations**, **Quiz**, **Mock Project**, **Syllabus**, **Assignments**, **Answers** are copyright by FPT Software Corporation.
- This course also uses some information from external sources and non-confidential training document from Freescale, those materials comply with the original source licenses.