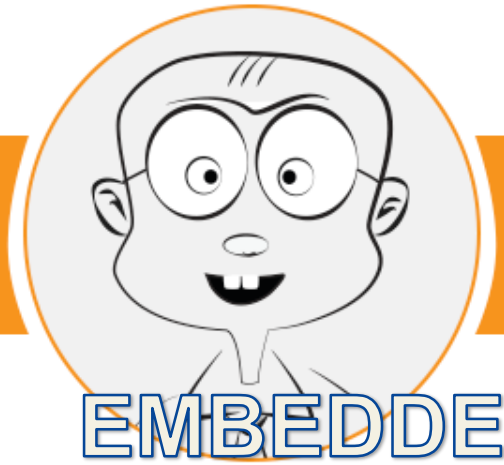


EMBEDDED SYSTEM COURSE

LECTURE 7: PERIPHERALS PIT TIMER

Learning Goals

- Understanding basis concepts about timer, specifically in KL46 SoC.
- Understanding on how to configure the PIT module in KL46.
- Understanding on how to configure the Real Time Clock – RTC.

Table of contents

Table of contents

Overview on KL46 Timer modules

- The KL46 supports following timer modules:
  - **Timer/PWM Module (PWM)**: two to eight channel 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. Refer link.

Table of contents

Periodic Interrupt Timer (PIT)

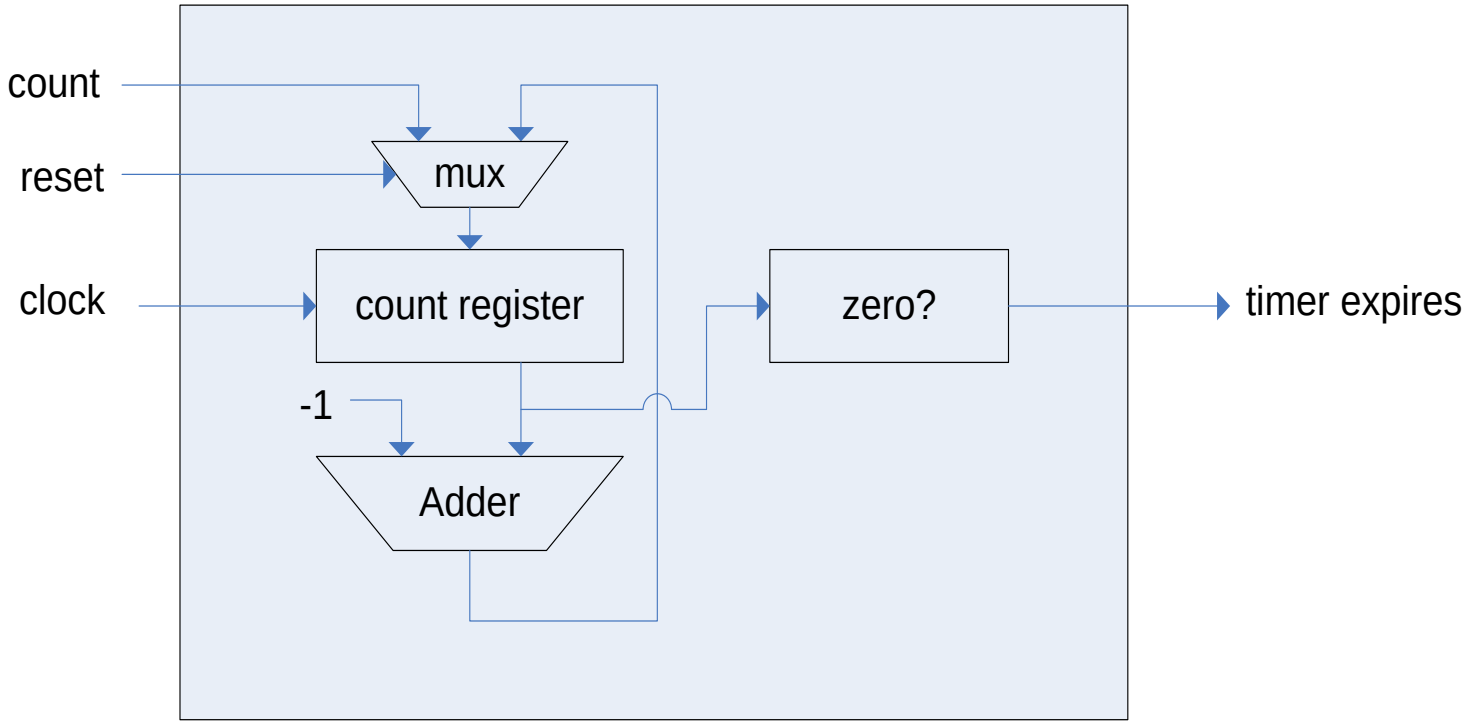
- PIT is an counter that generates an output signal when it reaches a programmed count. The output signal is often used to trigger an interrupt
- PIT may be one-shot or periodic.
  - One-shot timers will signal only once and then stop counting.
  - Periodic timers signal every time they reach a specific 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

EX: The clock frequency is 50MHz.

The needed time period is 5.1ms.

What is the count value?

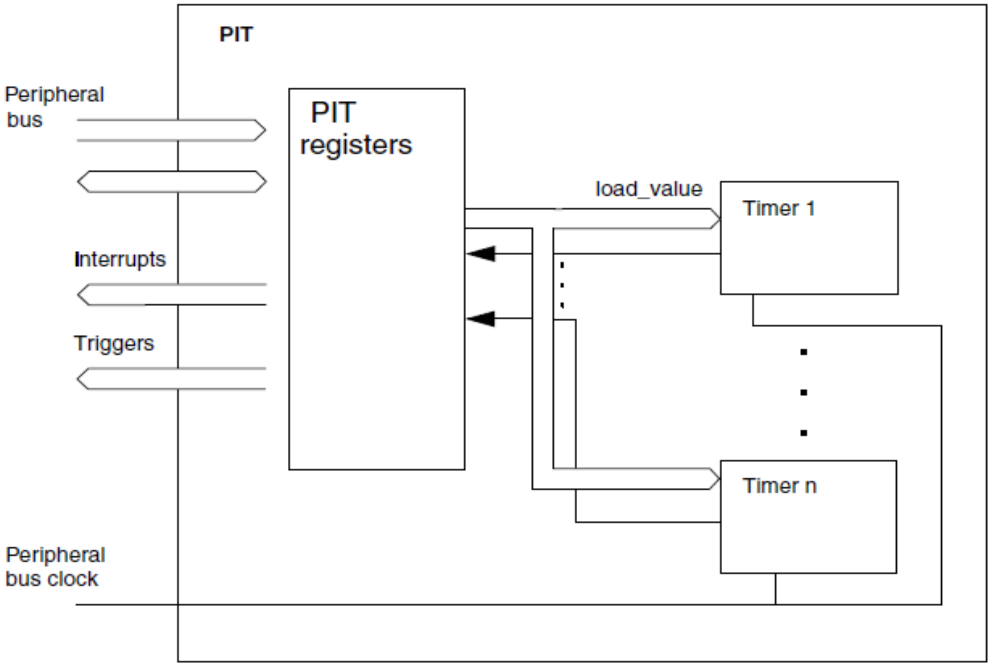
Periodic Interrupt Timer (PIT) – cont.

- **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.

❖ **KL46 PIT Module**

- The KL46 PIT module is an array of two timers that can be used to **raise interrupts** and **trigger DMA** channels.
- As its name: “Periodic timers” the KL46 PIT timer is automatically restart when counter reaching to zero
- The KL46 PIT block diagram



Periodic Interrupt Timer (PIT) – cont.

- **The KL46 PIT register descriptions**

PIT memory map

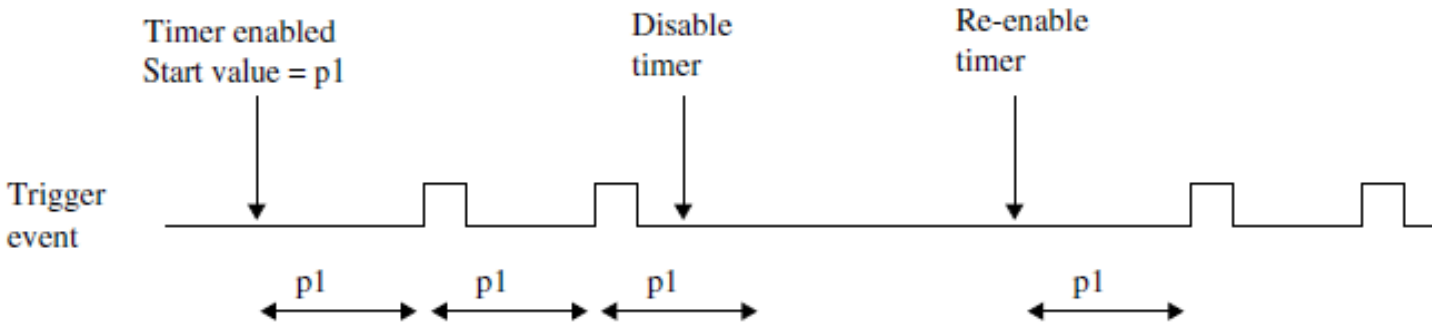
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
4003_7000	PIT Module Control Register (PIT_MCR)	32	R/W	0000_0006h
4003_70E0	PIT Upper Lifetime Timer Register (PIT_LTMR64H)	32	R	0000_0000h
4003_70E4	PIT Lower Lifetime Timer Register (PIT_LTMR64L)	32	R	0000_0000h
4003_7100	Timer Load Value Register (PIT_LDVAL0)	32	R/W	0000_0000h
4003_7104	Current Timer Value Register (PIT_CVAL0)	32	R	0000_0000h
4003_7108	Timer Control Register (PIT_TCTRL0)	32	R/W	0000_0000h
4003_710C	Timer Flag Register (PIT_TFLG0)	32	R/W	0000_0000h
4003_7110	Timer Load Value Register (PIT_LDVAL1)	32	R/W	0000_0000h
4003_7114	Current Timer Value Register (PIT_CVAL1)	32	R	0000_0000h
4003_7118	Timer Control Register (PIT_TCTRL1)	32	R/W	0000_0000h
4003_711C	Timer Flag Register (PIT_TFLG1)	32	R/W	0000_0000h

Periodic Interrupt Timer (PIT) – cont.

- **The KL46 PIT register descriptions**
  - **PIT Module Control Register (PIT_MCR)** : enable/disable the PIT timer clock and control the timer when PIT enters debug mode
  - **PIT Upper/Lower Lifetime Timer Register (PIT_LTMR64H PIT_LTMR64L)**: counting the MCU lifetime (number of cycle has been elapsed from last restart)
  - **Timer Load Value Register (PIT_LDVALn)**: 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
  - **Timer Flag Register (PIT_TFLG)**: holds that status of the timer interrupt flag.

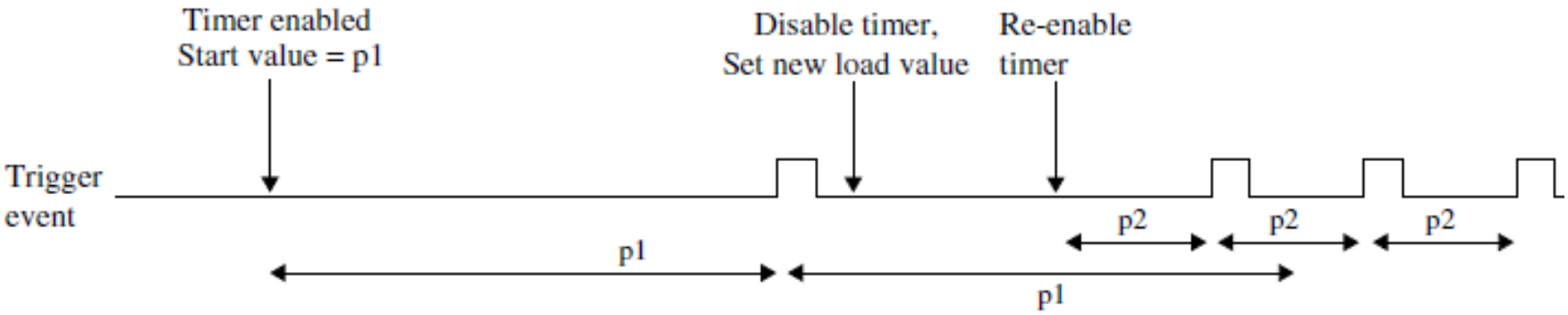
Periodic Interrupt Timer (PIT) – cont.

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



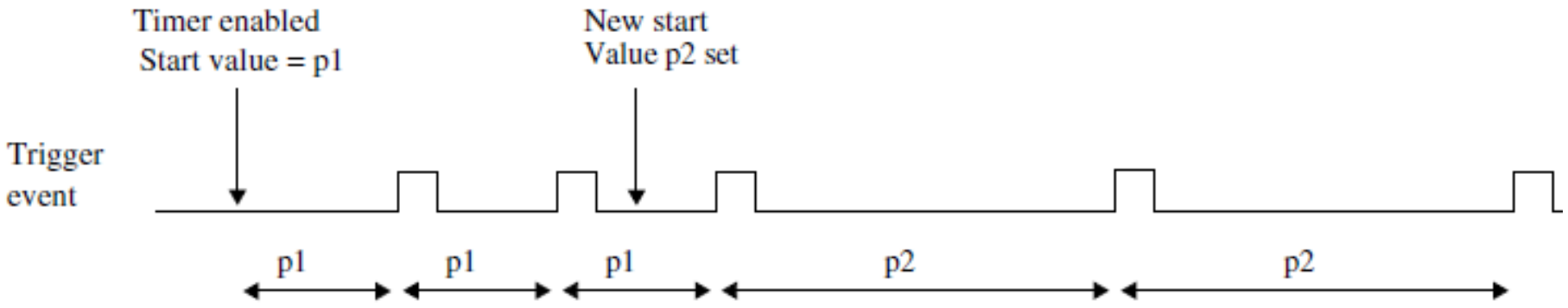
Periodic Interrupt Timer (PIT) – cont.

- **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 LDVAL 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**

- The PIT clock has a frequency of 50 MHz.
- Timer 1 creates an interrupt every 5.12 ms.
- Timer 2 creates an interrupt every 30 ms.

```
// turn on PIT
PIT_MCR = 0x00;
// Timer 1
PIT_LDVAL1 = 0x0003E7FF; // setup timer 1 for
256000 cycles
PIT_TCTRL1 = TIE; // enable Timer 1 interrupts
PIT_TCTRL1 |= TEN; // start Timer 1
// Timer 2
PIT_LDVAL2 = 0x0016E35F; // setup timer 2 for
1500000 cycles
PIT_TCTRL2 |= TEN; // start Timer 2
```

Periodic Interrupt Timer (PIT) – cont.

- **Some examples**
  - The PIT clock has a frequency of 100 MHz.
  - Two timer module are available
  - An interrupt shall be raised every **1 hour.**

```
// turn on PIT
PIT_MCR = 0x00;
// Timer 1
PIT_LDVAL1 = 0x23C345FF;0x00000009; // setup timer 1 for 600 000 000 cycles
PIT_TCTRL1 = TIE; // enable Timer 1 interrupts
PIT_TCTRL1 |= TEN; // start Timer 1
// Timer 2
PIT_LDVAL3 = 0x00000009;; // setup timer 2 for 10 cycles
PIT_TCTRL2 = TIE; // enable Timer 2 interrupt
PIT_TCTRL2 |= CHN; // chain Timer 2 to Timer 1
PIT_TCTRL2 |= TEN; // start Timer 2
```

Table of contents

Real Time Clock – RTC

- In modern device today, almost has clock which can be display in LCD with time and date information.
- The Real Time Clock – RTC is used to maintain and control the date & time which runs off its own battery.
- The RTC also used to other functions including alarm and event scheduling.

Real Time Clock – RTC (con.t)

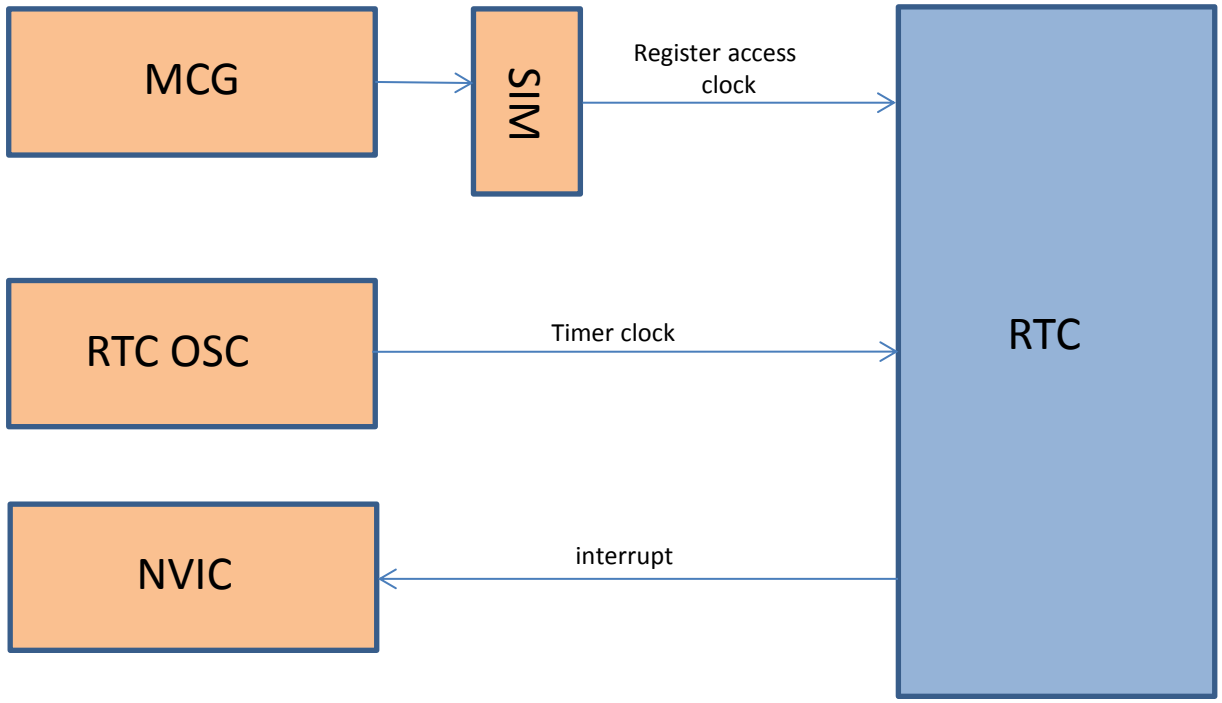
❖KL46 RTC Module

- The RTC module features include:
  - 32-bit seconds counter with roll-over protection and 32-bit alarm
  - 16-bit prescaler with compensation that can correct errors between 0.12 ppm and 3906 ppm
  - Register write protection: Lock register requires POR or software reset to enable write access
  - 1 Hz square wave output
  - The RTC remains functional in all low power modes and can generate an interrupt to exit any low power mode.

Real Time Clock – RTC (con.t)

❖KL46 RTC Module

- Internal connection



Real Time Clock – RTC (con.t)

❖ KL46 RTC Module

- Registers:
  - All registers must be accessed using 32-bit writes and all register accesses incur three wait states.
  - Write accesses to any register by non-supervisor mode software, when the supervisor access bit in the control register is clear, will terminate with a bus error.

RTC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
4003_D000	RTC Time Seconds Register (RTC_TSR)	32	R/W	0000_0000h
4003_D004	RTC Time Prescaler Register (RTC_TPR)	32	R/W	0000_0000h
4003_D008	RTC Time Alarm Register (RTC_TAR)	32	R/W	0000_0000h
4003_D00C	RTC Time Compensation Register (RTC_TCR)	32	R/W	0000_0000h
4003_D010	RTC Control Register (RTC_CR)	32	R/W	0000_0000h
4003_D014	RTC Status Register (RTC_SR)	32	R/W	0000_0001h
4003_D018	RTC Lock Register (RTC_LR)	32	R/W	0000_00FFh
4003_D01C	RTC Interrupt Enable Register (RTC_IER)	32	R/W	0000_0007h

Real Time Clock – RTC (con.t)

❖ KL46 RTC Module – Step to configure the RTC

- In system powered-up state
  - Configure the SIM for access to the RTC register block

```
/*enable the clock to SRTC module register space*/
SIM_SCGC6 = SIM_SCGC6_RTC_MASK;
```

- After VBAT POR or software reset
  - Enable the oscillator

```
RTC_CR |= RTC_CR_OSCE_MASK;
```

  - The time invalid flag is set and must be cleared by writing the seconds register to a valid value

```
RTC_TSR = 0x00000005;
```

  - Enable the counter

```
RTC_SR |= RTC_SR_TCE_MASK;
```

Real Time Clock – RTC (con.t)

❖ KL46 RTC Module – Step to configure the RTC

- Using time compensation use the RTC_TCR register
  - Select compensation interval value (1 to 256)
  - Select time compensation register value (-127 to +128)

```
RTC_TCR = 0x0000_0100;  //apply every 2 seconds
RTC_TCR |= 0x0000_0080; // add 128 cycles to the prescaler register
```

- To lock the Status, Control, or Time Compensation Registers use the RTC Lock register (hard lock).
  - Can only be cleared on Vbat POR or software reset via RTC_CR

```
RTC_LR  = 0x00000010; // lock SR, CR, and TCR. Will require VBAT POR
```

Summary

- Understanding about the basic concepts regarding periodic timer and especially the KL46 PIT and RTC timer modules.
- Understanding on how to setup a simple timer and a chain timer.
- Understanding on how to setup a Real Time Clock

Question & Answer

Thanks for your attention !

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.