**PROJECT 2:** **TRAFFIC LIGHTS**

# Objectives

* Continue working on General Purpose Input/Output (GPIO) of ARM.
* Continue working with real electronic devices: Led, Led 7 Segments.
* Start working with General Purpose Timer of Stellaris board.

Introduction

0

0

0

0

When reset system, counter shows value 7 and only green light is turned on. Then the counter counts down from 7 to 0 in frequency 1Hz. After that, the green light is turned off and the yellow light is turned on for 3 seconds. Next turn, red light is turn on and the counter displays value 9. The counter will count down until it gets value 0. At this time, the system will repeat all above cycle.

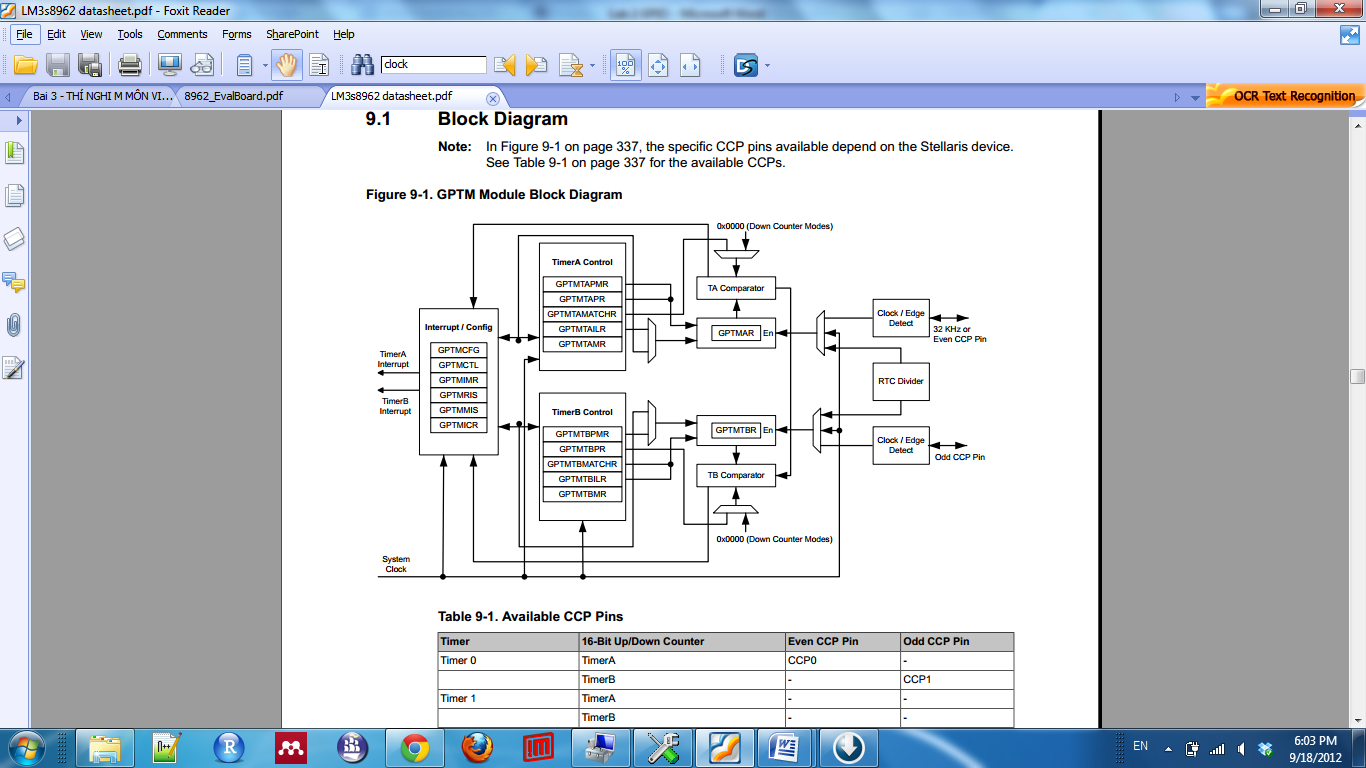
This is only demo a traffic light system so that it is used only for 1 line. The project doesn’t have any inputs. It has 3 outputs to the led and 7 outputs for one led 7 segments.

# General Purpose Timers

This is a programmable timer that can be used to count or time external events that drive the input pins. The Stellaris General-Purpose Timer Module (GPTM) contains four GPTM blocks (Timer0, Timer1, Timer2, and Timer 3). Each GPTM block provides two 16-bit timers/counters (referred to as TimerA and TimerB) that can be configured to operate independently as timers or event counters, or configured to operate as one 32-bit timer or one 32-bit Real-Time Clock (RTC). (Page 337 Datasheet of LM3S8962)

**The General-Purpose Timers provide the following features:**

* Four General-Purpose Timer Modules (GPTM), each of which provides two 16-bit timers/counters. Each GPTM can be configured to operate independently:
  + As a single 32-bit timer
  + As one 32-bit Real-Time Clock (RTC) to event capture
  + For Pulse Width Modulation (PWM)
  + To trigger analog-to-digital conversions
* 32-bit Timer modes
  + Programmable one-shot timer
  + Programmable periodic timer
  + Real-Time Clock when using an external 32.768-KHz clock as the input
  + User-enabled stalling when the controller asserts CPU Halt flag during debug
  + ADC event trigger
* 16-bit Timer modes
  + General-purpose timer function with an 8-bit prescaler (for one-shot and periodic modes only)
  + Programmable one-shot timer
  + Programmable periodic timer
  + User-enabled stalling when the controller asserts CPU Halt flag during debug
  + ADC event trigger
* 16-bit Input Capture modes
  + Input edge count capture
  + Input edge time capture
* 16-bit PWM mode
  + Simple PWM mode with software-programmable output inversion of the PWM signal



When software writes the TAEN bit in the GPTM Control (GPTMCTL) register (see page 354), the timer begins counting down from its preloaded value. Once the 0x0000.0000 state is reached, the timer reloads its start value from the concatenated GPTMTAILR on the next cycle. If configured to be a one-shot timer, the timer stops counting and clears the TAEN bit in the GPTMCTL register. If configured as a periodic timer, it continues counting. Depend on how long we want to generate from the counter, we can configure each timer to **32 bit One Shot/Periodic Timer mode** or **16 bit One Shot/Periodic Timer mode.**

# Setting System Clock

For timing exactly, system clock should be configured correctly. There are many ways to configure the system clock (datasheet page 181). But in our Lab, we can use external 8MHz input from Main Oscillator and configure the System Clock to 8MHz that mean each cycle machine process in 1/8MHz = 125ns.

*SysCtlClockSet (SYSCTL\_SYSDIV\_1 | SYSCTL\_USE\_OSC | SYSCTL\_OSC\_MAIN | SYSCTL\_XTAL\_8MHZ);*

To use this system function, we have to declare library of **system controller** at the beginning of the source code.

*include “driverlib/sysctl.h”*

**Experiment 1**: Blinking the Led at exactly frequency 1Hz using timer.

* Create Project Lab2a.
* Copy file Lab2a to the project. Each system clock to count down the timer is 8Mhz need to preload count-down counter value *8.000.000* or *0x7A1200*. So that in this file, we need to configure using Timer 0 in **32 bit Periodic Timer Mode**.
  + Ensure the timer is disabled (the **TAEN** bit in the **GPTMCTL register** is cleared) before making any changes.
  + Write the **GPTM Configuration Register (GPTMCFG**) with a value of 0x0.
  + Set the **TAMR** field in the **GPTM TimerA Mode Register (GPTMTAMR**). Write a value of 0x2 for Periodic mode.
  + Load the start value 0x7A1200 into the **GPTM TimerA Interval Load Register (GPTMTAILR).**
  + Set the **TAEN** bit in the **GPTMCTL register** to enable the timer and start counting.
  + Poll the **TATORIS** bit in the **GPTMRIS register**. The status flags are cleared by writing a 1 to the **TATOCINT** bit of the **GPTM Interrupt Clear Register (GPTMICR).**
* Run, analysis and answer questions:
  + Explain below code. When do we use them?

*// Enable the TIMER 0*

*SYSCTL\_RCGC1\_R = SYSCTL\_RCGC1\_TIMER0;*

*// Clear TAEN bit of GPTMCTL Register*

*TIMER0\_CTL\_R &= ~0x0001;*

*// Write to GPTMCFG value 0x0*

*TIMER0\_CFG\_R = 0x0;*

*// Set TARM field in the GPTMTAMR.*

*TIMER0\_TAMR\_R |= 0x00; // 0x1;*

*// Load value 0x7A1200/2 = 0x3D0900 into GPTMTAILR.*

*TIMER0\_TAILR\_R = 0x003D0900;*

*// Start the timer count down*

*TIMER0\_CTL\_R |= 0x0001;*

*TIMER0\_ICR\_R |= 0x01;*

**Experiment 2: Simple traffic light**

* Create new project Lab2b
* Rewrite the code for 1 traffic light project without Led 7segments. 3 led works in rule: 1 led turn on in 7 second, then it turns off. Then the Led 2 turn on in 3 second then it turn off. Last turn, Led 3 is turn on in 10 seconds then it turns off. Repeat the cycle for this 3 Leds.

**Experiment 3: Full demo traffic light**

* Create the new project Lab 2c.
* Implement a traffic light with a Led 7 segment. The number displayed on 7 segment Led counts down follow the rule in Experiment 2. Note: In the Rapid board, Led 7 segments turn on when the input signal is high.

