**PROJECT 1: WORKING ON GPIO PORT OF ARM**

**CORTEX-M3 LM3S8962**

# **Prepare:**

* Install software IAR Embedded Workbench IDE
* Install driver for Stellaris EKI-LM3S8962 Evaluation Kit.

# **Document**

* User manual of Stellaris EKI-LM3S8962 Evaluation Kit.
* User Guide Stellaris Peripheral Driver Library.
* Datasheet of LM3S8962
* Cortex M3 Instruction Set Technical User Manual.
* IAR C/C++ Development Guide

# **Objectives**

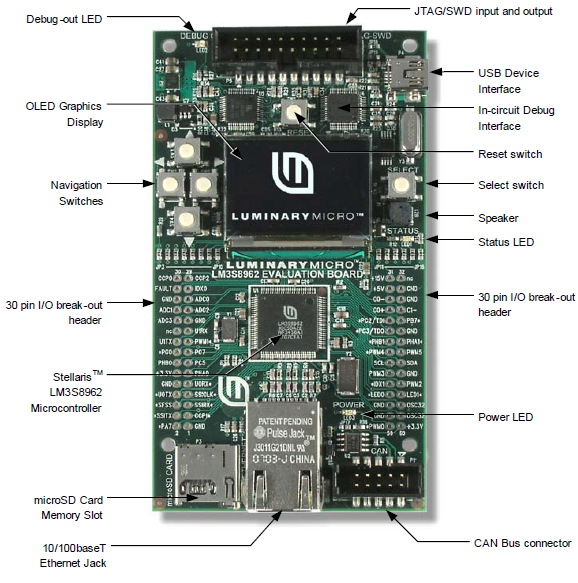
* Understand **System Control** registers and **General Purpose Input/Outputs** registers of ARM Cortex M3
* Understand how to programming ARM in C and in Assembly.
* Working with Stellaris Peripheral Driver Library.

# **Introduction**

**LM3S8962 Microcontroller**

* 32-bit RISC performance using ARM® Cortex™-M3 v7M architecture
* 50-MHz operation
* 42 interrupt channels with eight priority levels
* 256-KB single-cycle Flash, 64-KB single-cycle SRAM
* Four general-purpose 32-bit timers
* Integrated Ethernet MAC and PHY
* Controller area network (CAN) module
* Three fully programmable 16C550-type UARTs
* Four 10-bit ADC channels (inputs) when used as single-ended inputs
* One integrated analog comparator
* One I2C module
* Two PWM generator blocks
* One 16-bit counter
* Two comparators
* Produces two independent PWM signals
* One dead-band generator
* Two QEI modules with position integrator for tracking encoder position
* Two synchronous serial interfaces (SSIs)
* 0 to 42 GPIOs, depending on user configuration
* On-chip low drop-out (LDO) voltage regulator

|  |  |
| --- | --- |
| **Microcontroller Pin** | **Functions** |
| Pin 26 PA0/U0RX | Virtual COM port receive |
| Pin 27 PA1/U0TX | Virtual COM port transmit |
| Pin 19 PG0 | SD card chip select |
| Pin 30 PA4/SSI0RX | SD card data out |
| Pin 31 PA5/SSI0TX | SD card and OLED display data in |
| Pin 28 PA2/SSI0CLK | SD card and OLED display clock |
| Pin 34 PA6/CCP1 | OLED display data/control select |
| Pin 19 PG0 | OLED display chip select |
| Pin 18 PG1/PWM1 | Sound |
| Pin 61 PF1/IDX1 | Select switch |
| Pin 72 PE0/PWM4 | Up switch |
| Pin 74 PE2/PHB1 | Left switch |
| Pin 75 PE3/PHA1 | Right switch |
| Pin 73 PE1/PWM5 | Down switch |
| Pin 47 PF0/PWM0 | User LED |

 **EKI-LM3S8962 Evaluation Board**

To use the GPIO, the peripheral clock must be enabled by setting the appropriate GPIO Port bit field (GPIOn) in the RCGC2 register.

***RCGCn****: Run Mode Clock Gating Control Register (Datasheet of LM3S8962 - page 129).*

This register controls the clock gating logic. Each bit controls a clock enable for a given interface, function, or unit. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled (saving power).

***GPIODEN****: GPIO Digital Enable****GPIODIR****: GPIO Direction*

***GPIOPUR****: GPIO Pull-Up Resistor*

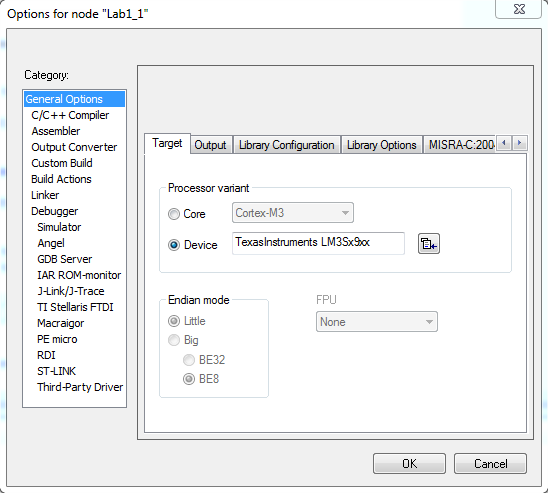
# **Experiment 1: Turn on User Led at Pin 47 PF0/PWM0.**

1. ***Create new Project in IAR.***

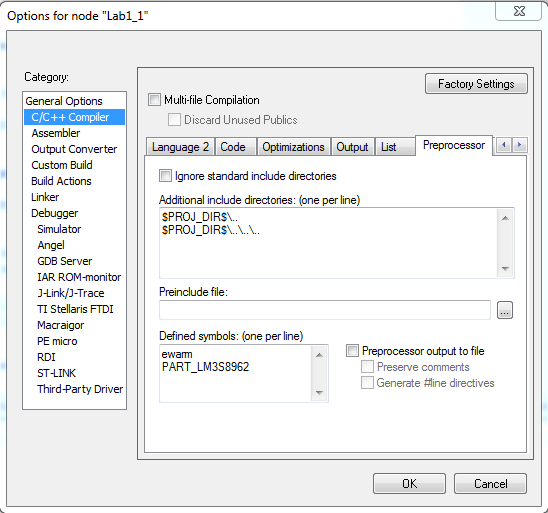
Open IAR Embedded Workbench IDE. Select Workspace in direction *C:\StellarisWare\boards\ek-lm3s8962*. Generate new project in IAR. Create new folder for the Project named Lab1a and give the project a name call Lab1a. Write below C code into file main.c.

|  |
| --- |
| //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  // Simple example to blink the on-board LED.  //  // Copyright (c) 2007-2010 Texas Instruments Incorporated. All rights reserved.  // Software License Agreement  //  // Texas Instruments (TI) is supplying this software **for** use solely and  // exclusively on TI's microcontroller products. The software is owned by  // TI and/or its suppliers, and is protected under applicable copyright  // laws. You may not combine this software with "viral" open-source  // software in order to form a larger program.  //  // THIS SOFTWARE IS PROVIDED "AS IS" AND WITH ALL FAULTS.  // NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT  // NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR  // A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. TI SHALL NOT, UNDER ANY  // CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL, OR CONSEQUENTIAL  // DAMAGES, FOR ANY REASON WHATSOEVER.  //  // This is part of revision 6288 of the EK-LM3S8962 Firmware Package.  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **#include "inc/lm3s8962.h"**  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // A very simple example that blinks the on-board LED.  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **int** main(**void**)  **{**  **volatile** **unsigned** **long** ulLoop;  // Enable the GPIO port that is used **for** the on-board LED.  SYSCTL\_RCGC2\_R = SYSCTL\_RCGC2\_GPIOF;  // Do a dummy read to insert a few cycles after enabling the peripheral.  ulLoop = SYSCTL\_RCGC2\_R;  // Enable the GPIO pin **for** the LED (PF0). Set the direction as output, and  // enable the GPIO pin **for** digital function.  GPIO\_PORTF\_DIR\_R = 0x01;  GPIO\_PORTF\_DEN\_R = 0x01;  // Loop forever.  **while**(1)  **{**  // Turn on the LED.  GPIO\_PORTF\_DATA\_R |= 0x01;  // Delay **for** a bit.  **for**(ulLoop = 0; ulLoop < 200000; ulLoop++)  **{**  **}**  // Turn off the LED.  GPIO\_PORTF\_DATA\_R &= ~(0x01);  // Delay **for** a bit.  **for**(ulLoop = 0; ulLoop < 200000; ulLoop++)  **{**  **}**  **}**  **}** |

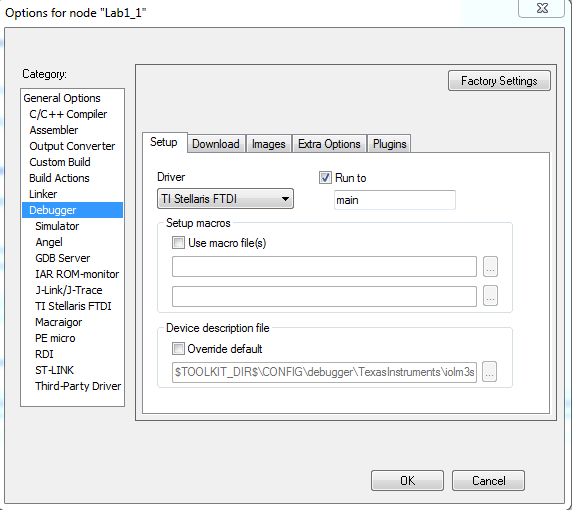
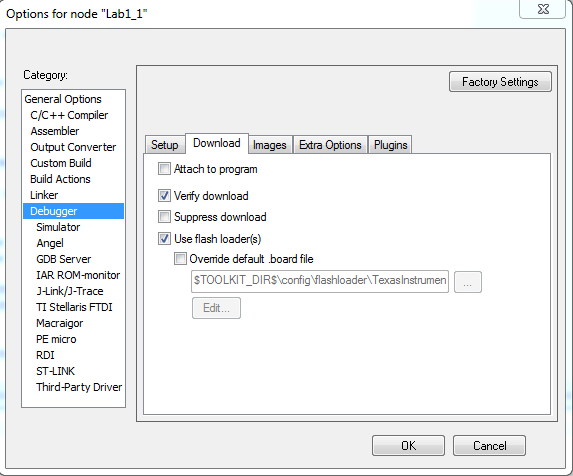
For the project can run, some addition steps need to be done in IAR. Add the startup\_ewarm.c to the project. You do it by right clicking in the project Lab1a, select Add\Add file.. and find the starup\_ewarm.c in folder C:\StellarisWare\boards\ek-lm3s8962\blinky. This file is need for IAR compilation.

Next,right click to the project again and select Option. Follow these steps to build the correct environment to EK-LM3S8962 Board.

Select right device in General Option Tab, the TexasInstruments LM3Sx9xx.

In tab C/C++ Compiler, set the Addition include directory and the Defined symbols

Do the same steps with Preprocessor tab in Assembler.



In debuger tab, select driver TI Stelaris for Setup tab, and Verify and Use flash for Download tab as figures above.

Now, use short cut Ctrl-D to compile project and to download to the board. Verify its function in the board. The Green Led should blinking in the board.

1. ***Analysis assembly code.***

In Debug mode, turn on window memory by select menu View\Memory. In memory window, select view mode 4x unit and Little Endian, open View\Disassembly and open View\Register.

Now, using F10 key to jump to next intrustion in debug mode. For each step, pay attention in to Disassembly and Registers window to see the effect of each intrustion to the registers. Try to understand what is each line of the C code mean. All high case variables in C code are declare in file lm3s8962.h. Open this file by right click to the first line and select *Open "inc/lm3s8962.h"*.

1. ***Questions:***

* What is the SYSCTL\_RCGC2\_R ? (Tips: search in lm3s8962.h and refer to datasheet of lm3s8962)
* What are SYSCTL\_RCGC2\_R, GPIO\_PORTF\_DIR\_R, GPIO\_PORTF\_DEN\_R and GPIO\_PORTF\_DATA\_R?
* Write out the mapped memory address of SYSCTL\_RCGC2\_R, SYSCTL\_RCGC2\_R, GPIO\_PORTF\_DIR\_R, GPIO\_PORTF\_DEN\_R and GPIO\_PORTF\_DATA\_R.
* In Disassembly window, write out the assembly code of the C intrustion

GPIO\_PORTF\_DATA\_R |= 0x01. Explain in detail the mean of each assembly code. Why this intrustion can write logic 1 to the output port ?

**Note:**

* There are many instructions that can generate either a 16-bit encoding or a 32-bit encoding depending on the operands and destination register specified. For some of these instructions, you can force a specific instruction size by using an instruction width suffix. The .W suffix forces a 32-bit instruction encoding. The .N suffix forces a 16-bit instruction encoding.

# **Experiment 2: Using button select (Pin 61 PF1/IDX1) to blink the LED (Pin 47 PF0/PWM0)**

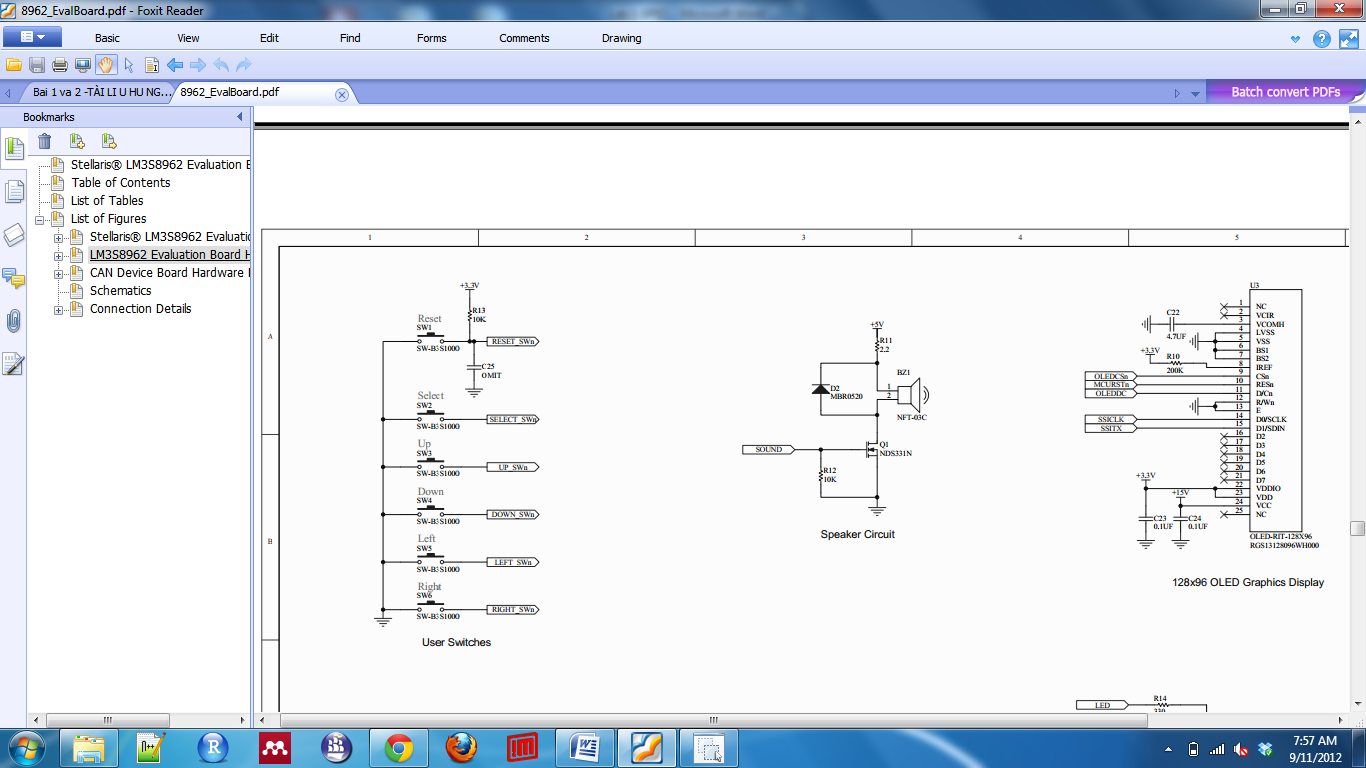
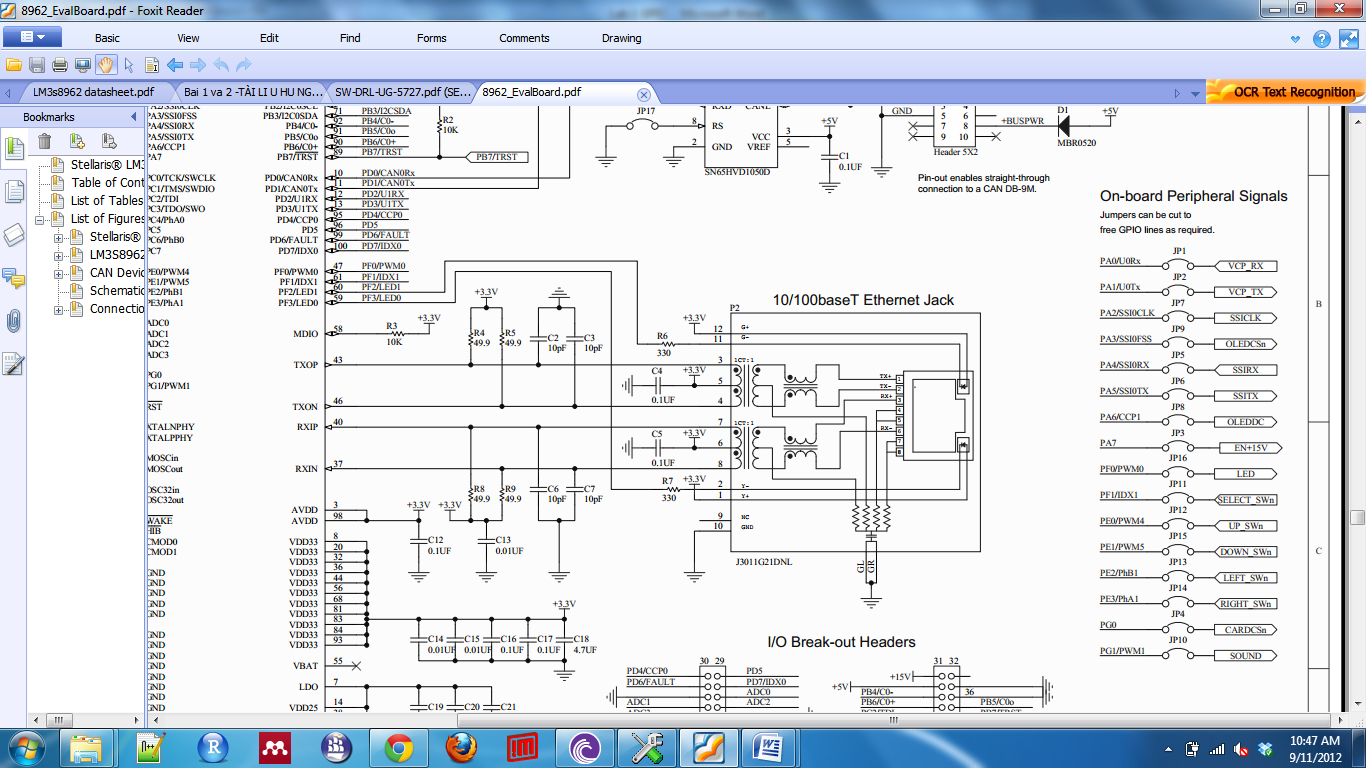
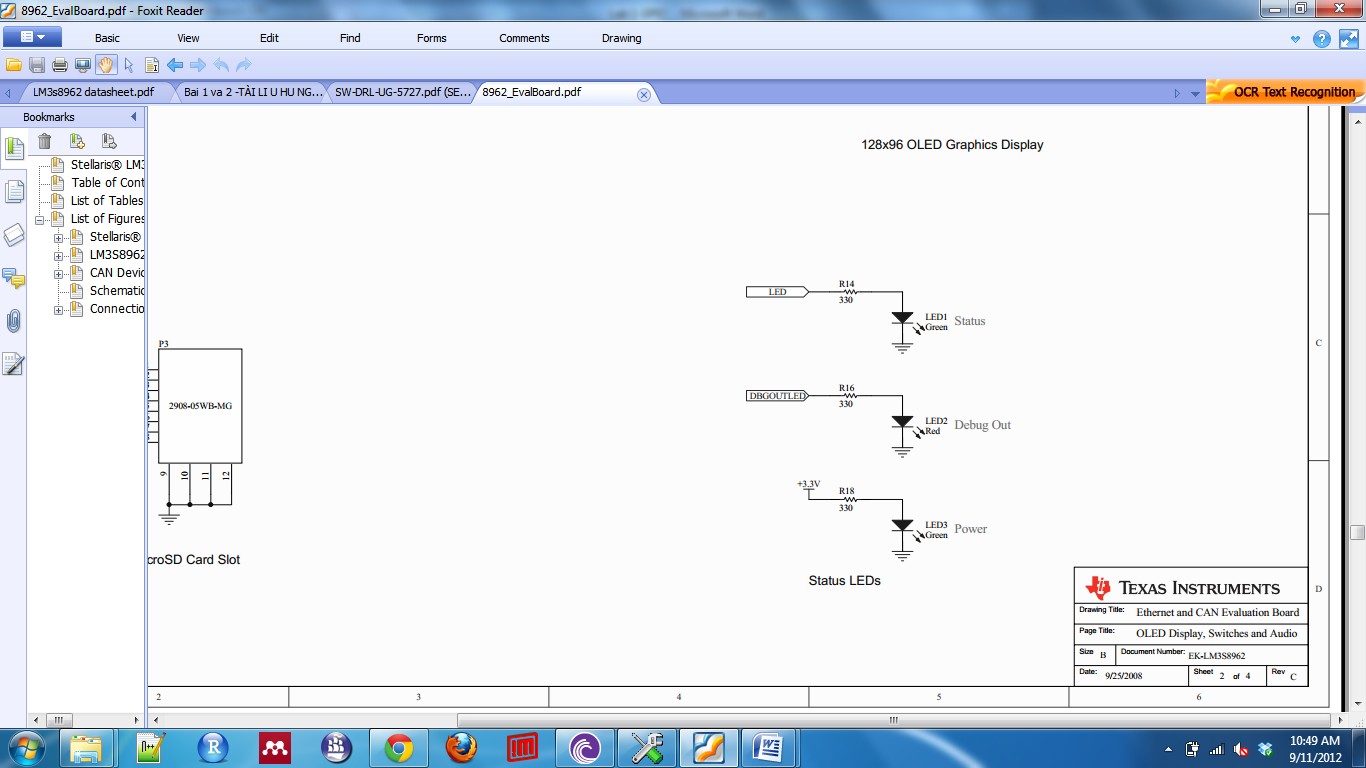
  

Figure: The electric schematic of buttons and sound

1. ***Create new project in IAR with name Lab1b and configure it as in Experiment 1.***
2. ***Copy and modify the code in Experiment 1 as following steps***
   1. Configure pin PF1 to Input mode
   2. Configure pin PF0 to Output mode
   3. The controlling rule is: if the select button is pressed, the LED will blink in frequency 1Hz. As illustrated in above figure, the pressed select button will generate the logic 0 and the Led will blink.
3. ***Question:***
   1. Define the design flow to write to a GPIO Port.
   2. Define the design flow to read data from a GPIO Port.
   3. Why do we have to set the pull up register for PF1 ?
   4. If we don’t know what is the current value of PORTF, write the C code to set pin 3 and pin 7 of PORTF without affecting to other pins in PORTF.
   5. If we don’t know what is the current value of PORTF, write the C code to clear pin 2 and pin 6 of PORTF without affecting to other pin of PORTF.

# **Experiment 3: Traffic Light - Connecting with real electric device**

Car

Pool lane

Left lane

Right lane

A freeway on-ramp has the traffic light control system to regular the flow of cars onto the freeway. There are three lanes, each with its own red/green (stop/go) light. The middle Pool lane has higher priority then 2 other lanes if the car presents in this lane. Right lane has higher priority than Left lane if cars presents in both two lanes. Yellow sensor has default value 0 if there is not car go to that lane and has value 1 when there are car got to the lane.

Alls variable using in the project are:

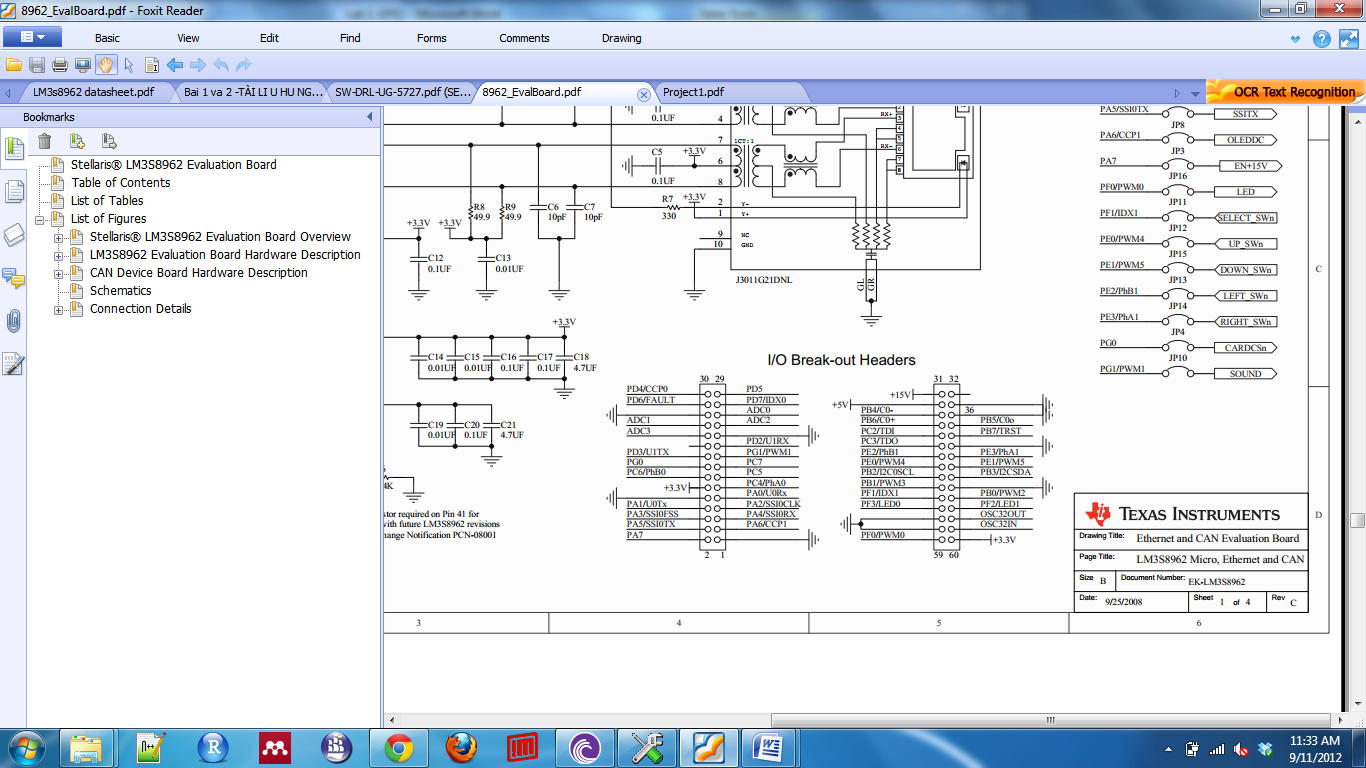
**Inputs:**

* CarPool: 1 if there is a car present in the car pool lane.
* CarLeft: 1 if there is a car present in the car left lane.
* CarRight: 1 if there is a car present in the car right lane.

**Outputs:**

* LightPoolR: Turn on the light Red of Pool Line
* LightPoolG: Turn on the light Green of Pool Line
* LightLeftR: Turn on the light Red of Left Line
* LightLeftG: Turn on the light Green of Left Line
* LightRightR: Turn on the light Red of Right Line
* LightRightG: Turn on the light Green of Right Line

**Setting up the project.**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Device** | **Pin using** | **Direction** | **Description** |
| CarPool | Up switch | ? | Input | Active in Low level. |
| CarLeft | Left switch | ? | Input | Active in Low level. |
| CarRight | Right switch | ? | Input | Active in Low level |
| LightPoolR | *External Led* | ? | Output | Active in Low level |
| LightPoolG | *External Led* | ? | Output | Active in Low level |
| LightLeftR | *External Led* | ? | Output | Active in Low level |
| LightLeftG | *External Led* | ? | Output | Active in Low level |
| LightRightR | *External Led* | ? | Output | Active in Low level |
| LightRightG | *External Led* | ? | Output | Active in Low level |

1. Create Project Lab1c named Traffic Control and configure as Lab1a and Lab1b
2. Configure all available pins with charateristic as above table.
3. Writing C code and verify program using View Register mode.
4. Implement the real circuit for the Project.