CMPE 443 PRINCIPLES OF EMBEDDED SYSTEMS DESIGN

LAB #007

“System Tick Timer”

“Analog to Digital Converter”

Motivation

Real-time and complex embedded systems require an RTOS (real-time OS) to handle all possible tasks in real time. ARM offers System Tick Timer for Cortex-M family. System Tick Timer is programmed to produce 10ms timer interrupts. In this way, the real-time application code becomes portable on all Cortex-M microcontrollers provided that the clock frequency is 100MHz.

Sensors provide analog data while actuators/actors may need analog signals to produce the desired actions. We will use Analog-to-digital Conversion (ADC) to detect light intensity via a Light Dependent Resistor (LDR) to adjust brightness of a LED with PWM which can also serve as a practical means of generating an analog value by exploiting the duty cycle of a digital signal.

In this experiment, you will learn

- configuring the System Tick Timer
- configuring the ADC port of a microcontroller
- reading data from an ADC pin

1) Problem Description

In this experiment, you produce an interrupt at every 10ms with System Tick Timer. Whenever System Tick Timer interrupt occurs you will start ADC module in order to light value of the LDR sensor. When ADC conversion finished, it will stop itself and PWM will change the brightness of the LED according to light value.

Note: When a question ends with ***(*)*** *notation, that means write on the code. When a question ends with* ***(?)*** *notation, that means write on the paper. When you see* ***(*?)****, the answer of this question should be written on the paper and code.*

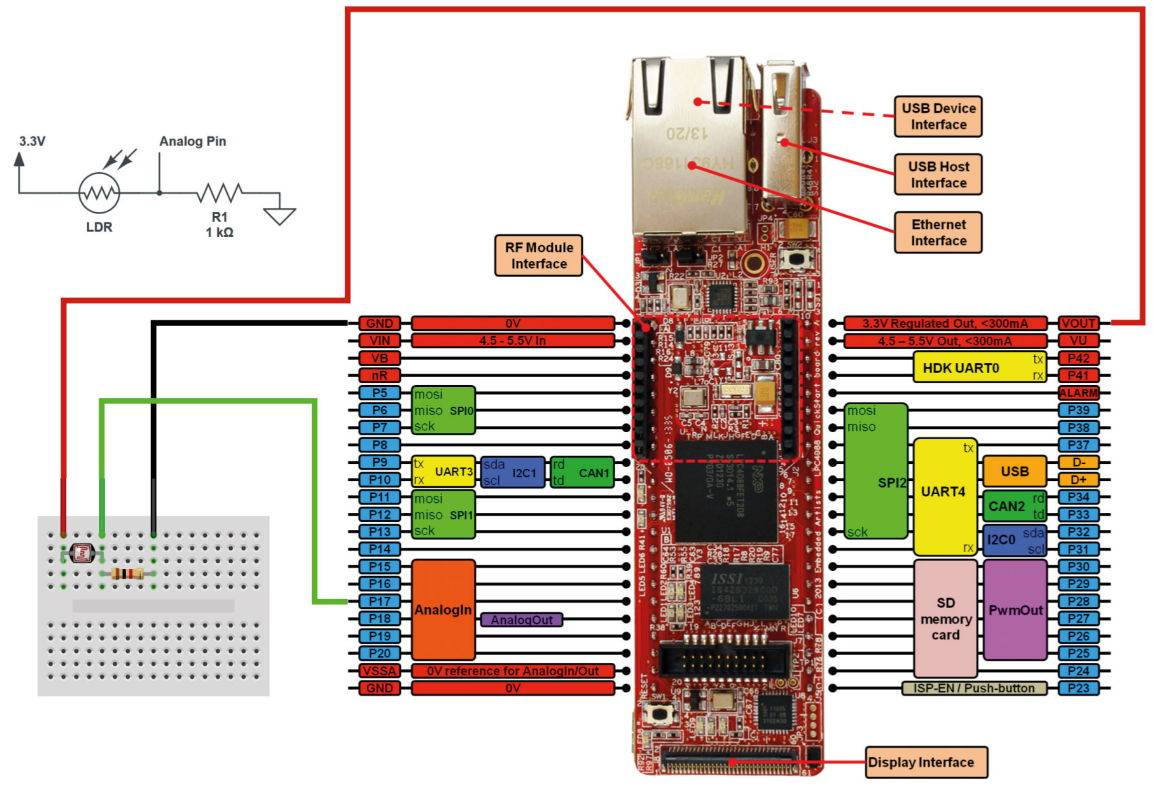
2) Initialize System Tick Timer *4 pts*

The System Tick Timer is an integral part of the Cortex-M4. It is intended to generate a fixed 10 millisecond interrupt. However, it can be used to generate interrupts at other frequencies by selecting the correct reload value. In this experiment, you will use System Tick Timer for generating a fixed 10 millisecond interrupt. In **SystemTickTimer.h** and **SystemTickTimer.c** files (CCLK = 120MHz):

- Write the correct address for System Tick Timer. **(*?)** ____________ *1 pt*
- Load a value to System Tick Timer for 10 ms **(*?)** ____________ *1 pt*

- Change the current value of System Tick Timer to zero **(*?)** ____________ *0,5 pts*
- Enable System Tick Timer and System Tick Timer Interrupt and Select CPU as its clock **(*?)** ____________ *1 pt*
- Enable SysTick_IRQn (Interrupt Request) **(*?)** ____________ *0,5 pts*

3) Connecting LDR Sensor to LPC4088 *1 pt*



Component Terminals	LPC4088 Pins
LDR	3.3V
1Kohm Resistor	Ground
LDR - 1Kohm Resistor (Middle)	P17 (P0_25)

4) LDR Sensor *3 pts*

LDR (Light Dependent Resistor) a component that has a variable resistance that changes with the light intensity that falls upon it. Resistance of any electronic component should not be measured on board, because it will never give the correct value. Hence, resistance of the LDR has to be found indirectly, by using multimeter and only measuring voltage on the serial resistor (not the LDR):

- Explain the method which makes you find the resistor value of the LDR by only measuring voltage value of the resistor serially connected to the LDR **(?)** ___________ *1 pt*
- Write the resistor value under the LAB lighting condition **(?)**
  ___________ *0,5 pts*
- Write the resistor value when you decrease the light intensity that falls upon it **(?)**
  ___________ *0,5 pts*
- Write the resistor value when you increase the light intensity that falls upon it **(?)**
  ___________ *0,5 pts*
- Draw the LDR characteristic (LDR Resistor versus Light Intensity) **(?)**
  ___________ *0,5 pts*

5) Initialize ADC *8 pts*

In order to initialize pins which are connected to circuit, you should find find the IOCON register addresses (**ADC.h**):

- Write the IOCON address of ADC Pin which is connected to the board. **(*)**
  ___________ *0,5 pts*

A/D Converters have a limited amount of resolution. (**ADC.h**):

- Write the max value of ADC (The return value when the max voltage value is read). **(?*)**
  ___________ *0,5 pts*

A/D Converter will work slowly in this experiment and A/D Converter Clock should be 1MHz. (PCLK = 60 MHz)

- Write the CLKDIV of ADC for 1 MHz. **(?*)** ___________ *0,5 pts*

Using a pin for A/D Converter, you should change functionality of the pin. However, this is not enough for ADC. You should also change MODE and ADMODE of the pin. (**ADC.c**)

- Change the function value of pin to ADC **(?*)** ___________ *0,5 pts*
- Change the mode value of pin to mode which should be selected if Analog mode is used. **(?*)**
  ___________ *0,5 pts*
- Change Analog/Digital mode of pin to Analog. **(?*)** ___________ *0,5 pts*

After pin initialization, you should initialize ADC:

- Turn on ADC. **(?*)** ___________ *0,5 pts*
- Select corresponding AD pins to be sampled and converted. Also, set the CLKDIV and make the A/D converter operational without Burst mode. **(?*)**
  ___________ *1,5 pts*

- Enable interrupt for corresponding pin. **(?*)** ___________ *0,5 pts*
- Enable ADC_IRQn (Interrupt Request). **(?*)** ___________ *0,5 pts*

In this experiment ADC should have a stop and start functionality. For this case:

- Write a code for starting A/D conversion. (In **ADC_Start** method) **(?*)** ___________ *0,5 pts*
- Write a code for stopping A/D conversion. (In **ADC_Stop** method) **(?*)** ___________ *0,5 pts*

Data Registers (DRs) contains the converted data. However, only specific bit range has the converted data. (**ADC_IRQHandler**)

- Write the converted data (only the converted data) to ADC_Last variable. **(?*)** ___________ *1 pt*

6) Changing Green LED Brightness by LDR *4 pts*

In this part, you will write codes into **update** method for:

- Under the LAB Lighting condition, Green LED should be turned off.
- When the light intensity is decreased, Green LED brightness will be increased.
- Change the brightness of the LED by using average value of last 10 ADC value. (10 A/D Conversion - 1 Brightness update, 10 A/D Conversion - 1 Brightness update ...)

Note: You can used **ADC_New_Data_Available** variable and **PWM_Write**, **ADC_GetLastValue** methods.