**Reaction Time Game Project**

Team Names:

Youssef Taha Saad Taha

Ziad Ahmed Mohamed Abdelfattah

Moamen Mohamed Ahmed Hassan

Supervised by:

Dr. Lamiaa El-Refaey

Eng. Mahmoud Nawar

**Table of Contents:**

1- Introduction

2- System Design  
 2.1. Hardware Components  
 2.2. Software Components

3- Software Implementation  
 3.1. Timer2A  
 3.2. GPIO\_PortF  
 3.3. Analog-to-Digital converter  
 3.4. LED  
 3.5. UART Communication  
 3.6. Nokia5110\_init

**1 - Introduction**

The Reaction Time Game is an interactive project designed to measure the user's reaction time. Built around the TM4C123GH6PM microcontroller, the system challenges users to press a button corresponding to an LED that lights up. The challenge is that the user needs to press the button during an interval of time and if the button isn’t pressed during this interval of time, then the user fails. The game is enhanced with levels of difficulty as the levels get higher and the user will have less time to press the button before the LEDs turn off. The user is also provided with real-time feedback about the number of attempts he has left on a Nokia 5110 LCD.

**2- System Design**

2.1 Hardware Components:

* TM4C123GH6PM Microcontroller
* LEDs (Red, Green)
* Push-button switches
* Potentiometer
* Resistors (330 Ω for LEDs, pull-up resistors for switches)
* Nokia 5110 LCD
* UART communication interface

2.2 Software Components:

* Embedded C programming
* Keil uVision IDE for code development
* TivaWare library for peripheral control

**3- Software Implementation**

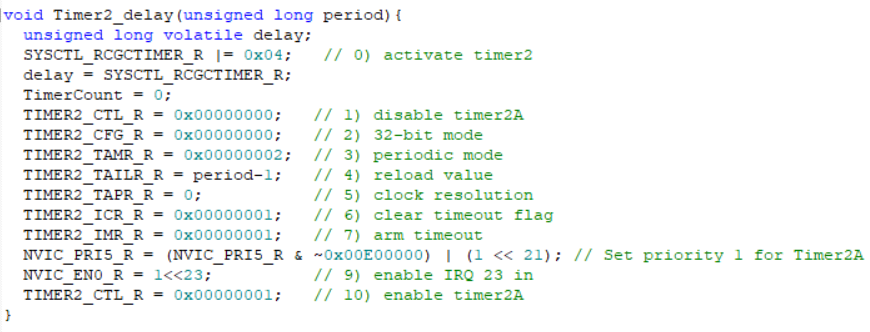
As shown previously, the project uses Timer2A, UART, Nokia5110, GPIOs, LED, and ADC.

**3.1. Timer2A:**

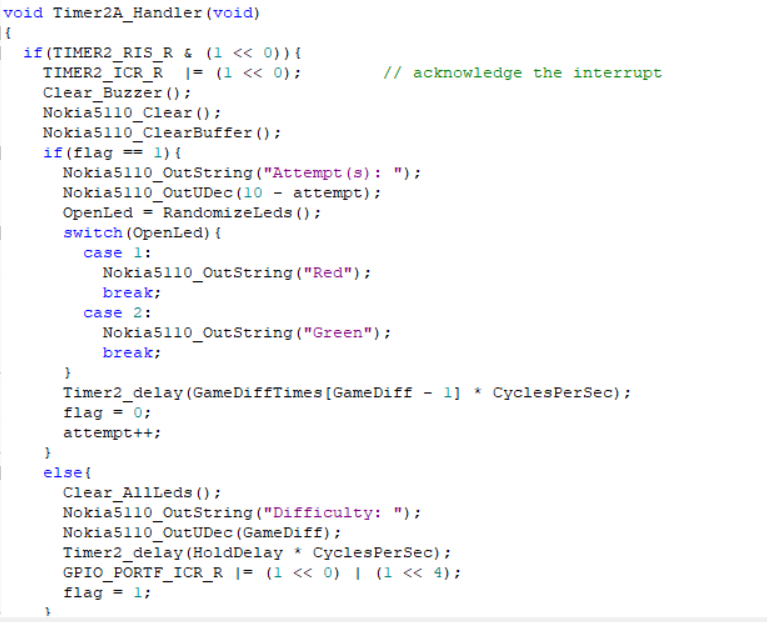
This timer function is used in coordination with its handler function to randomize the lighting up of the LEDs. The handler function is also used to show the user the number of attempts he has left before he runs out.

Timer2\_delay function:

This function is used to initialize Timer2A with a value written as its parameter. After the value inside the parameter is zero the interrupt handler will be called. The timer function can only have two possible values which are: the time value which is set by the user choosing a level of difficulty, and the time value which is set in between every attempt.



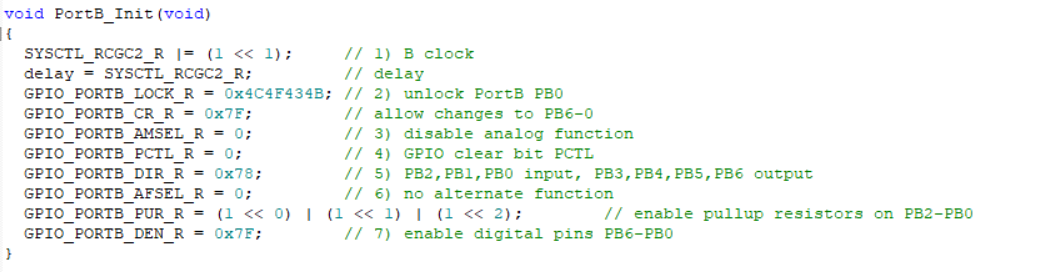
Timer2A\_Handler function:

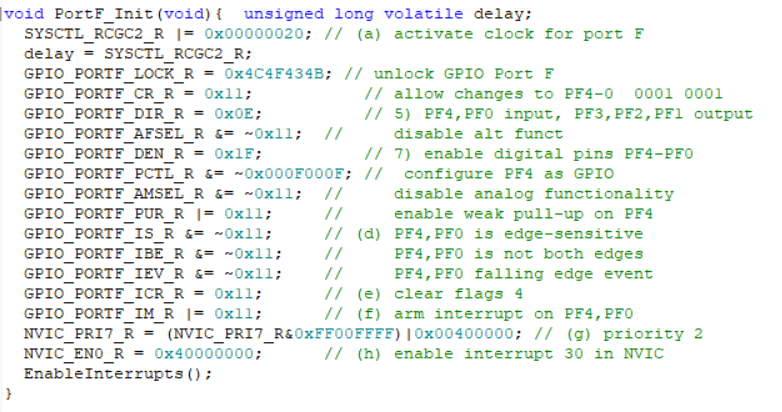


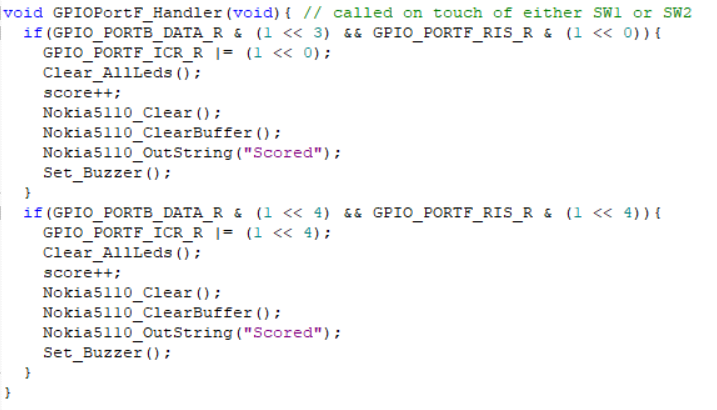
First, the handler function clears the screen of the Nokia5110. Then, it shows the number of attempts the user has left. Every user is allowed to attempt 10 attempts. The handler checks if the user is in delay time between the attempts; if so, it calls the RandomizeLeds function which is used to light up an LED in front of the user at random. Other than the user being in the time between the attempts, the handler will give the user a Be Ready message which means that an LED is going to light up.

**3.2. GPIO:**

PortF is used for handling Switches’ interrupts, PortB is for the output LEDs and Buzzer, and one switch on PortB to reset the game another time.





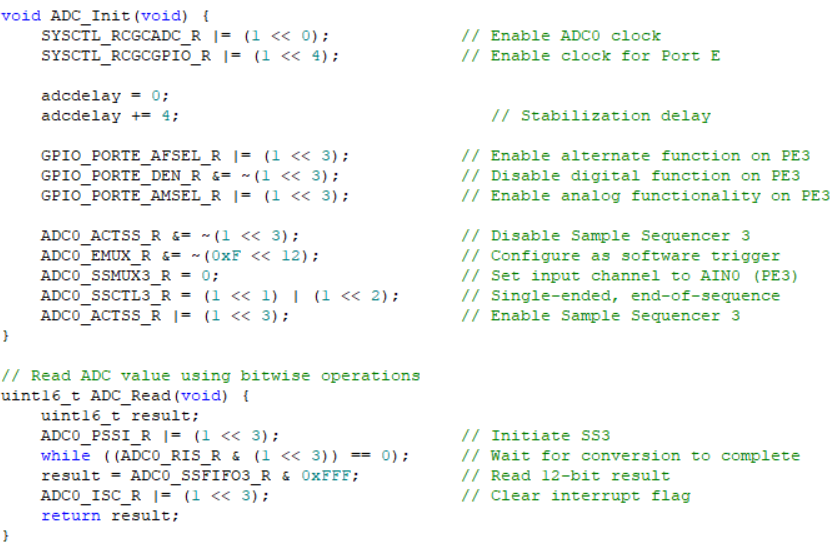


Also, the handler function checks for the pressed switch and its LED if the user pressed right or wrong there are two variables `attempt`, and `score`. The `attempt` variable is used to count the wrong and right attempts to make the game with only ten attempts per game and the `score` variable is used to count the right attempts only.

**3.3. Analog-to-Digital converter:**

The analog-to-digital converter is used for the potentiometer. The ADC uses the voltage on the potentiometer to allow the user to choose the desired difficulty.

**ADC1\_initialize:**



ADC Initialization (ADC\_Init)

* Enables the clock for ADC0 and Port E.
* Configures PE3 as an analog input pin (disables digital functions and enables analog mode).
* Sets up Sample Sequencer 3 to use software triggering and single-ended input (AIN0).
* Configures the ADC to generate interrupts upon completing a conversion and marks the end-of-sequence.

ADC Read (ADC\_Read)

* Starts a conversion on Sample Sequencer 3.
* Waits for the conversion to complete by polling the ADC interrupt status.
* Reads the 12-bit result from the FIFO and clears the interrupt flag to prepare for the next conversion.

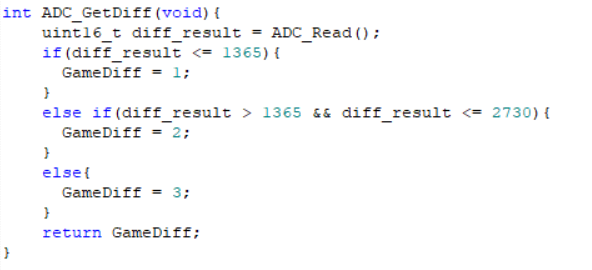
This setup lets you read analog values from a potentiometer or sensor connected to PE3.

**ADC\_GetDiff:**

This function, ADC\_GetDiff, is a method that chooses the difficulty of the game. The entire ADC range (0-4095) is divided into three equal sections:

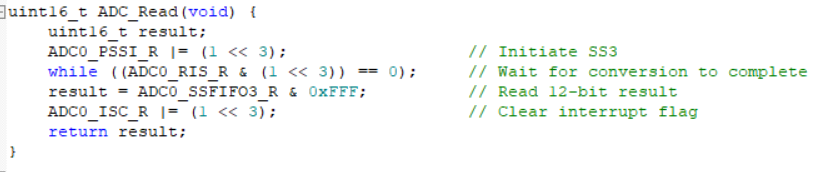
* Level 1: 0 - 1365
* Level 2: 1366 - 2730
* Level 3: 2731 - 4095

The corresponding difficulty levels of the game (GameDiff) are assigned based on the ADC value (diff\_result).



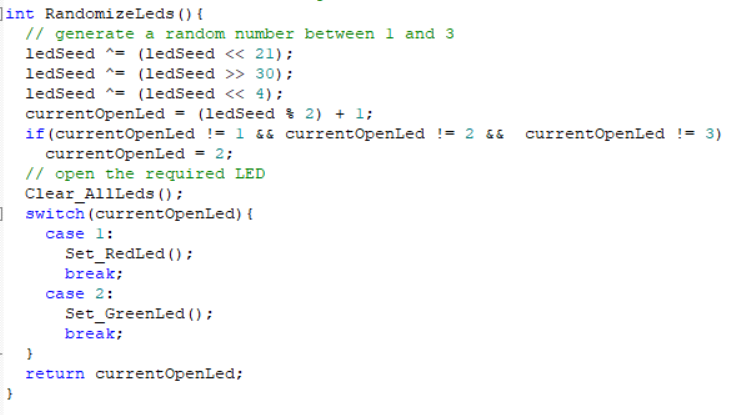
**ADC\_Read:**

The `**ADC\_Read**` function reads a 12-bit analog-to-digital conversion result from an ADC module. It begins by initiating a conversion on Sample Sequencer 3 (SS3) by setting the corresponding bit in the `**ADC0\_PSSI\_R`** register. The function then waits for the conversion to complete by monitoring the `**ADC0\_RIS\_R`** register until the appropriate bit is set, indicating readiness. Once complete, it retrieves the 12-bit result from the **`ADC0\_SSFIFO3\_R`** register, which stores the output of the conversion, and masks it with **0xFFF** to ensure only the 12 least significant bits are considered. Finally, it clears the interrupt flag for SS3 by writing to the **`ADC0\_ISC\_R`** register and returns the converted digital value as a 16-bit unsigned integer **(uint16\_t)**.

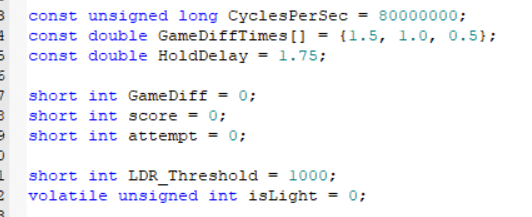


**LED:**

This file is used to initialize the LEDs in port B, Set the Buzzer, and Clear the LEDs. It also has a function called “RandomizeLeds” This function is used to set a random LED.



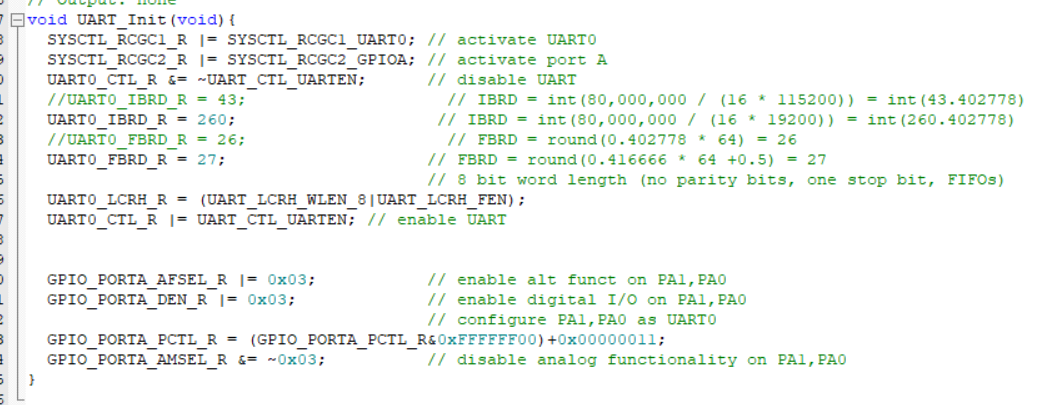
**GlobalConfiq:**

This file has all the variables that are being used in all files such as the array of the game difficulty, attempt, and score variables.

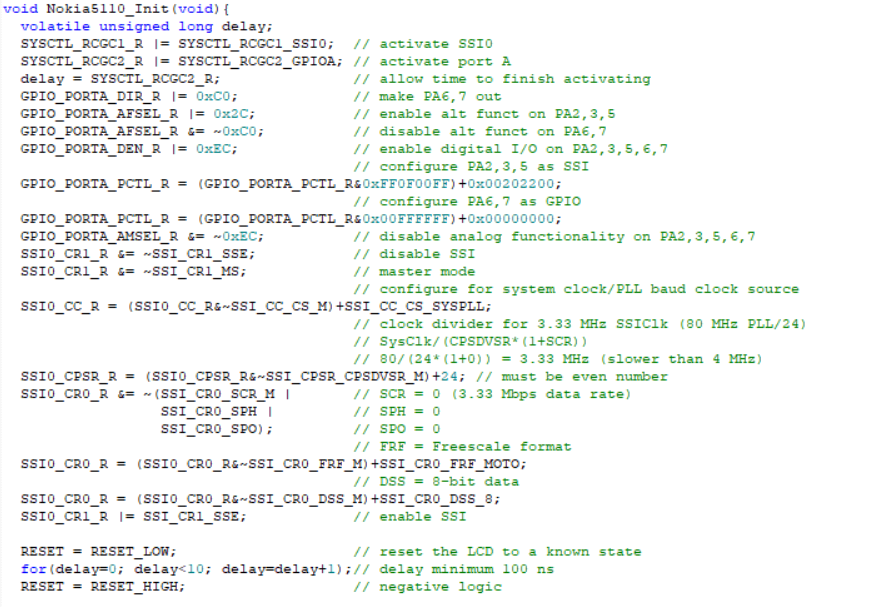
**UART:**

This file effectively implements UART communication and provides utility functions for handling data in various formats. It is well-suited for debugging and data display in embedded systems. Enhancements in documentation, modularity, and testing can further improve its robustness and maintainability.

The `UART\_Init()` function initializes the UART0 module to communicate at a baud rate of 19200. It activates UART0 and Port A, configures the baud rate parameters (IBRD and FBRD), and sets the data format to 8-bit word length, no parity, one stop bit, and enables FIFOs. It also configures pins PA0 (U0Rx) and PA1 (U0Tx) for UART functionality by enabling their alternate functions and disabling their analog mode. Finally, it enables the UART by setting the UARTEN bit in the control register.



**Nokia5110\_init:**



**Steps in Initialization:**

1. **Activate SSI0 and Port A:**
   * Enables the clock for SSI0 and GPIO Port A.
2. **Configure GPIO Pins:**
   * PA2, PA3, and PA5 are set to alternate functions for SSI communication.
   * PA6 and PA7 are configured as GPIO outputs for reset (RESET) and chip select (CS).
3. **Disable Analog Functions:**
   * Disables analog functionality for PA2, PA3, PA5, PA6, and PA7.
4. **Configure SSI Settings:**
   * Sets SSI in master mode, Freescale SPI format, and an 8-bit data size.
   * Configures SSI clock to operate at 3.33 MHz.
5. **Reset the LCD:**
   * Sends a reset pulse to ensure the LCD is in a known state.
6. **Set Extended Instruction Mode:**
   * Configures the LCD to use the extended instruction set.
   * Adjusts contrast, temperature coefficient, and bias settings.
7. **Switch to Basic Instruction Mode:**
   * Switches back to the basic instruction set.
   * Configures the display for normal mode.