

|  |
| --- |
| Name: Abhishek N |
| College Reg No: 4VV19EC002 |
| Batch No: PGDEA101 |
| Cranes Reg. No: R2025040415710480 |
| Project Title: Vehicle Dashboard Monitoring System |
| S/w Used: Keil UVison4, Flash Magic |
| Peripherals Used: ADC, SPI, PWM, GPIO, LCD |
|  |
|  |
| Trainer Name: Prof Shaik Imam |

**Abstract**

This project presents the design and implementation of a simulated car dashboard system using the LPC1768 microcontroller, which is based on the ARM Cortex-M3 architecture. The system incorporates two Analog-to-Digital Converter (ADC) channels for real-time sensing and control. The first ADC is connected to a potentiometer that functions as an accelerator pedal. The analog voltage from the potentiometer is converted into a digital value, which is then mapped to a corresponding PWM (Pulse Width Modulation) signal to control the motor speed. This speed value is continuously displayed on an LCD module for user feedback.

To introduce a basic safety alert mechanism, a buzzer system is integrated based on vehicle speed. When the speed surpasses 80 units, a short beep lasting 0.1 seconds is generated. If the speed exceeds 120 units, a long beep of 1 second is triggered. These timed alerts are implemented using hardware timers to ensure precise duration and non-blocking operation.

The second ADC channel is connected to an LM35 temperature sensor to monitor the engine temperature. The analog temperature signal is digitized and processed to calculate the temperature in degrees Celsius. This temperature value is displayed on the LCD for general monitoring and also sent to a 7-segment display via SPI (Serial Peripheral Interface) for visual feedback.

This project demonstrates effective integration of multiple embedded system components ADC, PWM, timers, SPI, LCD, and buzzer—into a cohesive application. It highlights the microcontroller's capability to interface with analog and digital peripherals to simulate a functional and interactive automotive dashboard. The design can serve as a prototype for understanding real-time signal processing and peripheral management in embedded systems.

**Introduction to the Project**

The rapid advancement in embedded systems has significantly influenced the automotive industry, enabling intelligent and efficient vehicle control systems. One of the key components of any automobile is its dashboard, which acts as a user interface between the driver and the vehicle’s internal systems. It provides real-time data related to speed, temperature, fuel level, and warnings, allowing the driver to make informed decisions. This project aims to replicate a simplified version of such a dashboard system using the LPC1768 microcontroller, based on the ARM Cortex-M3 architecture. The primary objective is to monitor speed and engine temperature, provide visual feedback through displays, and generate audio alerts based on specific conditions.

To simulate acceleration, a potentiometer is used to act as an accelerator pedal. The analog output voltage from the potentiometer is fed into the microcontroller's ADC (Analog-to-Digital Converter), where it is digitized. The resulting digital value is directly mapped to a PWM (Pulse Width Modulation) signal that controls the simulated motor speed. The use of PWM allows fine-grained control of speed, which is then displayed in real-time on an LCD module. This setup closely mimics how modern vehicles respond to accelerator inputs and display speed data on digital dashboards.

In addition to speed display, the system incorporates a safety alert mechanism using a buzzer. If the vehicle's speed surpasses 80 units, a short beep of 0.1 seconds is triggered as a cautionary alert. If the speed exceeds 120 units, a long beep of 1 second is generated to indicate a more serious threshold breach. These audio warnings are precisely timed using hardware timers within the microcontroller, ensuring accuracy without blocking the main execution flow. This functionality emulates real-world dashboard warnings which alert drivers to excessive speed or system faults.

Furthermore, the system includes a temperature monitoring feature using an LM35 analog temperature sensor. The sensor outputs a voltage proportional to the ambient temperature, which is read via a second ADC channel on the microcontroller. The temperature is calculated in degrees Celsius and is continuously updated on the LCD. For enhanced visibility, this temperature data is also transmitted to a 7-segment display using SPI (Serial Peripheral Interface) communication. This feature adds another layer of realism and complexity, as SPI is commonly used in embedded systems for high-speed peripheral communication.

Overall, this project demonstrates the practical application of core embedded system concepts such as ADC reading, PWM generation, SPI communication, timer usage, and peripheral interfacing. It showcases how various modules can be integrated into a cohesive and responsive system that mimics the behavior of an actual automotive dashboard. The design not only reinforces theoretical knowledge but also serves as a foundation for more complex automotive and real-time embedded applications.

**Algorithm of the code**

**Algorithm : Main Function**

**Step 1: Initialization**

1. Set GPIO pins for output:
   * Configure **P1.19 to P1.27** as output for **LEDs and Buzzer** using FIODIR.
   * Clear these GPIOs initially using FIOCLR.
2. Initialize all required modules:
   * Call adc\_init() to initialize ADC.
   * Call spi\_init() to initialize SPI (used likely for 7-segment display).
   * Call lcd\_init() to initialize the LCD.
   * Call pwm\_init() to initialize PWM for speed control simulation.
   * Call timer0\_init() for software delay routines.

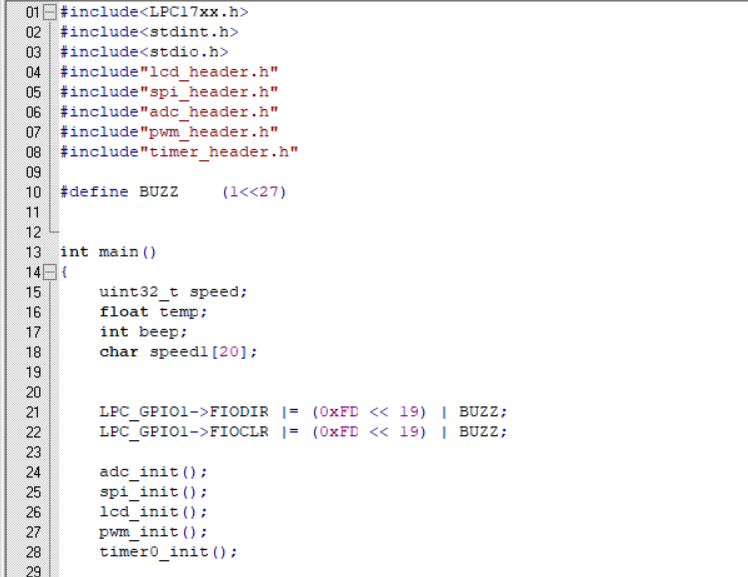
**Step 2: Main Loop**

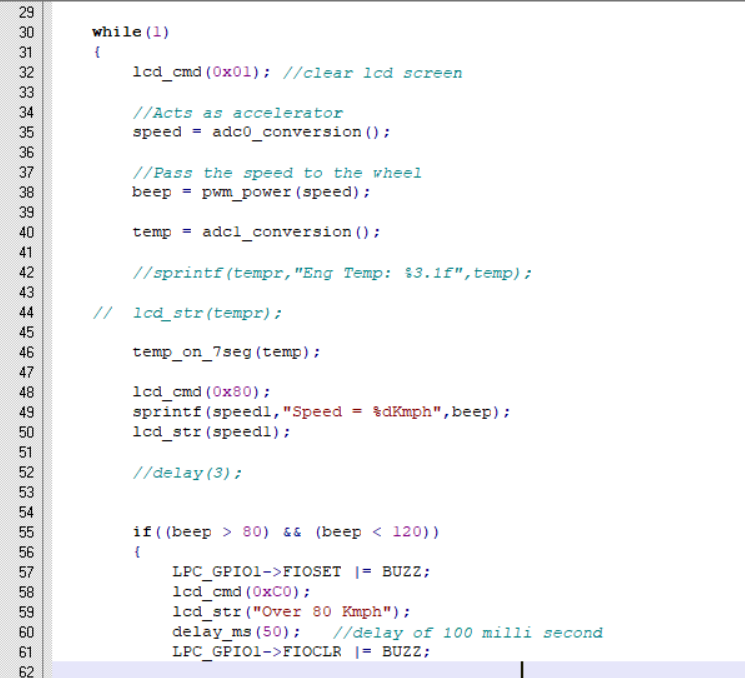
**Repeat Forever:**

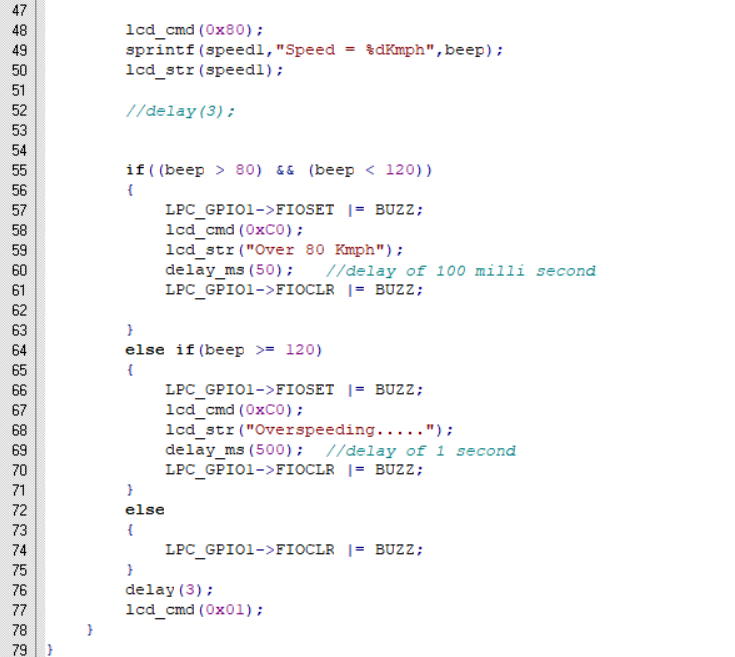
1. **Clear LCD screen**:
   * lcd\_cmd(0x01)
2. **Read Accelerator Speed:**
   * Call adc0\_conversion() → speed
   * This reads analog value simulating accelerator pedal.
3. **Control Motor Speed via PWM:**
   * Call pwm\_power(speed) → beep
   * This function converts ADC value to PWM duty cycle and returns equivalent speed (km/h).
4. **Read Engine Temperature:**
   * Call adc1\_conversion() → temp
5. **Display Engine Temperature on 7-segment:**
   * Call temp\_on\_7seg(temp) to send temperature to SPI-driven 7-segment.
6. **Display Speed on LCD:**
   * Set LCD cursor to first line: lcd\_cmd(0x80)
   * Format string "Speed = %dKmph" and display with lcd\_str(speed1)
7. **Speed Alert Conditions:**
   * If beep > 80 && beep < 120:
     + Turn ON buzzer.
     + Display "Over 80 Kmph" on second LCD line.
     + Delay for 50 ms.
     + Turn OFF buzzer.
   * Else if beep >= 120:
     + Turn ON buzzer.
     + Display "Overspeeding....." on second LCD line.
     + Delay for 500 ms.
     + Turn OFF buzzer.
   * Else:
     + Ensure buzzer is OFF.
8. **Delay and Clear LCD:**
   * Delay for 3 seconds (delay(3)).
   * Clear LCD (lcd\_cmd(0x01)) for the next refresh.

**Codes with screen shot**

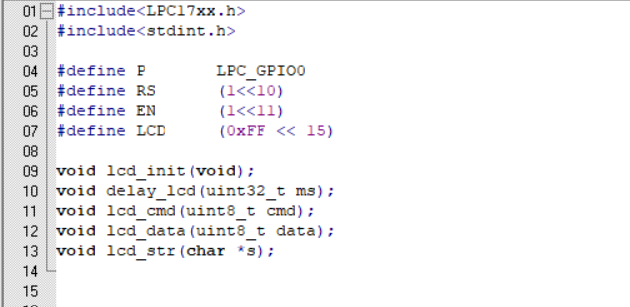
**Main Function**

****

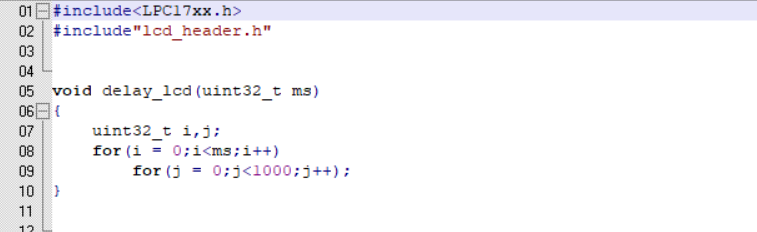
****

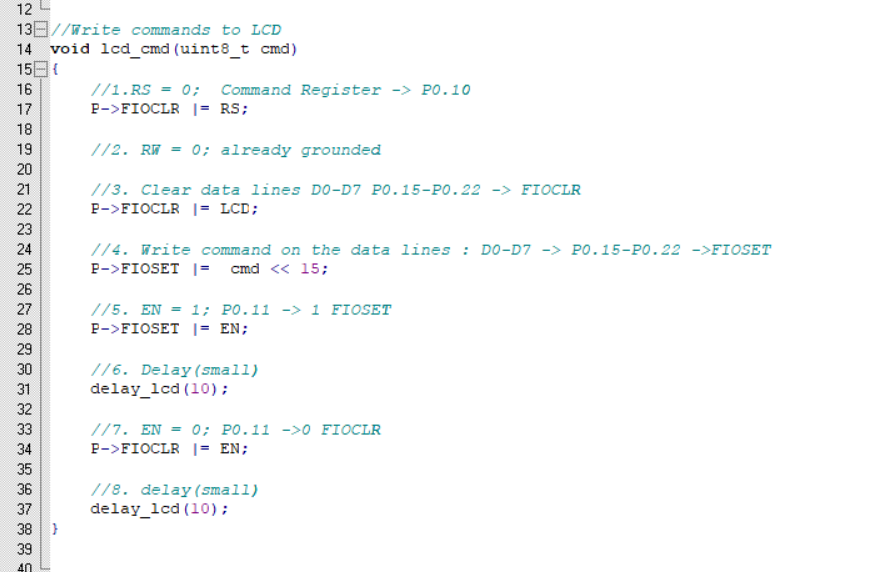
****

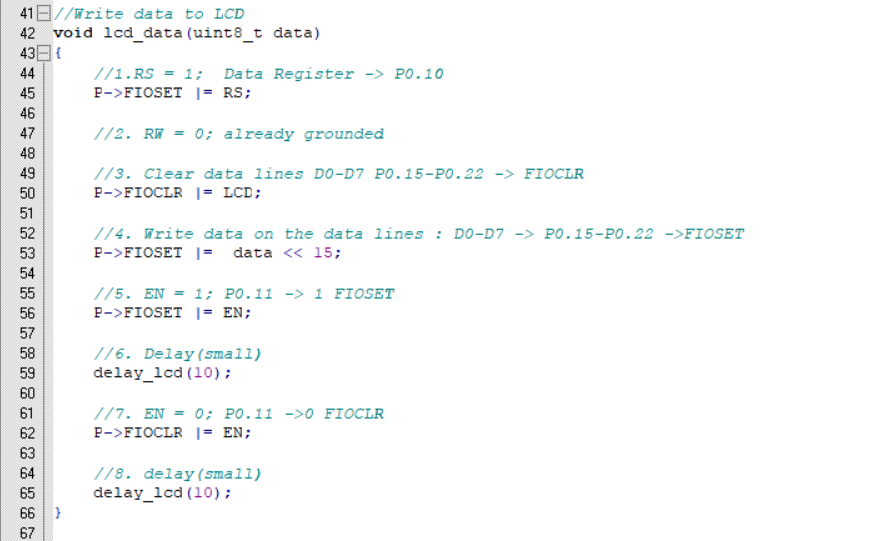
**LCD Header File**

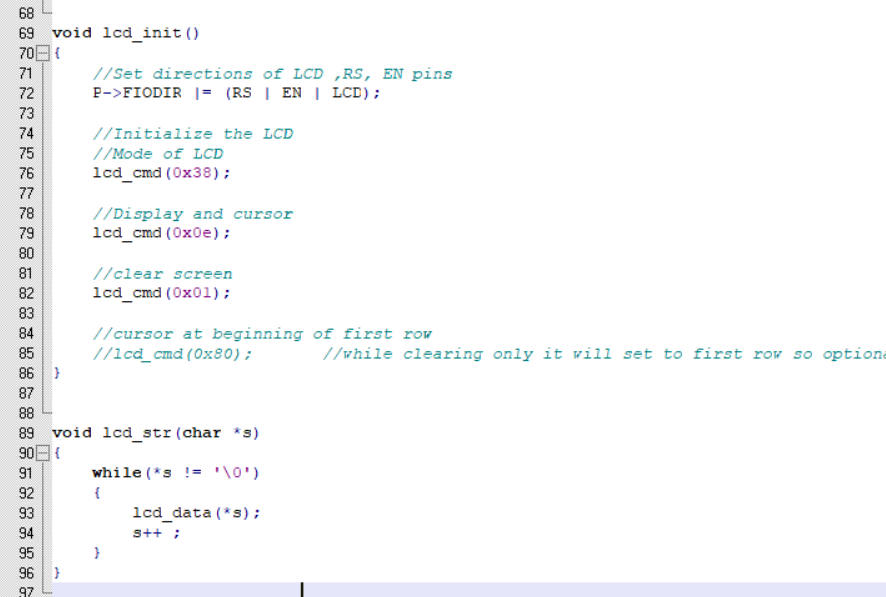
****

**LCD Function File**

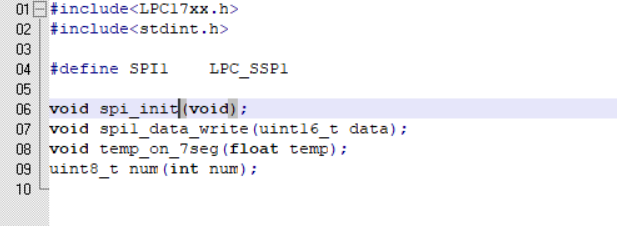
****

****

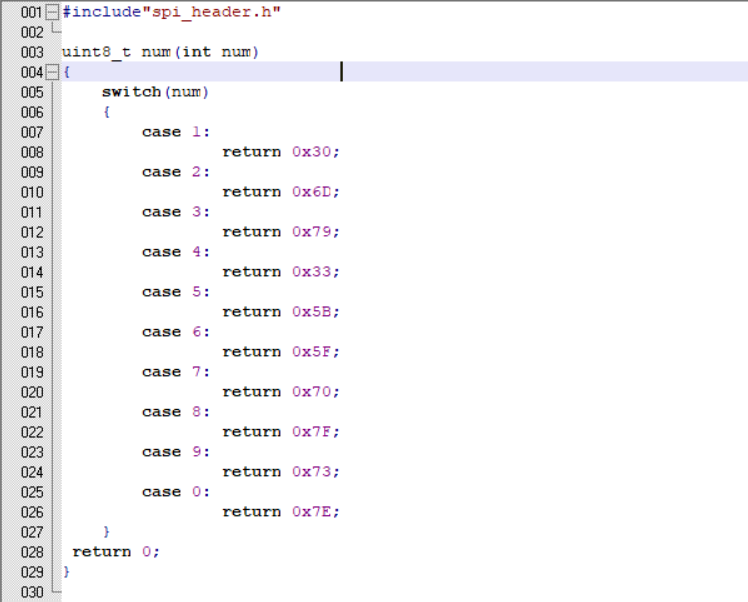
****

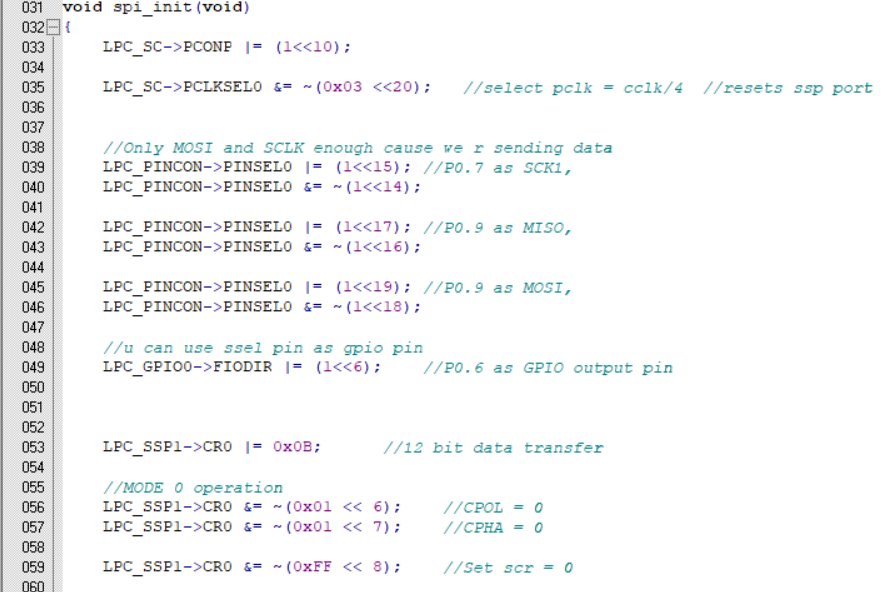
****

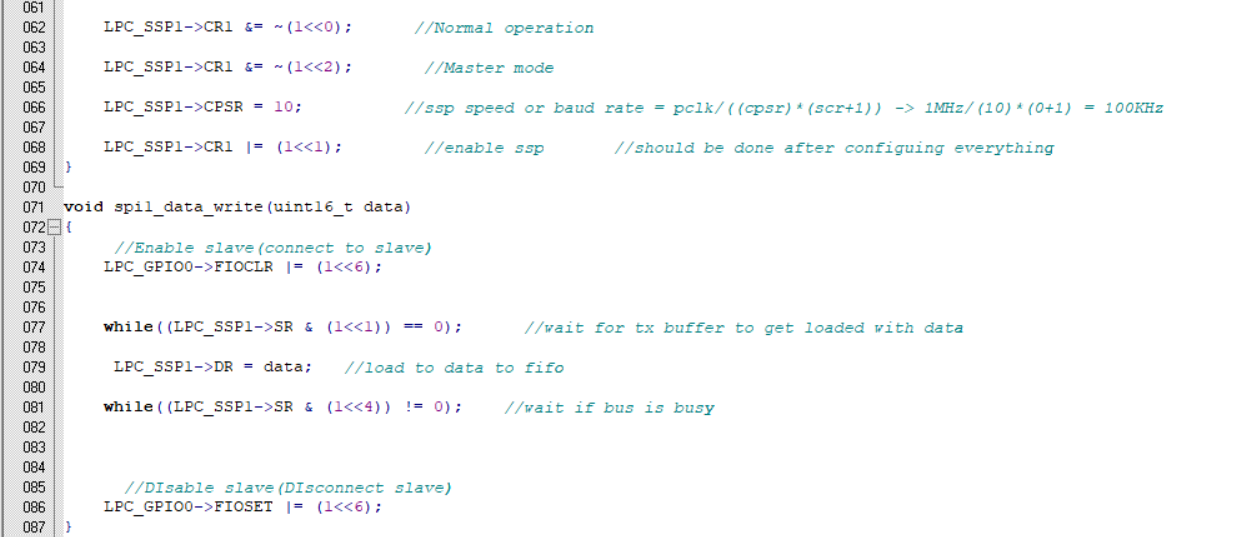
**SPI Header File**

****

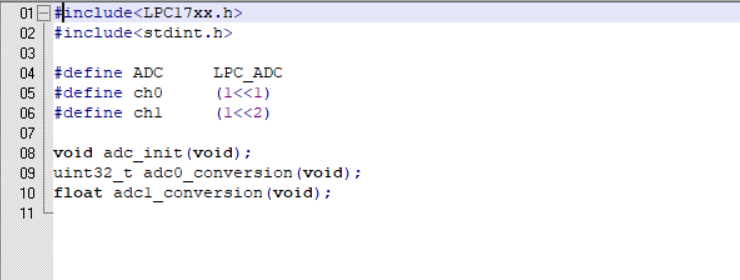
**SPI Function File**

****

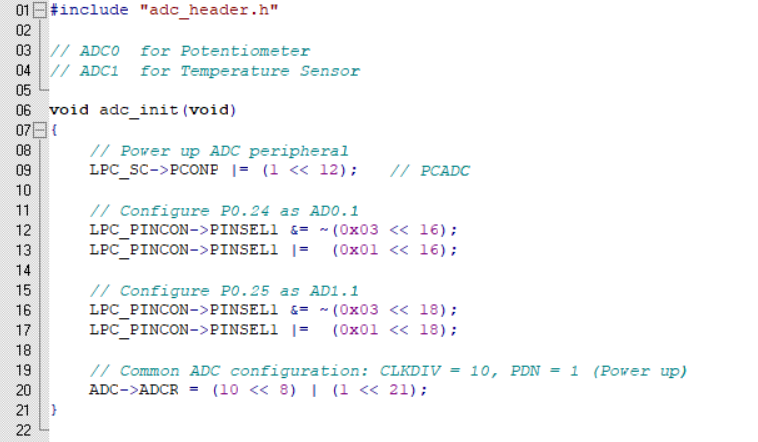
****

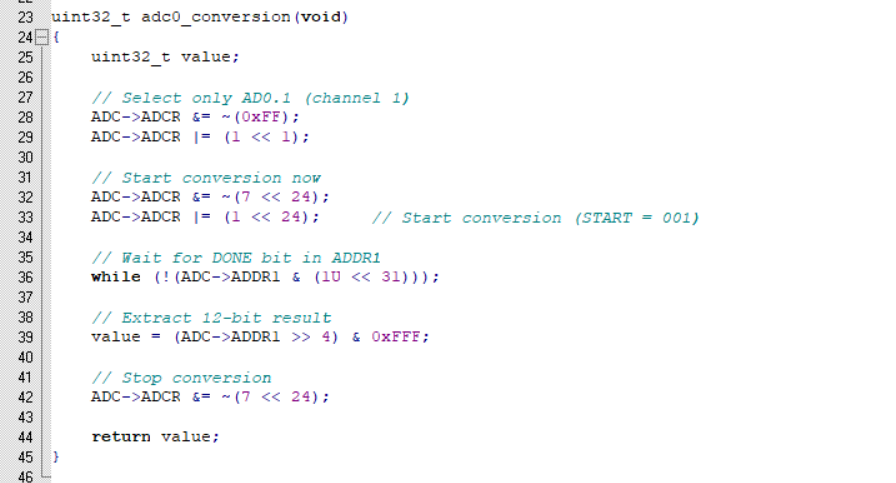
****

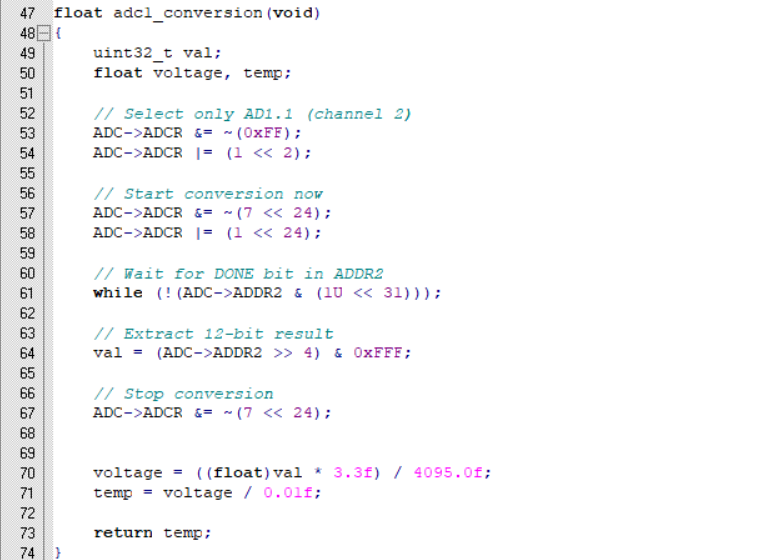
**ADC Header File**

****

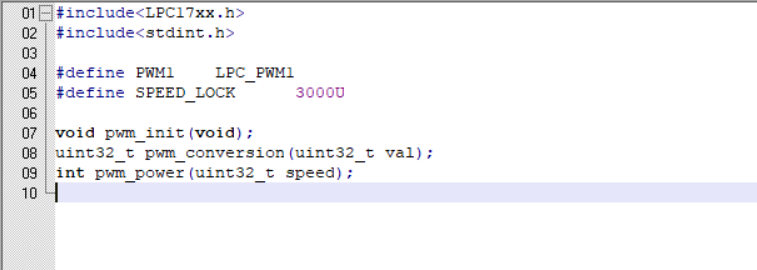
**ADC Function File**

****

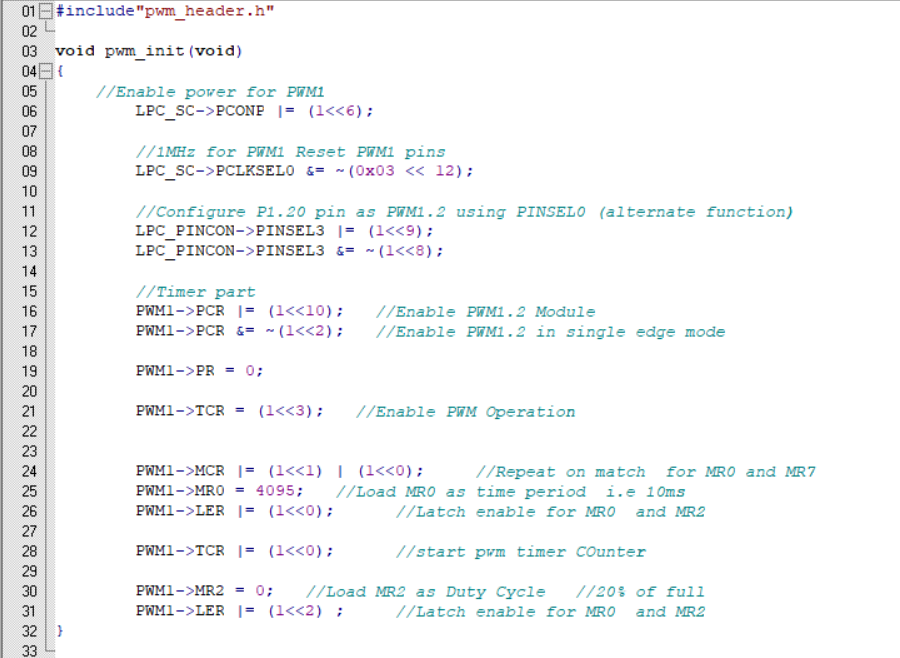
****

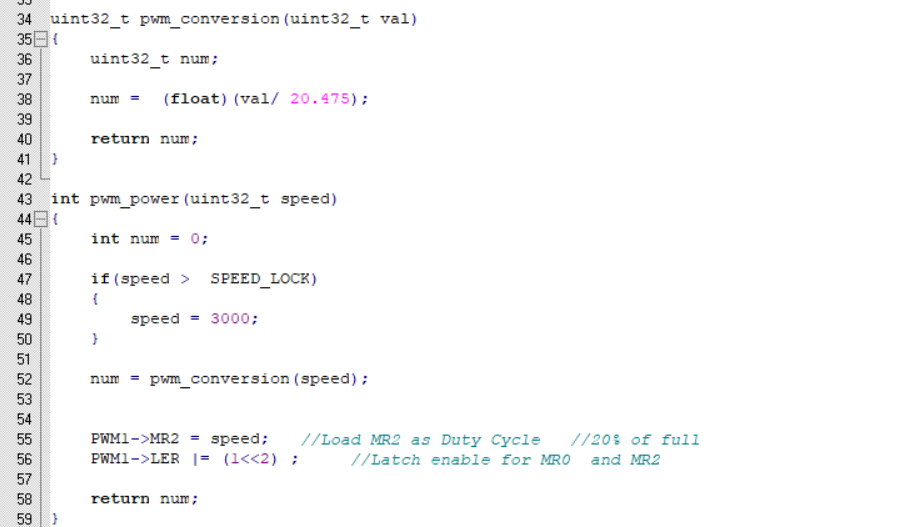
****

**PWM Header File**

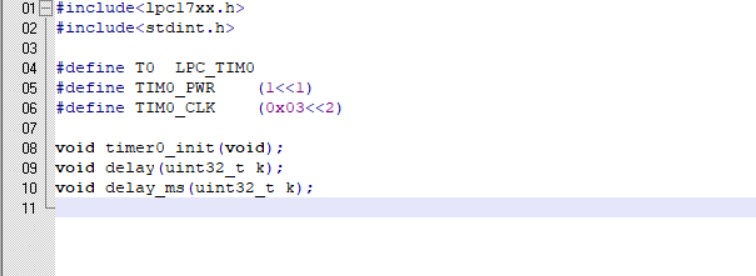
****

**PWM Function File**

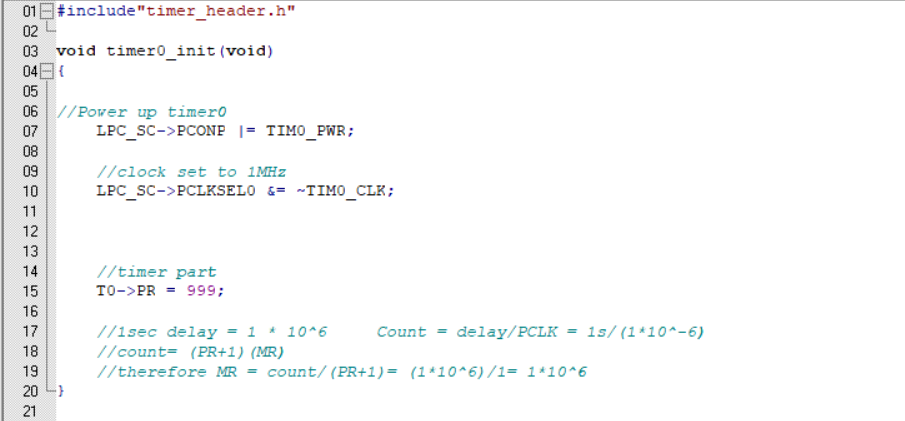
****

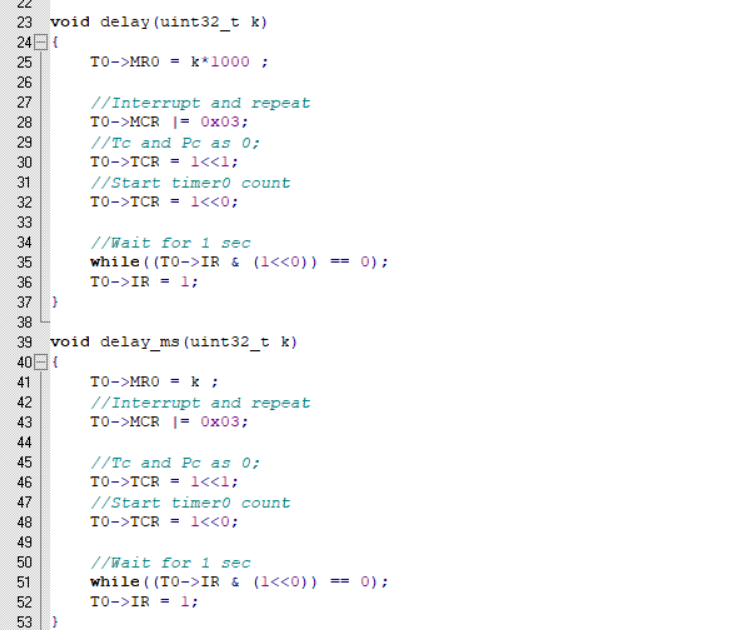
****

**TIMER Header File**

****

**TIMER Function File**

****

****

**Conclusion**

The car dashboard simulation project using the LPC1768 microcontroller successfully demonstrates the integration of various embedded system components to emulate real-time automotive monitoring and control. By utilizing ADC channels, the system effectively reads analog signals from a potentiometer and an LM35 temperature sensor, converting them into meaningful speed and temperature values.

The PWM module enables dynamic speed control based on user input, simulating an accelerator pedal mechanism. Real-time data visualization is achieved through an LCD and a 7-segment display, with the latter driven via SPI communication. Furthermore, the buzzer alert system enhances safety feedback by providing audio warnings based on predefined speed thresholds, with precise timing controlled by hardware timers.

This project not only showcases the practical application of ADC, PWM, SPI, LCD, timers, and buzzer integration but also provides a solid foundation for building more advanced automotive systems. It highlights the importance of real-time data processing and user feedback in embedded design. The system is modular, scalable, and serves as a strong prototype for future developments in smart automotive dashboards.

Overall, the project meets its objectives and serves as a valuable learning experience in embedded systems programming, peripheral interfacing, and system-level integration.

**Future Enhancements**

* **Integration with Real DC Motor or Servo Motor :**Instead of simulating motor speed, the PWM output can be connected to an actual DC or servo motor to control and visualize physical rotation based on accelerator input.
* **Bluetooth or Wi-Fi Connectivity :**Wireless modules such as HC-05 (Bluetooth) or ESP8266 (Wi-Fi) can be integrated to transmit speed and temperature data to a mobile app or web dashboard in real-time.
* **Graphical Display Interface (TFT/OLED) :**Replacing the 16x2 LCD with a graphical display would allow richer visual representation of parameters like speedometers, temperature dials, and warning icons.
* **Data Logging via SD Card :**Adding an SD card module would enable storing of speed and temperature data for future analysis, useful for diagnostics or performance review.
* **Battery Monitoring System :**Integration of a voltage sensor to monitor battery health and display it alongside speed and temperature.
* **Overheating Control System :**Automatically activate a cooling fan (via relay) when the engine temperature crosses a specific threshold to simulate real engine management.
* **CAN Protocol Integration :**Implementing CAN (Controller Area Network) communication to connect multiple microcontrollers, simulating a real car's distributed dashboard and sensor network.
* **Fault Detection and Alerts :**Include warning lights or messages on the display for conditions like overvoltage, sensor failure, or disconnected peripherals.
* **Voice Alert System**Use an audio playback module (e.g., ISD1820) to provide voice-based warnings instead of simple beeps for enhanced driver feedback.

**References**

* **NXP Semiconductors –** *LPC1768 Microcontroller Datasheet*
* **Texas Instruments –** *LM35 Temperature Sensor Datasheet*
* **Keil µVision IDE –** *Embedded C Development Environment for ARM Cortex-M*
* **Embedded Systems Programming Tutorials –** *PWM, ADC, Timer, and SPI Interfacing*
* **SPI Communication –** *Serial Peripheral Interface Protocol Basics*
* **LCD Interfacing with Microcontroller –** *16x2 LCD Programming and Control*
* **Timer Programming in LPC1768 –** *ARM Cortex-M3 Timer and Delay Functions*
* **PWM Motor Control Concepts –** *Pulse Width Modulation Techniques for Motor Control*