COAL (EL-2003)
Spring-2025

Final Project Report

“Real-Time Clock with Alarm System using MASM”

Group Members:

Hafsa Rashid (23K-0064)

Syeda Sara Ali (23K-0070)

Instructor:

Sir Ubaidullah

1. Introduction

This project presents the implementation of a **Real-Time Clock with an Integrated Alarm System** using MASM (Microsoft Macro Assembler). Designed as a system-level application, it demonstrates the capabilities of low-level programming by utilizing the Irvine32 library alongside Windows API functionalities. The project continuously monitors and displays the current system time in the HH:MM:SS format, allows users to set an alarm, and includes user-friendly features such as alarm snoozing and stopping. The use of color-coded console output and sound integration enriches the overall interactive experience.

2. Objectives

The primary objectives of this project are:

- To display **the real-time system clock** on the console in a readable format.
- To **enable users to input and validate an alarm time** interactively.
- To **trigger an alarm** when the set time matches the current system time.
- To **provide options** for either snoozing (by 2 minutes) or stopping the alarm.
- To **enhance user interaction** using colored messages on the console.
- To play sound notifications upon triggering or snoozing the alarm.

3. Tools and Technologies Used

The following tools and libraries were employed in the development of this project:

- MASM (Microsoft Macro Assembler)
- Irvine32 Library
- Windows API Functions
- winmm.lib for PlaySound functionality
- Assembly macros from macros.inc
- Console and system time handling

4. Working Mechanism

Upon execution, the program prompts the user to enter a valid alarm time (hours and minutes), ensuring that inputs fall within acceptable bounds (00–23 hours and 00–59 minutes). Once validated, the program enters a continuous monitoring loop:

- System Time Fetching: The program retrieves the system time in real time and formats it to HH:MM:SS.
- Time Display: The time is updated every second on the console using formatted and colored output.
- Alarm Comparison: It checks if the current time matches the user-defined alarm.
- Alarm Activation: Upon a match, the program plays an alarm sound and prompts the user to snooze or stop.
- Snooze Functionality: Adds a predefined delay (2 minutes) to the current time and resets the alarm.
- Stop Functionality: Terminates the alarm and resumes clock display without further alerts.

5. Features

- Real-Time Clock Display: Continuously displays the current system time in the format HH:MM:SS on the console.
- Alarm Functionality: Allows the user to set an alarm time, which is triggered when the system time matches the set time.
- Input Validation: Ensures that the alarm time entered by the user is in a valid 24-hour format, preventing incorrect entries.
- Snooze Option: Offers the user a snooze option when the alarm rings, which delays the alarm by 2 minutes before ringing again.
- Stop Alarm Option: Enables the user to stop the alarm entirely when it is triggered.
- Colored Console Output: Uses different colors in the console display to highlight time, alerts, and prompts for better readability and user experience.
- Sound Alerts: Integrates audio playback using the PlaySound API to signal the user when the alarm is triggered or snoozed.
- User-Friendly Interface: Provides a clear and interactive text-based interface for setting alarms and managing alerts.

6. Code Highlights

Below are key procedures and functions that define the core logic of the application:

- InputAlarmTime – Accepts and validates user input for alarm time.
- GetCurrentTime – Retrieves and parses the system's current hour, minute, and second.
- UpdateTimeDisplay – Outputs the current time to the console with proper formatting and alignment.

- CheckAlarm – Compares system time with alarm time and decides whether to activate the alarm.
- TriggerAlarm – Handles alarm activation, sound playback, and user input for snoozing/stopping.
- HandleSnooze – Calculates new alarm time after snooze and resumes monitoring.

7. Conclusion

This project exemplifies how Assembly Language can be used to develop interactive, time-sensitive system applications. By leveraging the Irvine32 library and Windows system functions, a functional and user-friendly alarm clock was built, incorporating real-time display, sound alerts, snooze features, and robust input handling. The project also highlights the practical applications of low-level programming in building utilities that directly interact with hardware and the operating system, thus laying a strong foundation for embedded systems development.