Simple Math Game - Extended Project Documentation

1. Project Overview

1.1 Introduction:
This document provides a comprehensive description of the Simple Math Game implemented using the ATmega32 microcontroller. The game displays puzzles on an LCD, accepts user input via a keypad, and uses Timer0 interrupts to enforce a strict 10■second time limit per puzzle.

1.2 Goals: Create a time■bound interactive embedded math game. Use Timer0 in CTC mode to generate precise 10ms ticks. Design a clear level■based structure with scoring and game■over conditions. Demonstrate interrupt-driven programming on an AVR microcontroller. **1.3 Scope of Delivery:** Hardware pin assignment and block diagram description. Complete software design flow, including drivers and ISR logic. Testing and validation process. Extendable game architecture.

2. System Architecture

The system integrates hardware interfaces (Keypad, LCD) with software modules (Timer ISR, Puzzle Engine, Display Manager).

2.1 System Block Diagram Description: ATmega32 Microcontroller – Core CPU running game logic. **Keypad** – Captures user responses. **LCD 16x2** – Displays puzzles, score, and status. **Timer0 ISR** – Generates 10ms periodic interrupts. **Game Logic Engine** – Validates answers, controls flow. **Input Buffer** – Temporary storage for keypad digits. **Data Flow:**
Keypad → Input Buffer → Game Logic → LCD Output → Score System → Next Level Trigger.

3. Hardware Implementation and Pin Assignment

LCD Pin Assignment: PORTC – Data Lines (D4–D7) PORTD.0 – RS PORTD.1 – RW PORTD.2 – EN **Keypad Pinout:** Rows: PORTA.0 – PORTA.3 Columns: PORTA.4 – PORTA.7 **Timer0 Configuration:** Mode: CTC (Clear Timer on Compare) Prescaler: 1024 OCR0: 80 Interrupt: TIMER0_COMP_vect enabled **Why 80?**
80 × 0.128ms = 10.24ms ≈ 10ms tick.

4. Software Design and Communication Protocol

4.1 Software Modules: Main Application – Controls levels, scores, and game flow. **LCD Driver** – Print text and numbers to LCD. **Keypad Driver** – Matrix scanning and debouncing. **Timer0 Module** – Configures interrupts. **ISR Handler** – Tracks elapsed time for each puzzle. **Puzzle Engine** – Stores puzzles and verifies answers. **4.2 Level Progression Protocol:** Level 1 → Puzzle shown → Timer Start → User Answer → Validate. If correct and in time → Level 2. If wrong or timeout → Game Over. **4.3 Time-Out Detection Logic:**
flag1 increments every 10 ms.
flag1 == 1000 → 10 seconds elapsed → timeout.

5. Subsystem Details

5.1 Input System: Keypad rows driven LOW one at a time. Columns read to detect key press. Debouncing added via short software delay. **5.2 Display System:** LCD clear/update between levels. Shows puzzles, score, and timeout messages. **5.3 Puzzle Engine:** Stores puzzle expressions (e.g., 3 + ?? = 9). Compares user answer with correct value. Stores results for scoring. **5.4 Timer Subsystem:** ISR executes every 10ms. Tracks countdown timer. Stops when answer is submitted.

6. System Restoration and Testing

6.1 Testing Procedure: Test keypad digit capture and debouncing. Test LCD printing and cursor positions. Verify Timer0 interrupt frequency using stopwatch. Check correct and incorrect answer handling. Check game■over screen timing. **6.2 Boundary Case Testing:** Pressing extra digits. No key pressed within time. Incorrect input formats. **6.3 Future Enhancements:** Random puzzle generator. More levels. Buzzer alert on timeout. High■score storage in EEPROM.