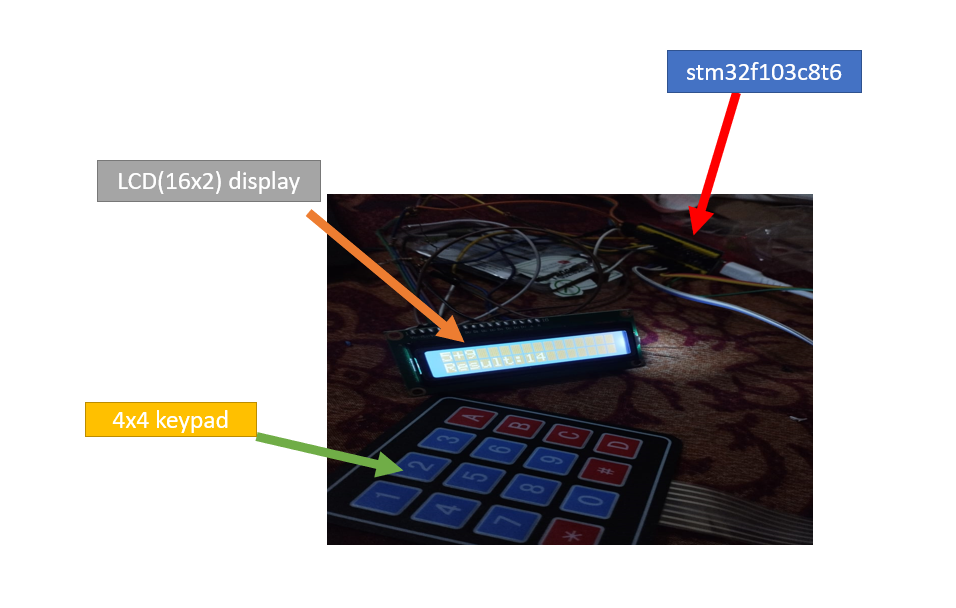
**Project Overview**

This project implements a calculator on an STM32F103C8 microcontroller, interfaced with a 16x2 LCD and a 4x4 keypad, without using I2C. It supports addition (+), multiplication (\*), and subtraction (-) operations, displaying inputs in real-time on the LCD and results upon pressing '='. The design leverages direct GPIO control for the LCD, a custom keypad scanning algorithm, and precise clock configuration, making it an excellent study in embedded systems programming.



## Features

* **Operations**: Addition (A button), Multiplication (B button), Subtraction (C button).
* **LCD Display**: Real-time input on row 0 (e.g., "123+456"), result on row 1 (e.g., "Result:579").
* **Clear Function**: Reset via 'C' button.
* **Debugging**: PA8 toggles every 500ms to confirm execution.

## Hardware

* **Microcontroller**: STM32F103C8 (Blue Pill).
* **LCD**: 16x2 HD44780-compatible, 4-bit mode (PA1–PA7).
* **Keypad**: 4x4 matrix (Rows: PA9–PA12, Columns: PB12–PB15).
* **Power**: 3.3V with LCD contrast via potentiometer.

### Keypad Layout

1 2 3 +

4 5 6 \*

7 8 9 -

C 0 # =

Fig: Keypad layout(modified)

### STM32 Programming

The project uses the STM32 HAL library for GPIO, timer, and system initialization. The main loop continuously scans the keypad, processes inputs, and updates the LCD, with error handling via an Error Handler function that sets PA8 HIGH if issues occur (e.g., clock misconfiguration).

### LCD Interfacing Without I2C

The 16x2 LCD is driven in 4-bit mode directly via GPIO pins (PA1–PA7), avoiding I2C for simplicity and direct control:

* **Pins**: RS (PA1), RW (PA2), EN (PA3), D4–D7 (PA7–PA4).
* **Initialization**:
  + A 50ms delay ensures LCD power-up stability.
  + Commands (0x33, 0x32, 0x28) configure 4-bit mode, 2 lines, and 5x8 font.
  + Additional commands (0x08, 0x01, 0x06, 0x0C) turn off the display, clear it, set entry mode, and turn it back on without a cursor.

**Data/Command Sending**:

* send\_to\_lcd() splits 8-bit data into two 4-bit nibbles, sent sequentially:
  + RS (0 for commands, 1 for data) selects the mode.
  + RW is grounded (PA2 LOW) for write-only operation.
  + D4–D7 carry the nibble, and EN pulses HIGH-to-LOW (20µs delay) to latch it.
* lcd\_send\_cmd() and lcd\_send\_data() handle commands (e.g., clear: 0x01) and characters (e.g., '1').
* lcd\_put\_cur() uses address offsets (0x00 for row 0, 0x40 for row 1) to position the cursor.

### Clock Configuration

The STM32’s clock is configured for reliable operation:

* **HSE**: 8MHz external crystal, multiplied by PLL (x9) to 72MHz system clock.
* **Dividers**: AHB (1), APB1 (2), APB2 (1) set HCLK to 72MHz, PCLK1 to 36MHz, and PCLK2 to 72MHz.
* **Flash Latency**: Set to 2 wait states for 72MHz operation.
* **TIM1**: Used for microsecond delays (e.g., LCD timing):
  + Prescaler: 72-1 (72MHz / 72 = 1MHz, 1µs per tick).
  + Period: 65535 (max count for flexibility).
  + delay() resets TIM1’s counter and waits until it reaches the desired µs value.

### 4x4 Keypad Working

The keypad is interfaced using a row-column scanning method:

* **Pin Mapping**:
  + Rows (R1–R4): PA9–PA12, configured as outputs, default HIGH.
  + Columns (C1–C4): PB12–PB15, configured as inputs with pull-ups.
* **Button Mapping**:
  + A (+): R1, C4; B (\*): R2, C4; C (-): R3, C4; D (=): R4, C4; Clear (C): R4, C1.
* **Scanning Algorithm**:
  + read\_keypad() iterates through rows, setting one LOW at a time (e.g., R1 LOW, others HIGH).
  + Columns are checked for LOW states (pull-up HIGH, button press pulls LOW).
  + Example: R1 LOW, C4 LOW → '+' (A button).
  + A 10ms delay before and after reading debounces the keys.
* **Logic**:
  + Rows output HIGH by default; pulling a row LOW activates it.
  + Columns use internal pull-ups, going LOW when connected to a LOW row via a pressed button.

### Calculator Logic

* **State Machine**:
  + State 0: Enter num1 (e.g., "123"), stored as an integer and in display\_buffer.
  + State 1: Operator (+, \*, -) entered, appended to display\_buffer.
  + State 2: Enter num2 digits, appended to display\_buffer (e.g., "123+456").
* **Calculation**:
  + '=' parses num2 from display\_buffer after the operator using atoi().
  + Result computed: num1 + num2, num1 \* num2, or num1 - num2.
* **Display**: Row 0 shows input; row 1 shows "Result:" followed by the result for 1s.
* **Clear**: 'C' resets all variables and the LCD.

## Procedure

1. **Hardware Setup**:
   * Connect LCD: RS (PA1), RW (PA2), EN (PA3), D4–D7 (PA7–PA4).
   * Wire keypad: Rows (PA9–PA12), Columns (PB12–PB15).
   * Power at 3.3V, adjust LCD contrast.

**Operation**:

* Enter "123 + 456 =", see "Result:579".
* Try "123 \* 4 =", see "Result:492".
* Test "500 - 123 =", see "Result:377".
* Press 'C' to clear.

**CODE:**

<https://github.com/Sadik107/STM32-Calculator-with-LCD-and-Keypad>

reff:

<https://controllerstech.com/interface-lcd-16x2-with-stm32-without-i2c/>