19CSE303 Embedded Systems

PROJECT

Controlling The Speed of DC Fan using STM32F103C8

Team Members:

G Chaitra Sai Chakravarthi CB.EN.U4CSE22514

G Rishi Deep CB.EN.U4CSE22515

Jeet Thakur CB.EN.U4CSE22518

Palla Uma Mahesh CB.EN.U4CSE22534

Project Abstract:

This project implements a temperature-based DC fan control system using the STM32F103C8 microcontroller . The system reads temperature values from a temperature sensor via the ADC and activates a DC fan when the temperature exceeds a user-defined threshold. The fan speed is adjustable through a potentiometer, allowing for real-time control of fan operation. The ADC data is displayed on a 16x2 LCD using an I2C module, providing a clear visual representation of the current ADC value. Additionally, an interrupt** is configured on PA0, and when triggered, it activates the fan connected to PC13. This interrupt-based control ensures that the fan can also be manually triggered for additional flexibility.

Communication Devices used in the Project : I2C

- I2C is a **multi-master** and **multi-slave** protocol. This means that a single STM32 microcontroller can act as the master and control multiple peripheral devices (slaves) on the same bus, each with a unique address.
- This is especially useful when you need to communicate with several sensors or other peripherals with a minimum number of GPIO pins.
- In I2C we are having 4 lines

- SCL – Carries the clock signal
- SDA – Serial Data Line
- Power
- Ground

Working Process of I2C:

- Start Condition: A master generates a start condition by pulling the SDA line low while the SCL line is high.
- Data Transfer: The ma
- ster generates the clock signal on the SCL line, and the data is transmitted on the SDA line, bit by bit. The devices on the bus read or write the data on the rising edge of the SCL clock.
- Stop Condition: When the communication is complete, the master generates a stop condition by pulling the SDA line high while the SCL line is high.
- I2C is connected to STM32F103C8 on **B6,B7** pins.
- When you want to display something on the LCD, the STM32 (acting as the **master**) sends commands and data via I2C to the LCD (acting as the **slave**).

ADC (Analog-to-Digital Converter) for Temperature Sensor:

- The STM32F103C8 has an integrated ADC that can read analog signals from sensors, such as the internal temperature sensor
- The sensor has a typical output voltage of around 1.43V at 25°C
- ADC is reading the values from the channel 16
- After the ADC conversion is complete, we are reading the resulting digital value from the **ADC data register** (ADC_DR), which gives you a value between 0 and 4095 .
- Then we are setting a threshold depending on that value we got from the ADC we are turning the DC Fan on and off
- We are using a potentiometer, to control the speed of the fan.
- We had set the ADC in continuous conversion mode, and we set the maximum sampling time to the channel 16.

NVIC (Nested Vector Interrupt Controller) :

- It is used for Interrupt Handling . It also allows for prioritization of interrupts, enabling higher-priority interrupts to preempt lower-priority ones.
- In our project we are using PA0 pin to trigger an interrupt and we are controlling the fan and also an led present at the PC13 when an interrupt is occurred.

- First we will configure PA0 as EXTI0, then we will configure NVIC to enable the interrupt for that pin.
- Then we will execute the ISR (turning the fan on) for EXTI0 .

Display Interface :

- We are using 16*2 LCD (Liquid Crystal Display) as the display interface in our project.

Pins Used:

- RS (Register Select)
- RW (Read/Write)
- EN (Enable)
- D4 to D7 (Data Pins for 4-bit mode)
- VSS (Ground), VCC (Power), and VO (Contrast)

LCD_INIT() : lcd initialization sequence in 4-bit mode

LCD_WriteCommand() : send command byte to the lcd

LCD_WriteData() : send data byte character to the lcd

LCD_DisplayMessage() : Loop the string and write each character to display

We are also some more functions like Set_Cursor(), LCD_Clear , etc .

The first row (Row 0) of the display can hold **16 characters**, and their addresses in memory are from 0x00 to 0x0F.

The second row (Row 1) also holds **16 characters,** but their addresses start from 0x40 and go up to 0x4F.

In the LCD Display we are displaying the real-time values of ADC

Exception Handling:

- Checking the **Over-temperature Exception (Analog Watchdog),** If the temperature reading exceeds a predefined threshold (based on the internal ADC reading), trigger an exception to stop the fan to prevent overheating.we will use the **Analog WatchDog** to monitor the temperature sensor values. The exception should trigger an interrupt if the temperature goes outside the defined range.
  Reference: Section *11.3.7*, Page *212*

- Checking for **ADC Overrun Exception,** in case the **ADC data register** is not read before a new conversion is complete, an **overrun** condition can occur. This can be detected using the OVR flag and managed by clearing the flag and resetting the conversion process.
  Reference: Section *11.12.1*, Page *228*

- Checking for **I2C Communication Failure with LCD** ,the project should handle potential communication issues between the STM32103C8 and the LCD display . If the LCD is unresponsive or data is not transmitted correctly, an error message or alternative display mechanism should be triggered.we need to handle communication errors when interfacing with the LCD through I2C.
  Reference: Section *9.2*, Page *163*

- Checking for **Power Supply Interruption Exception,**we are ensuring the fan and the board are protected against sudden power fluctuations.we will implement power-down handling using Power Control features like **Programmable Voltage Detector (PVD)** to detect low voltage conditions.
  Reference: Section *5.2*, Page *68*

- **Checking for ADC Calibration Failure Exception ,** During power-up or when the ADC is initialized, **calibration** is required to ensure accuracy. Failure in this calibration process could lead to inaccurate ADC readings. Calibration of the ADC is essential to correct internal capacitor mismatches. If the calibration fails, it could result in inaccurate temperature readings, which in turn will affect the DC fan's speed control based on incorrect temperature inputs.

Reference: Section *11.4*, Page *215*

We have taken the STM32F103C8 REFERENCE MANUAL AS REFERENCE. FOR ABOVE ALSO EXCEPTIONS ARE TAKEN FROM THE IDEA FROM THE REFERENCE MANUAL