Microprocessors and Interfacing

Spring-2022

Complex Engineering Problem-Report

"Digital Capacitance Measurement"

Students' Names: **1.** Muhammad Faiez Ali **2.** Izaan Mudassar **3.** Muhammad Maaz Khan	
Batch: BSEE 2020-24 **Section:** A	**Teacher:** Dr. Muhammad Aqil
Semester: 4th	**Group No.** 8

Department of Electrical Engineering

1.1 Abstract

This project is about to measure the capacitance of a capacitor with a STM-32 using the analog to digital converter & timer modules. The effective measurement range for the digital capacitance meter we'll be building is from 1nF to 100uF and the resolution about 0.5nF to a few hundreds at the high end of the range.

1.2 Introduction:

1.2.1 ADC:

Analogue to Digital Converter, or ADC, is a data converter which allows digital circuits to interface with the real world by encoding an analogue signal into a binary code.

Microprocessor-controlled circuits, Arduinos, Raspberry Pi, and other digital logic circuits need Analogue-to-Digital Converters (ADCs) to connect with the outside world. Many digital systems interact with their surroundings by monitoring the analogue signals from such transducers, which come from numerous sources and sensors that can measure sound, light, temperature, or movement in the actual world.

While analogue signals can be continuous and provide an endless number of voltage values, digital circuits function with binary signals that only have two discrete states: logic "1" (HIGH) or logic "0" (LOW). As a result, an electronic circuit is required to convert between the two domains of continuously changing analogue signals and discrete digital signals, which is where Analogue-to-Digital Converters (A/D) come into play.

In essence, an analogue to digital converter captures an analogue voltage at a single point in time and converts it to a digital output code that reflects the original value. The amount of binary digits (bits) utilized to represent this analogue voltage value is determined by the A/D converter's resolution.

For example, a 4-bit ADC will have a resolution of one part in 15, (24 – 1) whereas an 8-bit ADC will have a resolution of one part in 255, (28 – 1). Thus an analogue to digital converter takes an unknown continuous analogue signal and converts it into an "n"- bit binary number of 2n bits.

1.2.2 INTERUPT:

In digital computers, an interrupt (sometimes referred to as a trap) is a request for the processor to interrupt currently executing code (when permitted), so that the event can be processed in a timely manner. If the request is accepted, the processor will suspend its current activities, save its state, and execute a function called an interrupt handler (or an interrupt service routine, ISR) to deal with the event. This interruption is often temporary, allowing the software to resume[a] normal activities after the interrupt handler finishes, although the interrupt could instead indicate a fatal error.

Interrupts are commonly used by hardware devices to indicate electronic or physical state changes that require time-sensitive attention. Interrupts are also commonly used to implement computer multitasking, especially in real-time computing. Systems that use interrupts in these ways are said to be interrupt-driven.

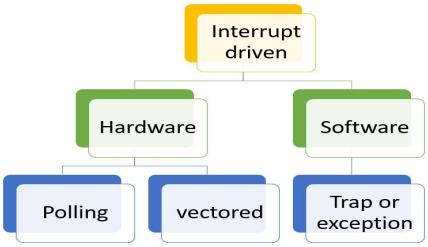


Figure 1.1 = Interrupt drive

1.2.3 TIMER:

Timers are small gadgets that allow you to set the time between occurrences. The ability to accurately set the timing between events is particularly frequent in embedded systems design. In multi-threaded operating systems, timers are used to determine how long a task is active before switching to another. Timers can be used to save energy by pulse width modulating (PWM) an LED. Timers can also be used to determine an analogue signal's sampling rate. Timers are used in almost every embedded software project.

A timer is a finite state machine that once each clock cycle adds or decrements a register. Every timer has a register that stores the current time value. Additionally, Timers also have a reload register. The reload register is used to set the period it takes for the timer expire. Most timers in a modern microprocessor will have more than two registers that dictate the behavior of the timer, but we can use these two registers to examine how timers allow us to measure a precise duration of time.

1.2.4 Digital Capacitance:

The functioning principle of a digital capacitance meter is based on the fact that a capacitor charges and discharges according to a specified equation (formula). As a result, we may apply a voltage through a given resistor R and measure the time it takes for a capacitor to reach a specific voltage threshold, which is usually represented in terms of tau, which is the Time Constant for capacitor charging and discharging.

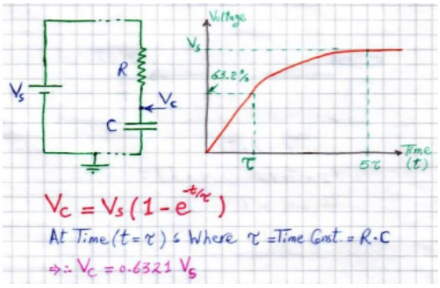


Figure 1.2 –The charging behavior of capacitor

One time constant (which is also equivalent to R x C) is the time it takes for a capacitor's voltage Vc to reach 63.2 percent of the source voltage. The sole unknown to answer for is the capacitance of the inserted capacitor C, given that we'll be selecting a handy R for this process. Then it's a simple matter of figuring it out.

To summarize, a digital capacitance meter applies a constant voltage source Vs to an unknown (to be measured) capacitor C via a constant known resistor R. It also monitors the time it takes for the voltage across the capacitor Vc to rise from 0v to a given threshold voltage (e.g., 1 or 2 or even fractions like 1.5). If the voltage threshold is 0.632Vs, the recorded time is equal to 1, which equals = R x C. We can solve for C and print out the capacitance value given the value of R.

1.3 Required Components:

1.3.1 Software

1. Kiel Micro vision
2. Stm-32 cube mx
3. ST-link Utility

1.3.2Hardware

1. Alphanumeric LCD Module
2. DC power Supply
3. Capacitor kit
4. Resistor kit
5. Jumper wires
6. Potentiometer
7. Push pull switches

1.4 LCD Configuration

LCD configuration is depicted in the following figure

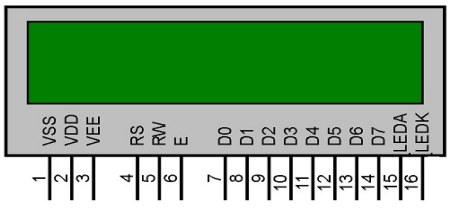
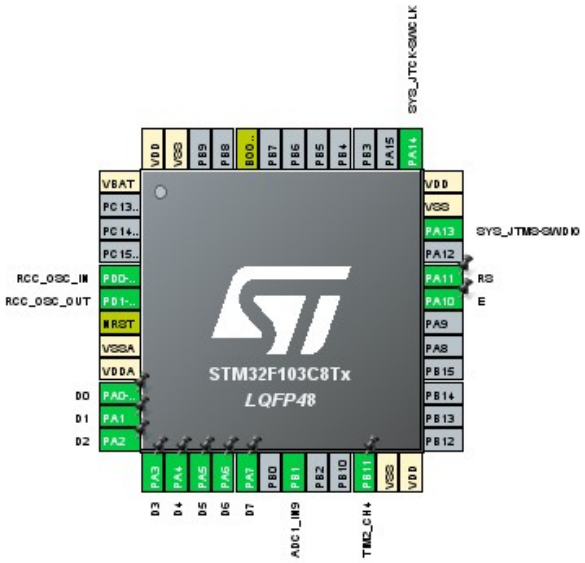


Figure – LCD 16x2 pinout configuration

This 16 pin device has two rows and can be accommodate sixteen characters each. IT HAS EIGHT data lines and three control lines that can be used for control purpose. By default,

LCD 16X2 is in 8 bit mode. We have used it in 4 bit by using LCD initialization and configuration.



Following pins configuration are considered

1. D0 to D7 pins are set in output mode for the data lines of LCD module.
2. A10 and A11 are set in output mode for the control lines of LCD module.
3. ADC was initialized in the channel 09, having following parameter settings

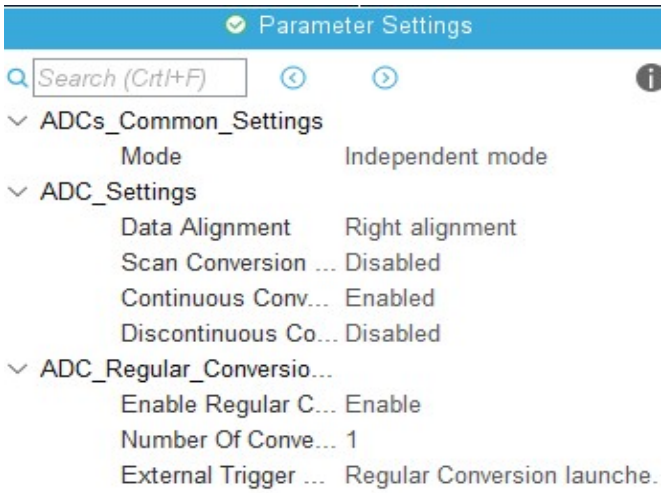


Figure – ADC paprameter settings

4. Timer module was set as internal clock. Following figure depicts its parameter settings

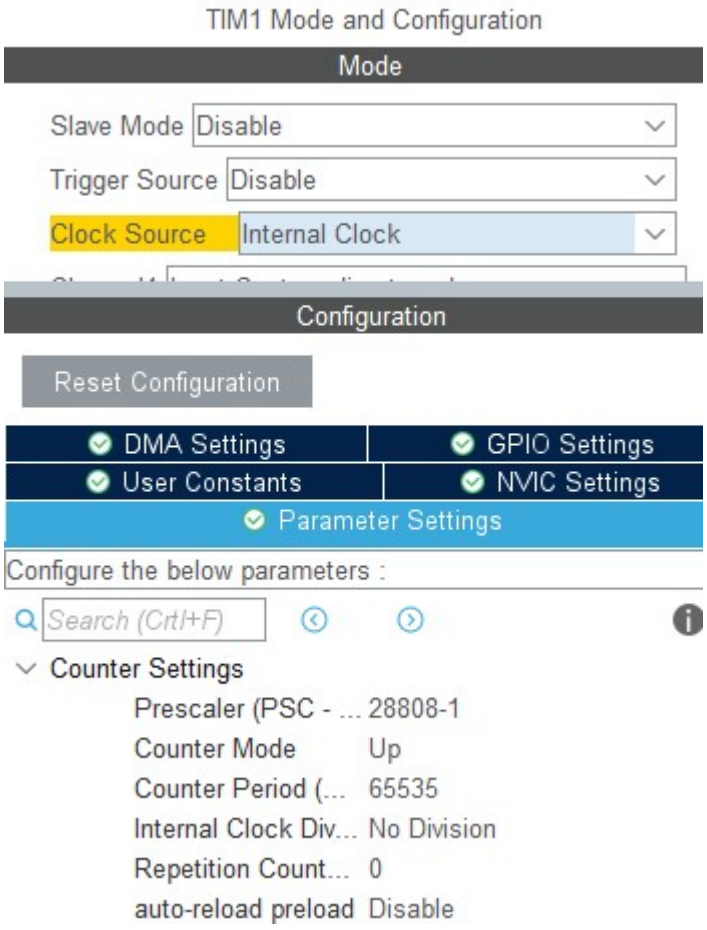


Figure – Timer mode configuration

The timer is also enabled with the trigger and commutation interrupt:

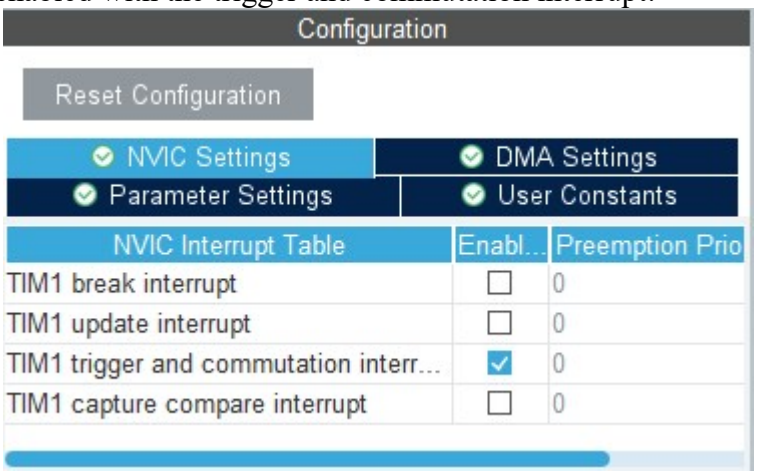
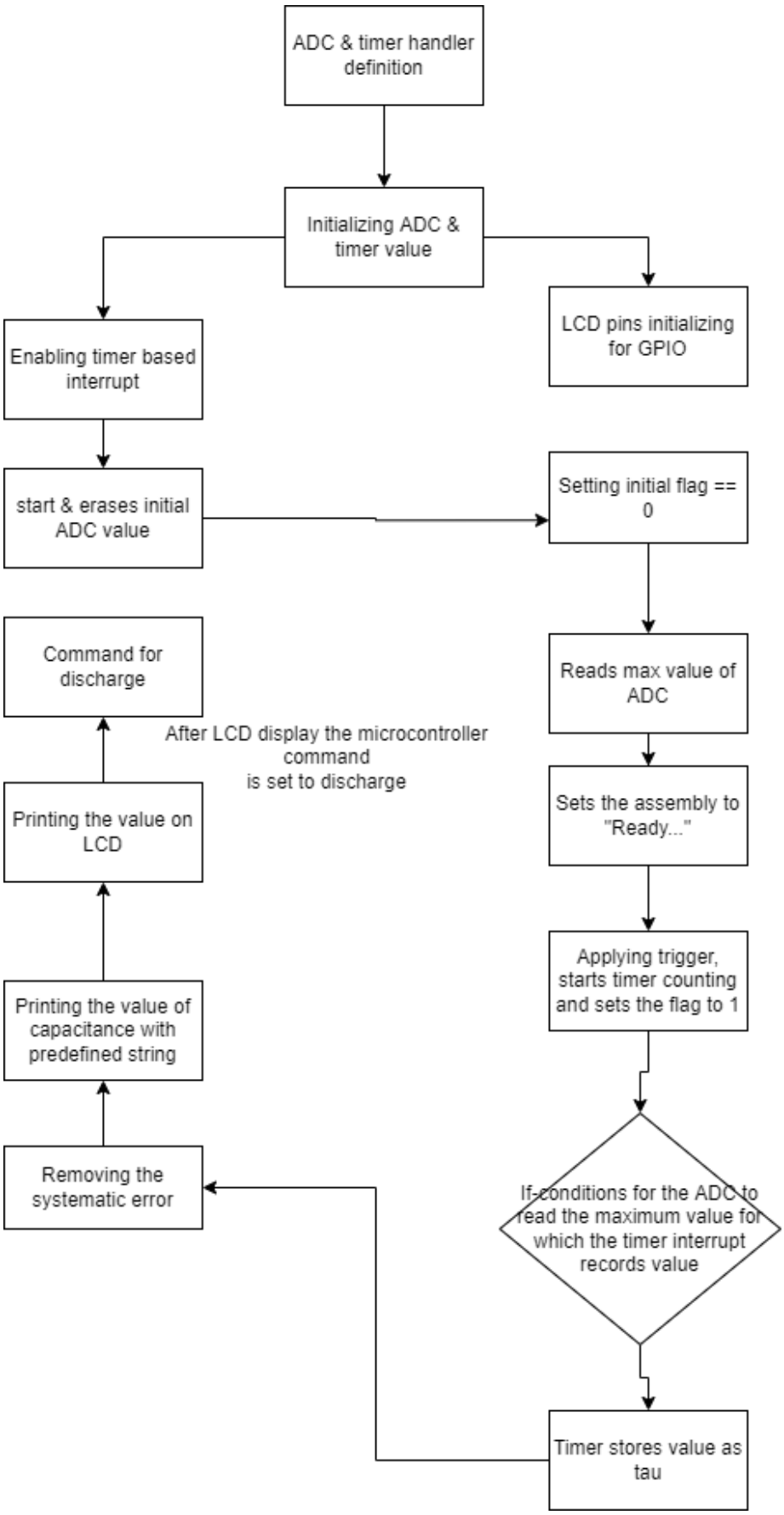


Figure – Trigger and Commutation interrupt

1.5 Flow chart



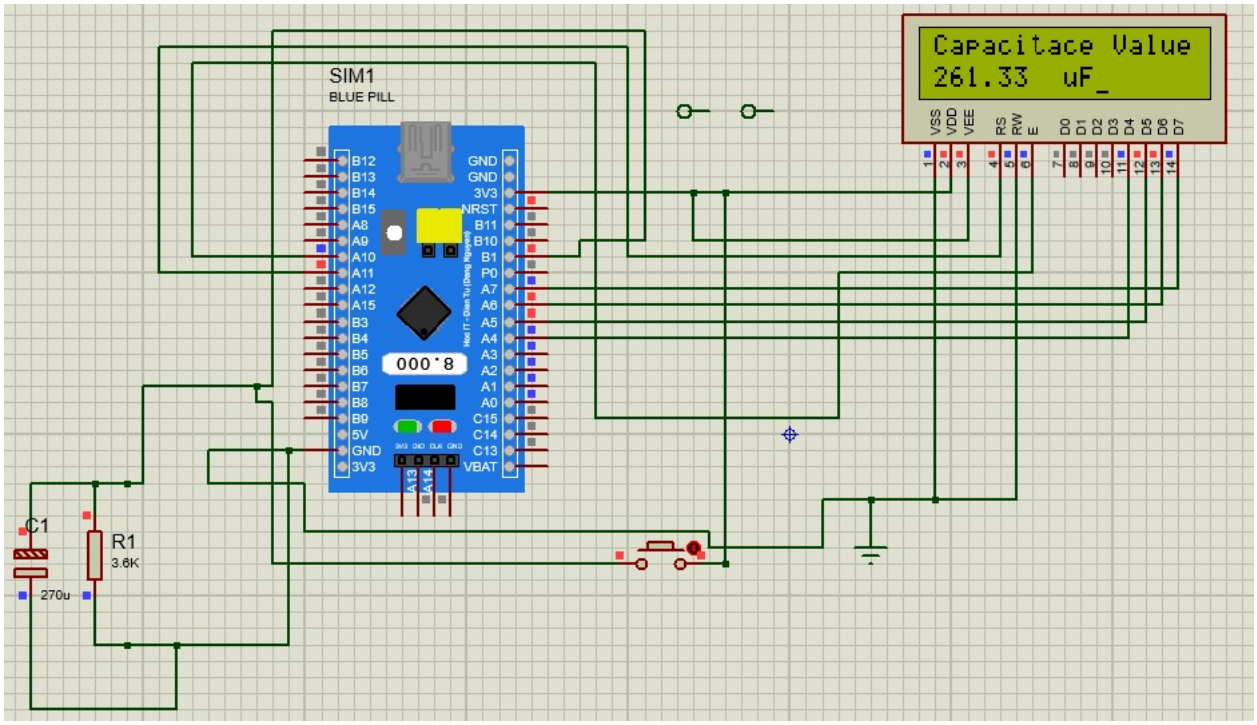
1.6 C code snippit:

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
 int a=1;
  LCD1602_1stLine();
  LCD1602_print("Digital Capcitor");
  LCD1602_2ndLine();
  LCD1602_print(" Measurement");
  LCD1602_1stLine();

  HAL_Delay(50);
  LCD1602_clear();
  double tau; char ValStr[16]; int flag=0;int input=0;
  HAL_TIM_Base_Start_IT(&htim1); // Timer starts here with the interrupt command enabled at Timer1

while (1)
{
            // ADC starts reading value as soon as the trigger is given

               LCD1602_noBlink();
              HAL_ADC_Start(&hadc1);
     adc_val = HAL_ADC_GetValue(&hadc1);
  if(flag==0)
  {
    // the maximum value read by the ADC comes to be 1376 which was then set as the threshold for timer
    if(adc_val==4095)

               LCD1602_1stLine();
              LCD1602_print(" Ready  "); //The timer interrupt triggers with the external push pull switch
               LCD1602_2ndLine();
              LCD1602_print("                 ");
             //LCD1602_2ndLine();
            // LCD1602_PrintInt(adc_val);
             HAL_Delay(1);
  }
```

```
//The max ADC value was divided into several if conditions for the amount of voltage drop observed when the triggger is applied
//the initial flag is predefined and the variable 'a' is applied to set the align of if-statments to be executed
  if(flag==1)
  {

          LCD1602_1stLine();
          LCD1602_print(" PROCRESSING..");
       LCD1602_2ndLine();
        LCD1602_print("         ");
         HAL_Delay(1);
  }
    // Here max value of ADC is 4095 i.e Vcc=5V and 0.693*Vcc=2837(calculated)
  //The timer keeps counting in this if-statmtement and the timer vlaue is stored in the pre-defined variable
     if(adc_val>=2837&&adc_val<=4095&&a==1)
     {
                              LCD1602_1stLine();
                            LCD1602_print("Timer counting");
                             LCD1602_2ndLine();
                            LCD1602_print("  ");
       __HAL_TIM_SET_COUNTER(&htim1,0);
      a=2;flag =1; //the flag is set to 1 and the function goes to processing statement just for delaying the time in which
      //interrupt reads the Timer value for the max ADC
     }
     //the timer comes back here for the remaining time for the capacitor to charge and holds the value with highest ADc vlaue
     if(adc_val>=1986&&adc_val<=2836&&a==2)
     {

                          timer_val = __HAL_TIM_GET_COUNTER(&htim1);

                              tau=((timer_val*0.4)*.278*21); // the value of timer is stored in a separate variable with Least Count factors
                                LCD1602_clear();
                              LCD1602_1stLine();
                             LCD1602_print("Capacitace Value");
                            LCD1602_2ndLine();
                            sprintf(ValStr,"%.*f",2,tau);
                           LCD1602_print(ValStr);
                         LCD1602_setCursor(2,8);
                         LCD1602_print("uF");
                           HAL_Delay(30);
       a=1;flag=0; input=0;
      while(input==0) // the pin 9 is triggered with the discharging path allowing the cappacitor to instantly discharge
```

```
    {
      input = HAL_GPIO_ReadPin (GPIOA, GPIO_PIN_9);
      HAL_Delay(3);
    }
    LCD1602_clear();
}
```

1.7 Circuit Schematics:



1.8 Discussion:

Obviously, the ADC conversion rate is rather limited by device specifications and moreover, the CPU takes time to service ADC interrupts to compare the current voltage level of Vc whether it's below 63.2% (2.2v) or not. Hence, we might miss (overshoot) the target voltage threshold and get a wrong measure for tau and consequently wrong capacitance value.

Add to that, a large resistance for R will result in a significant increase in tau and the charging time. Yielding a very long time required to measure a capacitor. For example, consider R=100kΩ, and you're willing to measure a capacitor C=1000µF. For this project, we'll stick to R=100kΩ. And limiting the C measurement range up to 10µF, at which the measurement time is a maximum of 1 second.

1.9 Conclusion:

Here in this project we have calculated the time constant (tau) is done in the ISR (interrupts service routine) whenever the ADC finds out that the Vc has crossed the 3.16v voltage threshold. Afterward, the unknown capacitance is easily calculated by dividing the resultant tau by R (the 100k resistor). And the result will be printed on the LCD inside the update display ( ) function.