MAJOR PROJECT

BY VAGGE SNEHA

[Date]

MAJOR PROJECT

"DIGITAL CLOCK USING A MICROCONTOLLER"

Microcontrollers are generally dedicated for a specific application. A microcontroller may take an input from the device it is controlling and controls the device by sending signals to different components in the device. Microcontroller normally consists a processor, memory, serial ports, and necessary logic circuits t o perform the specific function.

The advent of microcontroller technology has revolutionized the field of electronics and embedded systems, enabling the development of a wide range of applications. One such application is the integration of a 16X2 lcd display with an 8051 microcontroller to create a simple digital clock that also displays the current date. This project not only serves as a practical implementation of microcontroller programming and hardware interfacing but also highlights essential concepts in real-time operations, which are critical in many technological domains.

The primary goal of this project is to develop a time and date display system using the 8051 microcontroller. The system continuously tracks time in hours, minutes, and seconds, while also providing a date display in the DD/MM/YYYY format.

.

The implementation of this project involves several key components, including the 8051 microcontroller, a 16x2 LCD display, and supporting hardware such as a potentiometer for LCD contrast adjustment and a power supply.

A **microcontrolle**r is a compact integrated circuit designed to govern specific operations in embedded systems. It contains a processor (CPU), memory (RAM, ROM, or Flash), and input/output (I/O) peripherals, all on a single chip. Microcontrollers are the brain of many electronic systems, enabling automation and control in devices like household appliances, automobiles, and industrial machines.

1. **Central Processing Unit (CPU):** The CPU is the brain of the microcontroller, responsible for executing instructions from the program stored in memory. It performs arithmetic and logic operations and controls the flow of data within the system. here CPU processes all the instructions in the program, including updating the time and date, and sending data to the LCD.

2. Memory:

- RAM (Random Access Memory): Used for temporary data storage during program execution. it is used for storing temporary variables, such as the current value of the time and date.
- ROM (Read-Only Memory): Contains the program code and is non-volatile. where it holds the program code that governs the entire process of updating the time and interacting with the LCD.
- Flash Memory: A type of non-volatile memory used to store the program that can be reprogrammed.

3. Input/Output Ports (I/O): These pins allow the microcontroller to interface with external devices, such as sensors, motors, displays, and other components. They can either send data (output) or receive data (input). For I/O ports (specifically P2 and P3 in this case) are used to communicate with the LCD. These ports are responsible for sending control signals (e.g., read/write, enable) and transferring data to display the correct time and date.

4. Timers and Counters: These are used for timing operations, such as generating delays, counting external events, or controlling time-sensitive tasks like PWM (Pulse Width Modulation).

Timer1 in the 8051 is used to generate 1-second intervals, which is critical for incrementing the seconds, minutes, and hours in real time.

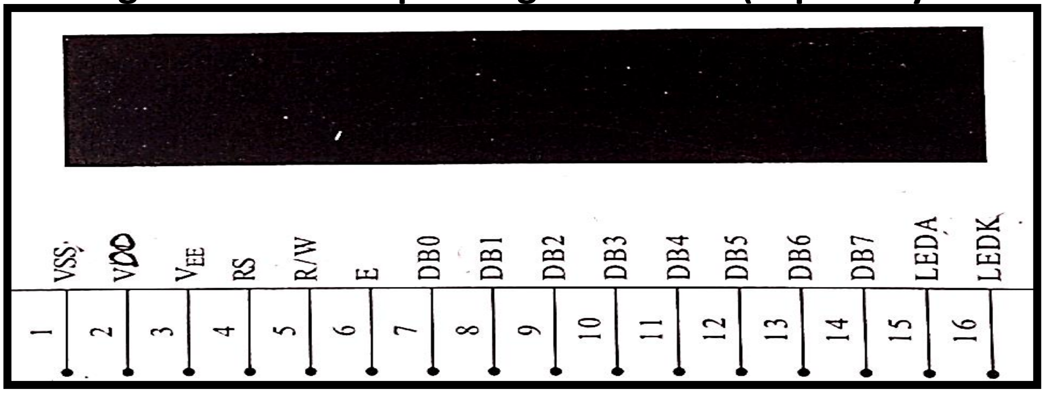
5. Analog-to-Digital Converter (ADC): Converts analog signals (e.g., sensor data like temperature or light) into digital form so the microcontroller can process them.

6. Serial Communication Interfaces: These are used to communicate with other devices, like a computer, sensors, or other microcontrollers. Common communication protocols include UART, SPI, I2C, and CAN.

7. Power Supply: Microcontrollers typically operate at low voltages like **5V or 3.3V**. They often include features like low-power modes for energy efficiency.

A **Liquid Crystal Display (LCD)** is a flat-panel display or electronic visual display that uses the light modulating properties of the crystals. Liquid Crystals do not emit light directly, they are available to display arbitrary images or fixed characters with low information content, which can be displayed such as present words, digits, 7-segment display as in digital clock.

The below figure shows the pin diagram of LCD (Liquid Crystal Display)



1. **VSS, VDD and VEE:** Pin 1 (VSS) is a ground pin and it is certainly needed that this pin should be grounded for LCD to work properly. VEE and VDD are given +5 Volts normally. However, VEE may have a potentiometer voltage divider network to get the contrast adjusted. But VDD is always at +5V.
2. **RS, R/W and E:** RS pin is used to select the registers of LCD for specific reading and writing operations. If we make the RS pin high and the put a data in the 8-bit data line (DB0 to DB7), the LCD module will recognize it as a data to be displayed. If we make RS pin low and put a data on the data line, the module will_recognize R/W pin is meant for selecting between read and write modes. High level at this pin enables read mode as a results micro controller reads the data from LCD and low level at this pin enables write mode as a result the microcontroller writes the data to LCD.

3. **E-PIN:** Enable [EJ pin is for enabling the module. A high to low transition at this pin will enable the module.

4. **DO-D7:** The 8-bit data pins, DO-D7, are used to send information to the LCD or read the contents of LCD.
5. **LEDA and LEDK:** LEDA is the anode of the back light LED and this pin must be connected to VCC through a suitable series current limiting resistor. LEDK is the cathode of the back light LED and this pin must be connected to ground.

❖ LCD MODULE COMMANDS FOR PROGRAMMING OF LCD

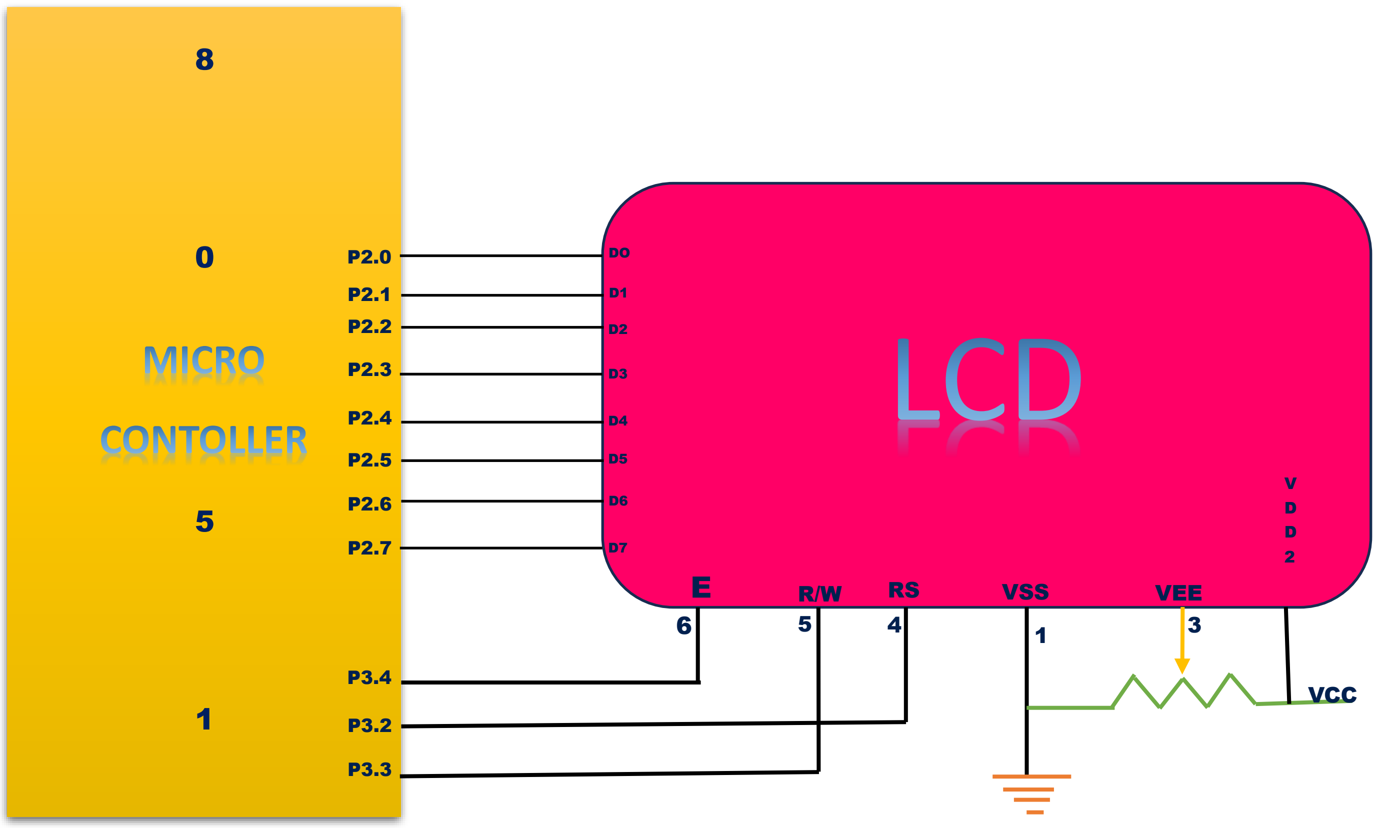
The 16*2 LCD module has a set of preset command instructions. Each command will make the module to do a particular task.

The commonly used commands and their function are given in the table below

Command	Function
OF	LCD ON, Cursor ON, Cursor blinking ON
01	Clear screen
02	Return home
04	Decrement cursor
06	Increment cursor

OE	Display ON, Cursor blinking OFF
80	Force cursor to the beginning of 1st line
C0	Force cursor to the beginning of 2nd line
38	Use 2Lines and 5*7 matrix
83	Cursor line 1 position 3
3C	Activate second line
08	Display OF F, Cursor OFF
C1	Jump to second line, position 1
0C	Display ON, Cursor OFF
C1	Jump to second line, position 1
C2	Jump to second line, position2

Schematic diagram of interfacing of LCD with 8051 to display the data



Hardware Detailed Connections:

1. Power Supply for the Circuit:

- VCC (Pin 40 of 8051): Connect to +5V from the regulated power supply.
- GND (Pin 20 of 8051): Connect to the Ground of the power supply.

2. Crystal Oscillator and Capacitors (Clock for the Microcontroller):

- Crystal Oscillator (12 MHz): Connect the two pins of the crystal oscillator to Pin 18 (XTAL2) and Pin 19 (XTAL1) of the 8051.
- 33pF Capacitors: Connect one end of each capacitor to Pin 18 and Pin 19. The other ends of the capacitors are connected to Ground.

3. Reset Circuit:

- Pin 9 (RST): Connect this pin to one end of a 10µF capacitor. The other end of the capacitor is connected to Ground.
- Connect a 10kΩ resistor between Pin 9 (RST) and VCC (5V) to create a reset-on-power-up circuit. You can also add a push button in parallel with the capacitor to manually reset the microcontroller.

4. 16x2 LCD Display Connections:

The 16x2 LCD has 16 pins that need to be connected to the 8051 microcontroller and power supply.

- Pin 1 (VSS): Connect to Ground (GND).
- Pin 2 (VDD): Connect to +5V.
- Pin3 (V0): Connect to the middle pin of a 10kΩ potentiometer to adjust the LCD contrast. The other two pins of the potentiometer are connected to VDD (+5V) and GND.
- Pin 4 (RS - Register Select): Connect to P3.2 of the 8051.
- Pin 5 (RW - Read/Write): Connect to P3.3 of the 8051. This will be set to ground in this case, as we are only writing to the LCD.
- Pin 6 (E - Enable): Connect to P3.4 of the 8051.
- Pins 7-14 (D0-D7 - Data pins): For simplicity, we are using the 8-bit mode to send data to the LCD.
- Connect D0-D7 of the LCD to P2.0 to P2.7 of the 8051.
- Pin 15 (LED+): Connect to +5V (for LCD backlight).
- Pin 16 (LED-): Connect to Ground.

5. Button for Manual Reset: Connect one side of a push button or resistor to the Pin 9 (RST) of the 8051 and the other side to GND. This allows the user to manually reset the microcontroller.

Software Detailed Connections:

Here's a detailed step-by-step guide on how to start debugging your code in Keil µVision, starting from opening the software:

Step 1: Install Keil µVision:

- If you don't have Keil µVision installed, you can download it from the official website: [Keil Downloads](https://www.keil.com/download/).
- Install the IDE by following the installation instructions.

Step 2: Create a New Project:

- Open Keil µVision.
- Go to Project → New µVision Project
- Select a directory to save your project and give it a name (e.g., `DigitalClock8051`), then click Save..
- In the Select Device dialog, choose your microcontroller family. For 8051-based projects, select Atmel → **AT89C51** (or any other 8051 variant you are using), and click OK.

Step 3: Add the C Code:

- After creating the project, you will be prompted to add a startup file. If asked, click No (we don't need a startup file for this project).
- Go to File → New to open a new code editor window.
- Copy and paste the complete C program into the new file.
- Go to File → Save As, and save the file with **a `.c` extension (e.g., `clock.c`).**

Step 4: Right-click on Source Group 1 in the Project Explorer window on the left.

- Select Add Existing Files to Group 'Source Group 1'.
- Select your **`clock.c`** file and click Add, then click Close.

Step 5: Configure the Target:

- Right-click on the project name in the Project Explorer and choose Options for Target 'Target 1'.
- In the Target tab, Ensure the Xtal (MHz) is set to **11.0592** MHz (or the crystal frequency of your microcontroller).
- In the Output tab, Check **Create HEX File** if you want to generate a HEX file for flashing onto the 8051 hardware. (select and click on create hex file)
- Click OK to save the settings.

Step 6: Compile the Program:

- Go to Project → Build Target or press the F7 key.
- Keil µVision will now compile the program. If there are no errors, you'll see a message in the output window saying **`Build: 0 Errors, 0 Warnings`**.

Step 7: Simulate the Program:

Keil µVision has a built-in simulator, which is useful for testing code before running it on actual hardware.

- Go to Debug→ Start/Stop Debug Session.
- You can now simulate the program and monitor the behavior of registers, memory, timers, and ports.

To see the LCD output simulation, you can manually observe the port pins (P2 and P3) to which the LCD is connected.

By following these steps, we are able to run, debug, and analyze your code within Keil µVision effectively!

SOURCE CODE (PROGRAM):

```
#include <reg51.h>    // Standard 8051 microcontroller register definitions


// Define LCD control pins

sbit RS = P3^2;  // Register Select

sbit RW = P3^3;  // Read/Write

sbit EN = P3^4;  // Enable


// Function Prototypes

void LCD_Command(unsigned char cmd);

void LCD_Char(unsigned char char_data);

void LCD_Init(void);

void Timer1_Init(void);

void delay(unsigned int ms);

void update_time(void);


// Global variables for time and date

unsigned char hours = 0;

unsigned char minutes = 0;

unsigned char seconds = 0;

unsigned char day = 1;

unsigned char month = 1;

unsigned int year = 2024;


// Main function

void main(void) {
```

```
LCD_Init();       // Initialize the LCD
Timer1_Init();    // Initialize Timer1 for 1-second time keeping

while (1) {
    // Display time on the LCD (HH:MM:SS)
    LCD_Command(0x80);  // Move cursor to beginning of first line
    LCD_Char((hours / 10) + '0');
    LCD_Char((hours % 10) + '0');
    LCD_Char(':');
    LCD_Char((minutes / 10) + '0');
    LCD_Char((minutes % 10) + '0');
    LCD_Char(':');
    LCD_Char((seconds / 10) + '0');
    LCD_Char((seconds % 10) + '0');

    // Display date on the LCD (DD/MM/YYYY)
    LCD_Command(0xC0);  // Move cursor to beginning of second line
    LCD_Char((day / 10) + '0');
    LCD_Char((day % 10) + '0');
    LCD_Char('/');
    LCD_Char((month / 10) + '0');
    LCD_Char((month % 10) + '0');
    LCD_Char('/');
    LCD_Char((year / 1000) + '0');
    LCD_Char(((year % 1000) / 100) + '0');
    LCD_Char(((year % 100) / 10) + '0');
```

```
        LCD_Char((year % 10) + '0');

        delay(1000);  // 1 second delay
    }
}

// Timer1 Initialization for 1-second interrupts
void Timer1_Init(void) {
    TMOD = 0x10;     // Timer1 in Mode1 (16-bit timer mode)
    TH1 = 0x3C;      // Load high byte for 1-second delay
    TL1 = 0xB0;      // Load low byte for 1-second delay
    ET1 = 1;         // Enable Timer1 interrupt
    EA = 1;          // Enable global interrupts
    TR1 = 1;         // Start Timer1
}

// Interrupt Service Routine for Timer1
void timer1_ISR(void) interrupt 3 {
    TH1 = 0x3C;  // Reload high byte for 1-second delay
    TL1 = 0xB0;  // Reload low byte for 1-second delay
    update_time(); // Call the function to update time
}

// Update time and date variables
void update_time(void) {
    seconds++;
    if (seconds >= 60) {
        seconds = 0;
        minutes++;
        if (minutes >= 60) {
            minutes = 0;
            hours++;
            if (hours >= 24) {
                hours = 0;
                day++;
                // Adjust days for different months
                if ((month == 4 || month == 6 || month == 9 || month == 11) && day >
30) {
                    day = 1;
                    month++;
                } else if (month == 2 && ((year % 4 == 0 && day > 29) || (year % 4 != 0
&& day > 28))) {
                    day = 1;
                    month++;
                } else if (day > 31) {
                    day = 1;
                    month++;
                }
                if (month > 12) {
                    month = 1;
                    year++;
                }
            }
        }
    }
}

// Delay function (approximate)
void delay(unsigned int ms) {
    unsigned int i, j;
    for (i = 0; i < ms; i++) {
        for (j = 0; j < 1275; j++);  // Approximate 1 ms delay at 12 MHz clock
    }
}

// LCD Initialization
void LCD_Init(void) {
    LCD_Command(0x38);  // Initialize in 8-bit mode
    LCD_Command(0x0C);  // Display ON, Cursor OFF
    LCD_Command(0x06);  // Auto-increment cursor
    LCD_Command(0x01);  // Clear display
    delay(5);           // Delay for LCD to initialize
}

// Send command to LCD
void LCD_Command(unsigned char cmd) {
    P2 = cmd;       // Send command to data pins
    RS = 0;         // RS = 0 for command
    RW = 0;         // RW = 0 for write
    EN = 1;         // High to Low pulse on EN
    delay(1);
    EN = 0;
}

// Send character to LCD
void LCD_Char(unsigned char char_data) {
    P2 = char_data;  // Send data to data pins
    RS = 1;          // RS = 1 for data
    RW = 0;          // RW = 0 for write
    EN = 1;          // High to Low pulse on EN
    delay(1);
    EN = 0;
}
```

This procedure could extend the utility of the project beyond a simple clock, transforming it into a comprehensive time management system. Once the hardware is assembled and code is uploaded to the microcontroller, the LCD displays the current time on the first line and the current date on the second line. The time is formatted as HH:MM:SS, while the date appears as DD/MM/YYYY. This user-friendly interface allows individuals to easily read and comprehend the information being presented.

NOTE:

- CLICK ON THESE FOR DIRECT INTERFACES:

main.c   aaaaaaa.uvproj

- TO COPY THE CODE USE BELOW ATTACHED "COPY FILE"