**A PROJECT REPORT ON**

DIGITAL CLOCK USING 8051 MICROCONTROLLER

**PREPARED BY**

***ADITYA KUMAR***

**ABSTRACT**

This project aims to design and implement a digital clock using the 8051 microcontroller. The

digital clock displays the current time in hours, minutes, and seconds on a set of 7-segment

displays. It includes functionality for setting the time using push buttons and accurately keeps

track of time using timer interrupts.

**INTRODUCTION**

The primary objective of this project is to create a functional digital clock using the 8051

microcontroller. The clock should display the current time, allow for time adjustment through

push buttons, and maintain accurate timekeeping.

A digital clock using a microcontroller is a modern timekeeping device that utilizes a

microcontroller, a small computing device, to accurately display and update the current time.

Here's a brief explanation of how it works.

**Microcontroller:** The heart of the digital clock is a microcontroller, such as an Arduino or PIC,

which is programmed to perform specific tasks, including timekeeping.

**Real-Time Clock (RTC) Module:** To maintain accurate time, the clock often employs an RTC

module. This module includes a dedicated clock crystal and circuitry to track time

independently of the microcontroller.

**Display:** Digital clocks commonly use LED, LCD, or OLED displays to show the time in a user

friendly format, such as hours and minutes. Some may also display seconds, date, or other

information.

**User Interface:** Depending on the design, the clock may feature buttons, a touchscreen, or even

voice commands to set the time and customize settings.

**Programming:** The microcontroller is programmed to read the time from the RTC module and

update the display accordingly. It also manages other functionalities, like alarm settings, time

formats, and daylight-saving adjustments.

**Power Supply:** The clock is typically powered by a battery or an external power source,

depending on whether it's intended for portable or stationary use.

**Accuracy:** Thanks to the RTC module, digital clocks can maintain precise timekeeping, even in

the absence of a continuous power supply. Some clocks synchronize with internet time servers

for even greater accuracy.

**HARDWARE COMPONENT USED**

▪ **8051 Microcontroller**

▪ **11.0592 MHz Crystal Oscillator**

▪ **LCD or 7-Segment display**

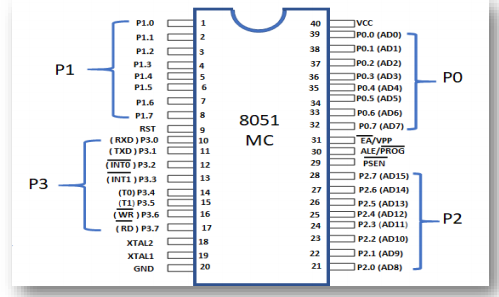
▪ **Resisters, Transistors, capacitors**

▪ **Push Buttons**

▪ **Jumper Wires**

▪ **Battery or Adapter**

▪ **REAL-TIME CLOCK**



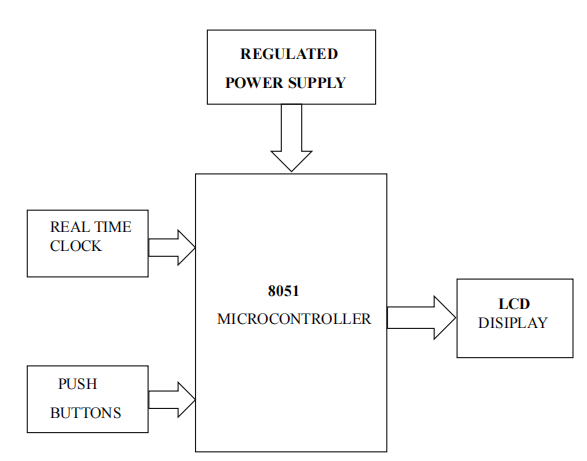
**8051 MICROCONTROLLER PIN DIAGRAM**

**SOFTWARE USED**

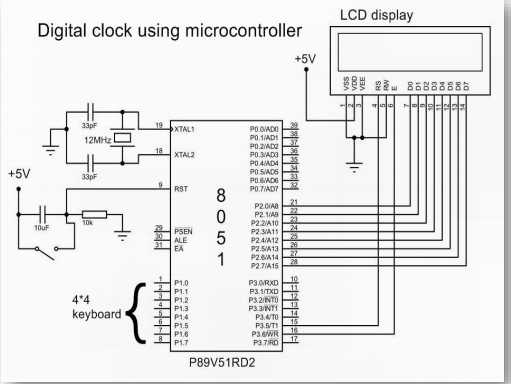
▪ **KEIL**

▪ **PROTEUS**

**BLOCK DIAGRAM**



**CIRCUIT DIAGRAM**



**DIGITAL CLOCK CIRCUIT DIAGRAM BY USING 8051 MICROCONTROLLER**

**PROGRAM**

#include<reg51.h>

#include<absacc.h>

#define dataport P2

#define port P1

#define lcdport P3

sbit reset = port^0;

sbit rs = port^1;

sbit rw = port^2;

sbit e = port^3;

sbit dig\_hr1 = port^4;

sbit dig\_min1 = port^5;

sbit start = port^6;

int min1 = 0, hr1 = 0;

int min0 = 60, hr0 = 25;

unsigned char temp = 60 , hr , min , sec , num[ 60] ={0X00 , 0X01 , 0X02 ,

0X03 , 0X04 , 0X05 , 0X06 , 0X07 , 0X08 , 0X09 , 0X10 , 0X11,0X12,

0X13 , 0X14 , 0X15 , 0X16 , 0X17 , 0X18 , 0X19 , 0X2O ,0X21,0X22,

0X23,0X24 , 0X25 , 0X26 , 0X27 , 0X28 , 0X29 , 0X30 , 0X31 , 0X32,

0X33, 0X34 , 0X35 , 0X36 , 0X37 , 0X38 , 0X39 , 0X40 , 0X41 ,0X42,

0X53 , 0X54 , 0X55 , 0X56 , 0X57 , 0X58 , 0X59};

void delay(unsigned int msec )

{

Int i , j ;

for (i=0 ; i < msec ; i++)

for (j=0 ; j < 1275 ; j++);

}

Void lcd\_cmd(unsigned char item )

{

dataport = item ;

rs = 0 ;

rw = 0 ;

e = 1 ;

delay(1) ;

e =0 ;

return ;

**}**

void lcd\_data(unsigned char item)

{

dataport = item ;

rs = 1 ;

rw = 0 ;

e = 1 ;

delay ( 1) ;

e = 0;

return ;

}

void lcd\_string(unsigned char \* str)

{

int i = 0 ;

while ( str [ i ] ! = ‘\ 0’)

{

lcd\_data (str [ i ] ) ;

i++ ;

delay ( 1) ;

}

return ;}

lcd\_int(int time\_val)

{

int int\_amt ;

int \_amt = time\_val/10 ;

lcd\_data(int\_amt+48);

int\_amt = time\_val % 10;

lcd\_data(int\_amt +48) ;

}

void lcd ( )

{

lcd\_cmd(0x38);

delay ( 5 ) ;

lcd\_cmd ( 0x0C) ;

delay ( 5) ;

lcd\_cmd(0x80) ;

delay (5) ;

}

void set\_rtc\_time( )

{

XBYTE[ 10] =0x20 ;

XBYTE[ 11] =0x82 ;

XBYTE[ 0] =0x00 ;

XBYTE[ 2] =min ;

XBYTE[ 4] =hr ;

XBYTE[ 7] =0x01 ;

XBYTE[ 8] =0x01 ;

XBYTE[ 9] =0x10 ;

XBYTE[1 ] =0xFF ;

XBYTE[ 3] =0xFF ; XBYTE[ 5] =0xFF ;

XBYTE[ 11] = 0x22 ;

}

void set\_hr 1( )

{

hr1++;

if(hr1>23)

hr 1 = 0 ;

lcd\_cmd(0xc3);

lcd\_int(hr 1) ;

lcd\_data( ‘:’) ;

hr0 = hr1;

}

void set\_min1( )

{

min1++ ;

if (min1>59)

min1 = 0 ;

lcd\_cmd(0xc6) ;

lcd\_int(min1) ;

min0 = min 1 ;

}

void set\_time( ) interrupt 2

{

lcd\_cmd(0x01);

if (start==0);

{

lcd\_string(“ SET TIMING”) ;

lcd\_cmd(0xc3) ;

lcd\_int(hr 1) ; lcd\_data(‘:’) ;

lcd\_int(min1);

while(start==0)

{

delay(10);

if(dig\_hr 1 == 0)

set\_hr1( ) ;

if(dig\_min1==0)

set\_min1( ) ;

}

}

lcd\_cmd(0x01) ;

hr = num [ hr 1] ;

min = num [min 1] ;

set\_rtc\_time( ) ;

lcd\_cmd(0x80) ;

lcd\_string(“TIME : ”) ;

hr0 = 25 ;

min = 60 ;

}

bcdconv( unsigned char mybyte )

{

unsigned char x,y ;

x= mybyte & 0x0F ;

x=x | 0x30 ;

lcd\_data( y) ;

lcd\_data ( x) ;

}

void read\_rtc\_display( )

{ XBYTE[ 11]=0x02 ;

hr = XBYTE[ 4] ;

lcd\_cmd(0x85);

{

bcdconv(hr) ;

lcd\_data(‘:’) ;

hr0 = hr ;

}

min = XBYTE [ 2 ];

{

bcdconv( min) ;

lcd\_data( ‘:’ ) ;

}

sec= XBYTE[0];

{

bcdconv( sec) ;

}

}

void main( )

{

reset = 1;

lcd ( ) ;

XBYTE[ ] = 0x20 ;

XBYTE[ ] = 0xFF ;

XBYTE[ ] = 0xFF ;

XBYTE[ ] = 0xFF ;

XBYTE[ ] = 0x02 ;

lcd\_cmd(0x01) ;

IE = 0x84 ;

lcd\_cmd(0x80) ;lcd\_string(“ TIME”) ;

while ( 1 )

{

read\_rtc\_display( ) ;

}

**PROCEDURE :**

1. Initially burn the program to the 8051 microcontroller

2. Now give the connections as per the circuit diagram.

3. Switch on the board supply.

4. Now you can observe the time on LCD. If you want to set the time make the start pin

low and press the push button connected to the P3.3.

5. LCD shows set time message. Now use push button which is connected to P1.4 to set

hours and use other push button to set minutes.

6. Now make start pin high to run the clock.

**APPLICATIONS**

**Home Automation:** Use the digital clock to control lights, thermostats, and appliances at specific

times, making it an integral part of a home automation system.

**Public Transportation:** Digital clocks at bus stops or train stations can display arrival and

departure times, helping commuters plan their journeys.

**Sports Timing:** Digital clocks are commonly used in sports events to time races, matches, and

games accurately.

**Industrial Control:** In manufacturing environments, digital clocks are crucial for synchronizing

operations and scheduling production runs.

**Laboratories:** Digital clocks in research laboratories help scientists and researchers time

experiments and record data.

**Public Spaces:** Digital clocks in parks, airports, and other public spaces provide a convenient

way for people to check the time.

**Traffic Management:** Digital clocks can be integrated into traffic light systems to synchronize

traffic flow and control signals.

**Public Safety:** In emergency response centers, digital clocks are used to coordinate responses

and track the timing of critical event .

We can also set the Alarm , Countdown timer , Stopwatch , Date display , Temperature and

Humidity , Remainders , etc.., in this project with little modification .

**CONCLUSION**

The project of designing a digital clock using the 8051

microcontroller has provided a valuable learning experience and a practical

application of microcontroller programming and electronics. This project has

successfully demonstrated .