MICROCONTROLLER-BASED CODE LOCK

■ K. S. SANKAR

Code locks can be constructed using digital and timer ICs employing pushbuttons or keypads for entering the code for authentication and operation of the code lock. However, such circuits would require a large number of ICs.

On the other hand, a microcontroller-based code lock will require very few peripheral components. With the cost of microcontrollers now dropping to the equivalent cost of approximately four digital ICs, it makes sense to design simple logic circuits using microcontrollers and free version of programming languages. Although free-version-language code length is normally limited to around 2 kB, but that is adequate for small projects like this one.

This simple code lock project is based on a 20-pin ATMEL microcontroller AT89C2051. It employs a 4-digit sequential code with time-out security feature. In addition to the microcontroller, the circuit uses a single additional IC (CD4050) and a transistor to drive a relay. Although the project uses a liquid-crystal display (LCD), it is useful for design and developmental purpose only and is not really an essential part of the circuit. The same can be removed from the circuit without any change in the software.

As regards LCD modules, these are available in 14- or 16-pin packages. The 16-pin variety has an additional backlight option. Popular brands available in India are Lampex, Hantronix and Hitachi. Most other models also have the same pin configuration. The model used in this project is Lampex LM16200 16-character×2-line alphanumeric dot-matrix display with backlight option. However, you may also use any other branded/unbranded LCD for the purpose.

Circuit description

As already mentioned, the project makes use of ATMEL AT89C2051 microcontroller, in 20-pin DIP package, which supports 2 kB of flash-based program memory. A 6MHz crystal is used for providing the clock. Port-1 of the microcontroller is used to drive the LCD in 4-bit mode with 10-kilo-ohm pull-up resistors. The 10-kilo-ohm potentiometer controls the contrast of the LCD panel. It works better when its wiper is nearer to ground potential.

Timer 0 of 89C2051 is used as an internal counter that increments a variable every second. This variable is used in the project to time out the delay for entering the code.

After initialisation, the software

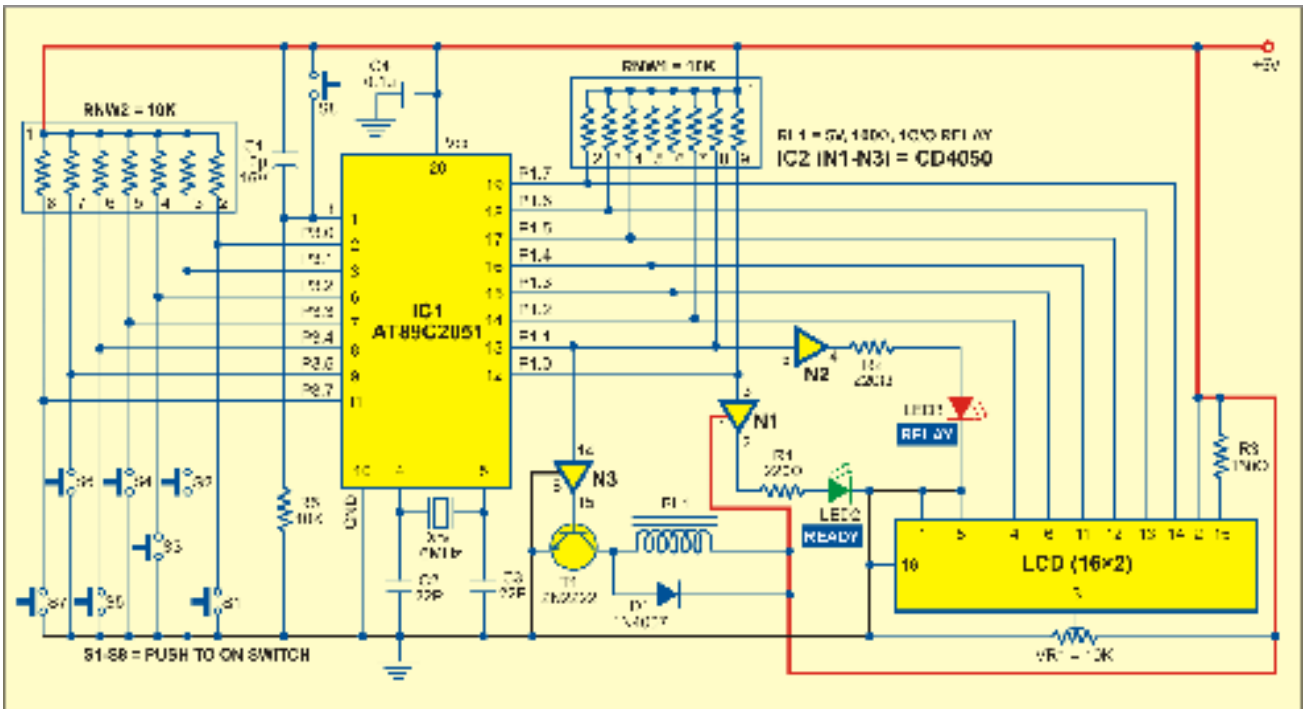

Fig. 1: Circuit of microcontroller-based code lock

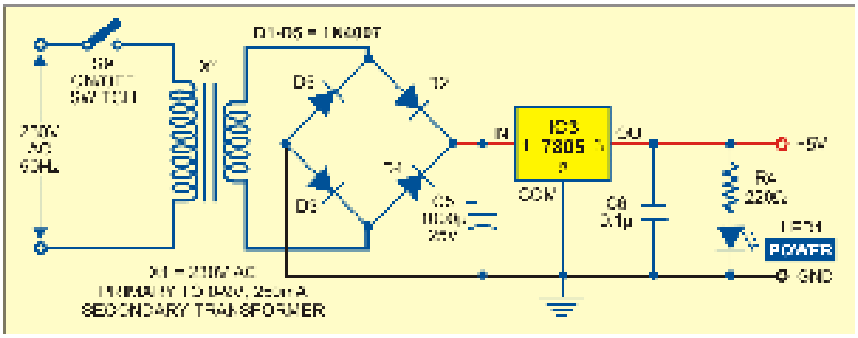

Fig. 2: Power supply

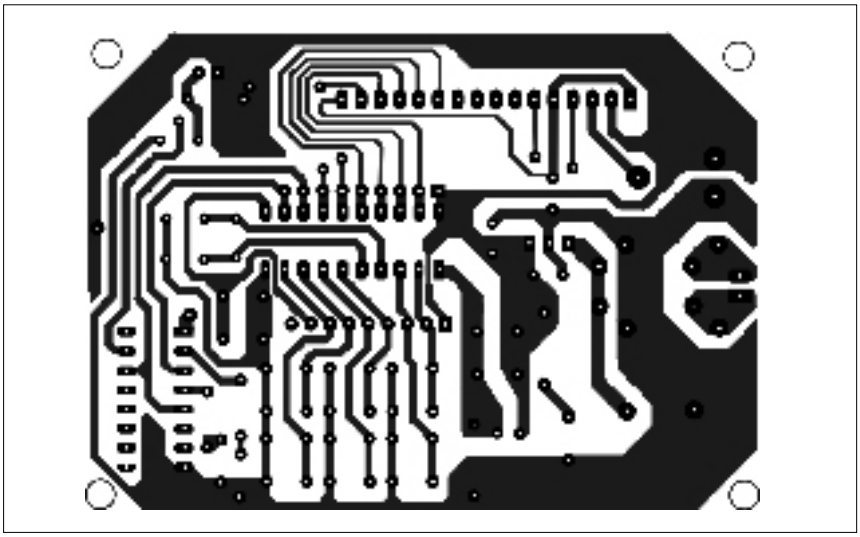

Fig. 3: Actual-size, single-side PCB of microcontroller-based code lock

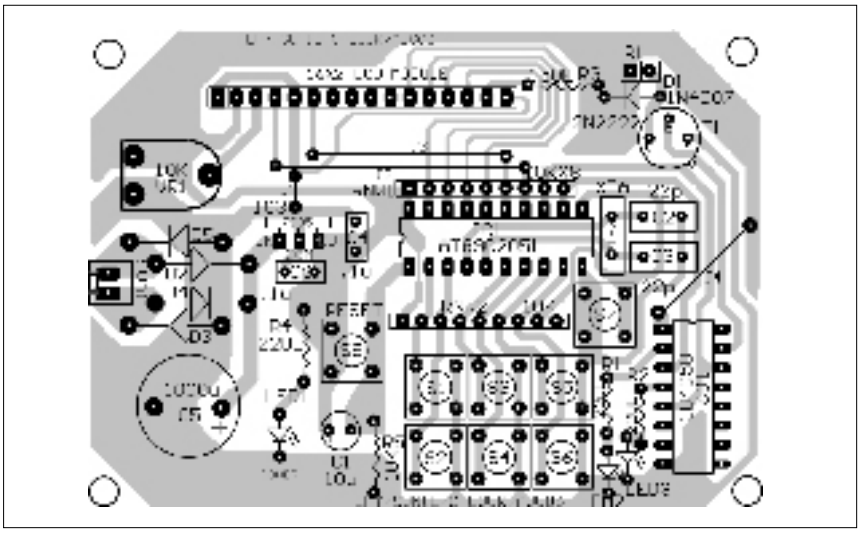

Fig. 4: Component layout for the PCB

switches on 'Ready' LED and waits for a 4-digit code to be entered. The valid code for this project is '1324.' The code is entered using the seven input switches that are connected to port-3. Port-3 does not have the bit 'P3.6' and hence the same is ignored by the software. Two LEDs at port-1 are interfaced to P1.0 and P1.1 pins to provide 'Ready' and 'Relay On' indication via respective LEDs.

The P1.1 line is also interfaced to relay driver transistor T1 through a buffer to switch on a 12V relay, which can activate an electrically operated lock.

Timer 0 is started on the first keystroke to validate the remaining three digits, provided these digits are entered within five seconds. If not, the software loops back to the initial state. After three unsuccessful attempts, the circuit will wait for about 10 seconds (before accepting the next keystroke) to avoid unwanted tampering attempts. All these timings can be changed through the software program to suit your specific requirements.

Power supply (Fig. 2). A conventional regulated power supply circuit employing a step-down transformer followed by bridge rectifier, smoothing capacitor and 5V regulator is used to meet the supply requirement for the code lock circuit shown in Fig. 1.

PARTS LIST

Semiconductors:

IC1	- AT89C2051 microcontroller
IC2	- CD4050 non-inverting buffer
IC3	- 7805 +5V regulator
T1	- 2N2222 npn transistor
D1-D5	- 1N4007 rectifier diode
LED1-LED3	- 5mm LED

Resistors (all ¼-watt, ±5% carbon):

R1, R2, R4	- 220-ohm
R3	- 150-ohm
R5	- 10-kilo-ohm
RNW1, RNW2	- 10-kilo-ohm resistor network
VR1	- 10-kilo-ohm preset

Capacitors:

C1	- 10µF, 16V electrolytic
C2, C3	- 22pF ceramic disk
C4, C6	- 0.1µF ceramic disk
C5	- 1000 µF, 25V electrolytic

Miscellaneous:

X_{TAL}	- 6MHz crystal
RL1	- 5V, 100-ohm, 1C/O relay
S1- S8	- Push-to-on switch
S9	- On/off switch
X1	- 230V AC primary to 9V, 250mA secondary transformer
	- 16-character×2-line LCD (male-53 pins used)
	- Bergstick connector (female- 36 pins used)
	- 1m long, 4-core ribbon

An actual-size PCB layout for the code-lock including the power supply is shown in Fig. 3 and its component layout in Fig. 4.

The software

The software is written using BASCOM-51. (You may find details about it in my article 'Real-Time Clock Using Microcontroller' published in Jan. 2005 issue of EFY magazine.) The source code file 'EFYCLOCK.BAS' for the project in BASCOM-51 is given at the end of the article. Although the program is self-explanatory, you need to be aware of BASCOM51 compiler directives and syntax of statements, which are available within the help menu of BASCOM compiler. BASCOM contains a lot of statements to control various peripherals including the LCD.

The BOSCOM compiler/IDE can be used to generate a hex file, which should be 'burnt' into the chip using any universal programmer. The hex code of the program is only 1.5k long, while AT89C2051 microcontroller can take up to 2k of code. This program may be modified to suit your specific requirement.

EFY note. All files pertaining to this project have been carried in this month's EFY-CD, while BASCOM-8051 package has already been included in the EFY-CD for May 2005.

EFYCLOCK.BAS

```
' pass = 1 4 2 8 bin ( 1324) decimal

$crystal = 6000000
$regfile = "89c2051.dat"

Dim I As Byte
Dim K(4) As Byte
Dim Pass(4) As Byte
Dim Key As Byte
Dim Invalid_pass As Bit
Dim Sec_count As Byte
Dim Clock_word As Word
Dim Passtime As Byte
Dim Attempts As Byte
Dim Maxattempts As Byte

Ready_led Alias P1.0
Relay_out Alias P1.1

For I = 1 To 4
K(i) = 0
Next I

Pass(1) = 1
Pass(2) = 3
Pass(3) = 2
Pass(4) = 4

Sec_count = 0
Passtime = 5
Attempts = 0
Maxattempts = 3

Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = P1.4 , Db5 = P1.5 , Db6 = P1.6 , Db7 = P1.7 , E = P1.3 , Rs = P1.2
'port 1

P1 = 0
P3 = 255

Config Timer0 = Timer , Gate = Internal , Mode = 2
'Timer0          use timer 0
'Gate = Internal   no external interrupt
'Mode = 2         8 bit auto reload

' set t0 internal interrupt  2000 times a sec
On Timer0 Timer_0_overflow_int
Load Timer0 , 250
Priority Set Timer0
Enable Interrupts
Enable Timer0


Begin:
If Attempts >= Maxattempts Then
Locate 0 , 0 : Lcd Maxattempts ; " attempts over"
Locate 2 , 0 : Lcd "try after 10 seconds"
Attempts = 0
Gosub Trylater
End If

Sec_count = 0

For I = 1 To 4
K(i) = 0
Next I

Cls
Cursor On Blink
 'clear the LCD display
Lcd "Enter Pass:"
'display this at the top line

Ready_led = 1

For I = 1 To 4

While 1 = 1

If Sec_count > Passtime Then
Exit For
End If

If P3 <> 255 Then
' some key pressed - check it

If I = 1 Then
' start timer0  on first keystroke
Sec_count = 0
Start Timer0
End If

Key = P3
 ' wait for key release
  While Key = P3
  Wend
K(i) = 255 - Key
If K(i) = 1 Then
Goto Lcd_out
End If
If K(i) = 2 Then
Goto Lcd_out
End If
If K(i) = 4 Then
K(i) = 3
Goto Lcd_out
End If
If K(i) = 8 Then
K(i) = 4
Goto Lcd_out
End If
If K(i) = 16 Then
K(i) = 5
Goto Lcd_out
End If
If K(i) = 32 Then
K(i) = 6
Goto Lcd_out
End If
If K(i) = 128 Then
K(i) = 7
Goto Lcd_out
End If

' invalid key combination
Key(i) = 0

Lcd_out:

Lcd K(i)
Waitms 30
Exit While
End If

Wend


Next I

Ready_led = 0

Stop Timer0

' check if time over
If Sec_count > Passtime Then
Locate 2 , 0 : Lcd "time over"
Incr Attempts
Gosub Error_flash
Wait 1
Goto Begin
End If


' check valdity
Invalid_pass = 0
For I = 1 To 4
If K(i) <> Pass(i) Then
Invalid_pass = 1
End If
Next I

If Invalid_pass = 1 Then
Goto Invalid
End If


Valid:
Locate 2 , 0 : Lcd "valid password"
Relay_out = 1
Wait 3
Relay_out = 0
Goto Begin

Invalid:
Locate 2 , 0 : Lcd "invalid"
Gosub Error_flash
Incr Attempts
Wait 1

Goto Begin


Trylater:
' wait for 10 seconds
For I = 1 To 10
Wait 1
Key = P3
Key = 255 - Key
 If Key = 3 Then
 Exit For
 End If
Next I

Wait 2
Return

Error_flash:
For I = 1 To 10
Ready_led = Ready_led Xor 1
Waitms 100
Next I
Ready_led = 0
Return


' interrupt subroutine ---------
Timer_0_overflow_int:
' program comes here 2000 times a sec with a 6mhz xtal
 Incr Clock_word

If Clock_word > 2000 Then
Clock_word = 0
Incr Sec_count
End If
Return

End
```