

Fig. 3: Flow-chart for countdown timer

suitable for Atmel AT89C2051 chip.

Circuit description

The circuit of the countdown timer is shown in Fig. 1. The microcontroller used is Atmel AT89C2051 (IC1), which is a 20-pin device with 2 kB of program memory. Port 1 is used to drive two 7-segment displays through ICs CD4511 (IC2 and IC3), which are BCD-to-7-segment converters. A 6MHz crystal is used for timing. Timer 0 is used as an internal counter and increments a variable every second. This variable is used in the project for providing accurate timing.

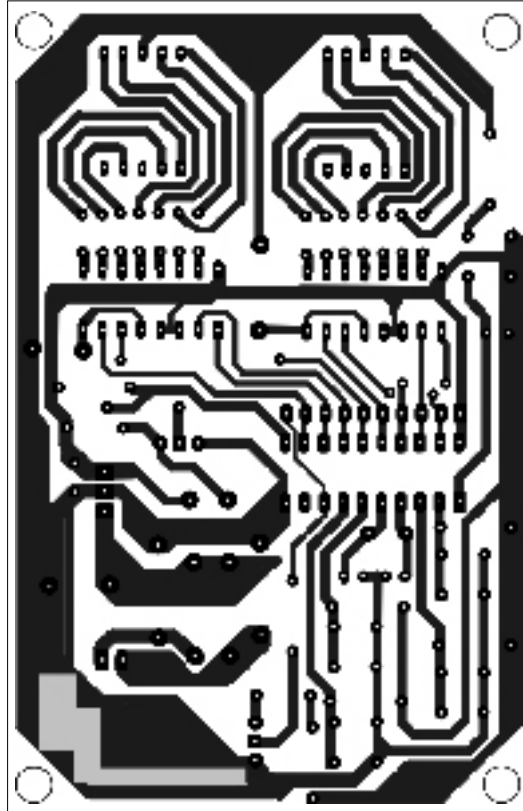


Fig. 4: An actual-size, single-side PCB layout for AT89C2051-based countdown timer

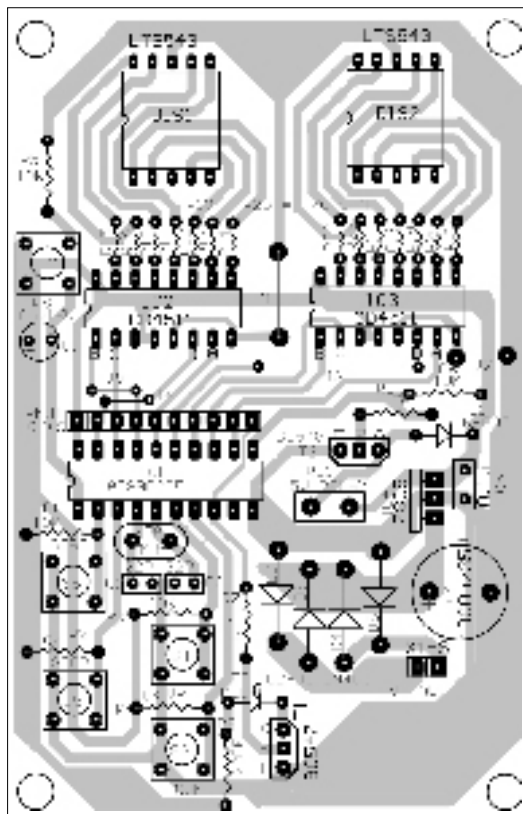


Fig. 5: Component layout for the PCB

PARTS LIST

Semiconductors:

IC1	- AT89C2051 microcontroller
IC2, IC3	- 4511 BCD-to-7-segment latch/decoder/driver
IC4	- 7805 5V regulator
T1, T2	- BC547 npn transistor
LED1	- Red LED
D1-D5	- 1N4007 rectifier diode
DIS1, DIS2	- LTS543 common-cathode, 7-segment display

Resistors (all 1/4-watt, ±5% carbon):

R1-R6	- 10-kilo-ohm
R7, R8	- 4.7-kilo-ohm
R9	- 220-ohm
R10-R23	- 470-ohm
RNW1	- 10-kilo-ohm resistor network

Capacitors:

C1	- 10µF, 16V electrolytic
C2, C3	- 22pF ceramic disk
C4	- 1000µF, 35V electrolytic
C5	- 0.1µF ceramic disk

Miscellaneous:

X1	- 230V AC primary to 9V AC, 250mA secondary transformer
S1	- On/off switch
S2-S6	- Push-to-on tactile switch

The software waits for 'Start' switch to be pressed to start timing operation. It can be stopped anytime by pressing 'Stop' switch momentarily.

'Up' and 'Down' set switches are used for setting the time (in minutes), as displayed on 7-segment display. This function is directly handled by interrupts 0 and 1 in the software. The Start, Stop, Up and Down switches are connected to port 3. Port 3 does not have the bit p3.6 and it is ignored.

A flashing LED connected to port 3.4 shows that the timing activity is in progress.

Relay energisation pin 11 is connected to a driver transistor to switch on a 5V relay that can activate any electrical device. (A different external voltage (9 to 12V) can also be used to power the relay and driver transistor T2, after disconnecting the 5V supply at the junction of relay RL1 and the cathode of D1.)

The BCD code for unit's is output at pins P1.0 through P1.3 and for ten's at pins P1.4 through P1.7. All these eight pins are pulled high through 10-kilo-ohm resistors of RNW1. These pins are coupled to 'A' through 'D' input pins

of BCD-to-7-segment decoder driver IC3 (for unit's) and IC2 (for ten's). The segment-driving outputs of IC3 and IC2 are coupled to 7-segment, common-cathode displays DIS2 (unit's) and DIS1 (ten's), respectively.

The 5V regulated power supply for the circuit is provided by a conventional circuit comprising step-down transformer X1, which steps down mains 230V AC to 9V AC. This output is rectified by a bridge rectifier comprising 1N4007 diodes D2 through D5

followed by 1000µF smoothing capacitor C4 feeding regulator 7805 (IC4). C5 is used for bypassing any ripple from the output of the regulator.

An actual-size, single-side PCB for the circuit of countdown timer (Fig. 1) including its power supply (Fig. 2) is shown in Fig. 4 and its component layout in Fig. 5.

Software

The software is written using BASCOM-51. (For detailed informa-

tion about it, please go through the author's 'Real-time Clock Using Microcontroller' article published in Jan. 2005 issue of EFY.) The flow-chart for the timer is shown in Fig. 3. The project can be converted into a 0-99 second timer by making suitable changes in the source code.

The source program cdtimer.bas in BASCOM-51, along with suitable comments, is given below. The same is also included in this month's EFY-CD.

CDTIMER.BAS

```

'-----
' 99 min countdown relay timer
' language used: BASCOM-51 from www.mcselec.com
' Micro controller used= Atmel AT89C2051
' - by K.S.Sankar www.mostek.biz
' 16-1-2006 '-----
' define crystal speed
$crystal = 6000000
$regfile = "89c2051.dat"

' define variables
Dim I As Byte
Dim Sec_count As Byte
Dim Min_count As Byte
Dim Clock_word As Word
Dim Setmode As Bit

' declare function used
Declare Sub Fn7seg(i As Byte)
Dim i As Byte

'-----
' declare interrupt routines
On Int0 Int0_int
On Int1 Int1_int
Enable Interrupts
Enable Int0
Enable Int1
'enable the interrupts
'-----

' define alias names for start / stop switches
Sw_start Alias P3.0
Sw_stop Alias P3.1
' up / down switches are connected to int0 and int1
'Switch_up P3.2 ( int0)
'Switch_down P3.3 ( int1)

Relay_out Alias P3.7
Led_out Alias P3.4

' make ports 0
P1 = 0
P3 = &B00111111
' p1 port to ic4511 bcd -> 7 seg convertor ( 2 displays)
' p3 as input and output port

' configure timer0
Config Timer0 = Timer , Gate = Internal , Mode = 2
'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
' dont start timer0 here

Begin:
' wait for sw-start press
' or interrupts up/down to take place
Setmode = 0
Relay_out = 0
Led_out = 0
Sec_count = 0

Min_count = 0
I = 0

'-----
Begin1:
Call Fn7seg(i)

If Sw_start = 0 Then
Goto Begin2
End If

If Sw_stop = 0 Then
While Sw_stop = 0
Wend
Relay_out = 0
Sec_count = 0
Goto Begin
End If

Goto Begin1

Begin2:
' relay on
Setmode = 0
Relay_out = 1
Start Timer0

Begin3:
I = 99 - Min_count
Call Fn7seg(i)

If I = 0 Then
Goto Over
End If

If Sw_start = 0 Then
Start Timer0
Setmode = 0
End If

If Sw_stop = 0 Then
Goto Over
End If

Goto Begin3

Over:
Stop Timer0
Relay_out = 0
Goto Begin

'end of main program

'----- function below -----
Sub Fn7seg(i As Byte)
Dim _ans As Byte
' display on two 7 seg
_ans = Makebcd(i)
P1 = _ans
If Setmode = 1 Then
' if in set mode make display flicker
P1 = 255
' blankout the display
Waitms 30
' turn it on again
P1 = _ans
Waitms 30
End If

End Sub

' 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
' A Flashing Led When Timing Is In Progress
' 1 sec on and 1 sec off
Led_out = Led_out Xor 1
End If

If Sec_count = 60 Then
Sec_count = 0
Incr Min_count
End If
Return

'-----
Rem The Interrupt Handler For The Int1 Interrupt
Int1_int:
'DOWN
Stop Timer0
Setmode = 1
Incr Min_count
If Min_count >= 99 Then Min_count = 98
I = 99 - Min_count
Call Fn7seg(i)
Waitms 100
Return

'-----
Rem The Interrupt Handler For The Int0 Interrupt
Int0_int:
' UP
Stop Timer0
Setmode = 1
Decr Min_count
If Min_count = 255 Then Min_count = 0
If Min_count = 0 Then Min_count = 99
I = 99 - Min_count
Call Fn7seg(i)
Waitms 100
Return

' this program when compiled creates a binary file
' of just 802 bytes with only 8 variables defined in
' the program
' if such a user friendly language can create compact
code
' I wonder why people still struggle to write in op
codes or
' languages full of semi-collons.....
' that is left to the reader to c
' end of program ----- written in bascom-51
'-----
' end of program -----

```