Rs 200/-
(with CD)

ELECTRONICS PROJECTS

VOL. 24

A Compilation of 91 tested Electronic Construction Projects and Circuit Ideas for Professionals and Enthusiasts

ISBN 978-81-88152-19-3

PUBLISHED BY EFY
ISO 9001:2000 CERTIFIED

Electronics Projects
Vol. 24

First Published in this Edition, November 2007

ISBN 978-81-88152-19-3

Published by Ramesh Chopra for EFY Enterprises Pvt Ltd,
D-87/1, Okhla Industrial Area, Phase-1, New Delhi 110020.
Typeset at EFY Enterprises Pvt Ltd and
Printed at J.K. Offset & Packages, C-21, DDA Shed,
Okhla Phase-1, New Delhi 110020.

ELECTRONICS PROJECTS VOL. 24

EFY Enterprises Pvt Ltd
D-87/1 Okhla Industrial Area, Phase-1
New Delhi 110020

EFY Books & Publications
FOR YOU

EFY is a reputed information house, specialising in electronics and information technology magazines. It also publishes directories and books on several topics. Its current publications are:

(A) CONSTRUCTION PROJECTS

1. **Electronics Projects, Vol. 1:** A compilation of selected construction projects and circuit ideas published in Electronics For You magazines between 1979 and 1980.
2. **Electronics Projects, Vol. 2 to 19 (English version):** Yearly compilations (1981 to 1998) of interesting and useful construction projects and circuit ideas published in Electronics For You.
3. **Electronics Projects, Vol. 20, 21, 22, 23 and 24 (with CD):** Yearly compilations (1999 to 2003).
4. **Electronics Projects, Vol. 16 (हिन्दी संस्करण):** Yearly compilations (1995) of interesting and useful construction projects and circuit ideas published in Electronics For You.

(B) OTHER BOOKS

1. **Learn to Use Microprocessors (with floppy):** By K. Padmanabhan and S. Ananthi (fourth enlarged edition). An EFY publication with floppy disk. Extremely useful for the study of 8-bit processors at minimum expense.
2. **ABC of Amateur Radio and Citizen Band:** Authored by Rajesh Verma, VU2RVM, it deals exhaustively with the subject—giving a lot of practical information, besides theory.
3. **Batteries:** By D.Venkatasubbiah. This publication describes the ins and outs of almost all types of batteries used in electronic appliances.

(C) DIRECTORIES

1. **EFY Annual Guide (with CD):** Includes Directory of Indian manufacturing and distributing units, Buyers' Guide and Index of Brand Names, plus lots of other useful information.
2. **Technical Education Institutes Directory:** Includes course-wise and state/city-wise listings of technical education institutes in India, besides the alphabetical main directory offering all the relevant information about them.

(D) MAGAZINES

1. **Electronics For You (with CD & without CD):** In regular publication since 1969, EFY is the natural choice for the entire electronics fraternity, be it the businessmen, industry professionals or hobbyists. From microcontrollers to DVD players, from PCB designing software to UPS systems, all are covered every month in EFY.
2. **Linux For You ((with CD & DVD):** Asia's first magazine on Linux. Completely dedicated to the Open Source community. Regular columns by Open Source evangelists. With columns focused for newbies, power users and developers, LFY is religiously read by IT implementers and CXOs every month.
3. ***'i.t.'* (Information Technology):** A monthly magazine for 'Techies' and those who want to be. Its readers have two things in common—a background related to IT and the thirst to know more. Topics covered boast technical depth and aim to assist in better usage of IT in organisations.
4. **Facts For You:** A monthly magazine on business and economic affairs. It aims to update the top decision makers on key industry trends through its regular assortment of Market Surveys and other important information.
5. **BenefIT:** A technology magazine for businessmen explaining how they can benefit from IT.
6. **Electronics Bazaar:** A monthly B2B magazine for sourcing electronics components, products and machineries. Ideal for buying decision makers and influencers from electronics and non-electronics industry.

★ *Registered Post or Courier Delivery for Books and CDs: Rs 40 for first copy and Rs 15 for every additional copy of any book or directory.* ***Add Rs 50/- on an outside Delhi cheque. Important.***

★ *Payment should be sent strictly in advance by demand draft/money order/postal order in favour of EFY associates* ***Kits'n'Spares.***

For retail orders:

Kits'n'Spares
D-88/5, Okhla Industrial Area,
Phase-1, New Delhi 110020
Phone: 26371661, 26371662
E-mail: kits@efyindia.com
Website: www.kitsnspares.com

For magazine subscriptions:

EFY Enterprises Pvt Ltd
D-87/1 Okhla Industrial Area, Phase-1
New Delhi 110020
Phone: 26810601-03
Fax: (011) 26817563
E-mail: info@efyindia.com

For bulk orders:

Paramount Book Agency
Arch No. 30 (West Approach)
Mahalaxmi, Mumbai 400034
Phone: (022) 24925651, 24927383
Fax: 24950392
E-mail: circulations@ibhworld.com

FOREWORD

This volume of Electronics Projects is the twenty forth in the series published by EFY Enterprises Pvt Ltd. It is a compilation of 22 construction projects and 69 circuit ideas published in 'Electronics For You' magazine during 2003.

We are also including a CD with this volume, which not only contains the datasheets of major components used in construction projects but also the software source code and related files pertaining to various projects. This will enable the reader to copy these files directly on to his PC and compile/run the program as necessary, without having to prepare them again using the keyboard. In addition, the CD carries useful software, tutorials and other goodies (refer 'contents' in CD).

In keeping with the past trend, all modifications, corrections and additions sent by the readers and authors have been incorporated in the articles. Queries from readers along with the replies from authors/EFY have also been published towards the end of relevant articles. It is a sincere endeavour on our part to make each project as error-free and comprehensive as possible. However, EFY cannot resume any responsibility if readers are unable to make a circuit successfully, for whatever reason.

This collection of tested circuit ideas and construction projects in a handy volume, would provide all classes of electronics enthusiasts—be they students, teachers, hobbyists or professionals—with a valuable resource of electronic circuits, which can be fabricated using readily-available and reasonably-priced components. These circuits could either be used independently or in combination with other circuits, described in this and other volumes. We are confident that this volume, like its predecessors, will generate tremendous interest amongst the readers.

CONTENTS

Section A: Construction Projects

Section B: Circuit Ideas:

SECTION A :
CONSTRUCTION PROJECTS

CAR SECURITY SYSTEM WITH REMOTE CONTROL

T.K. HAREENDRAN

Here is an inexpensive electronic car security system with remote control that uses readily available, low-cost components. The gadget comprises a base unit, which remains fitted inside the vehicle, and a remote control handset for activating/deactivating the base unit. The base unit, in enabled state, can be used to set off an alarm device when an unauthorised person tries to gain access to the vehicle.

Remote control handset

The remote control handset (see Fig.1) is used to switch on/off the base unit installed inside the vehicle. When push-to-on switch S1 on the handset is depressed, the circuit wired around PLL tone decoder LM567 (IC1) energises and IR-LED emits infrared light modulated at around 4.5 kHz. Transistor T1 works as the IR LED driver, while resistor R3 maintains the IR-LED current within safe limits. Note that only the VCO section of IC1 is used here.

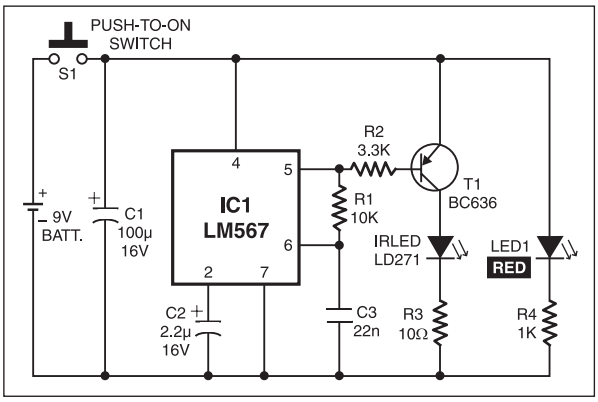


Fig. 1: Circuit diagram of remote control handset

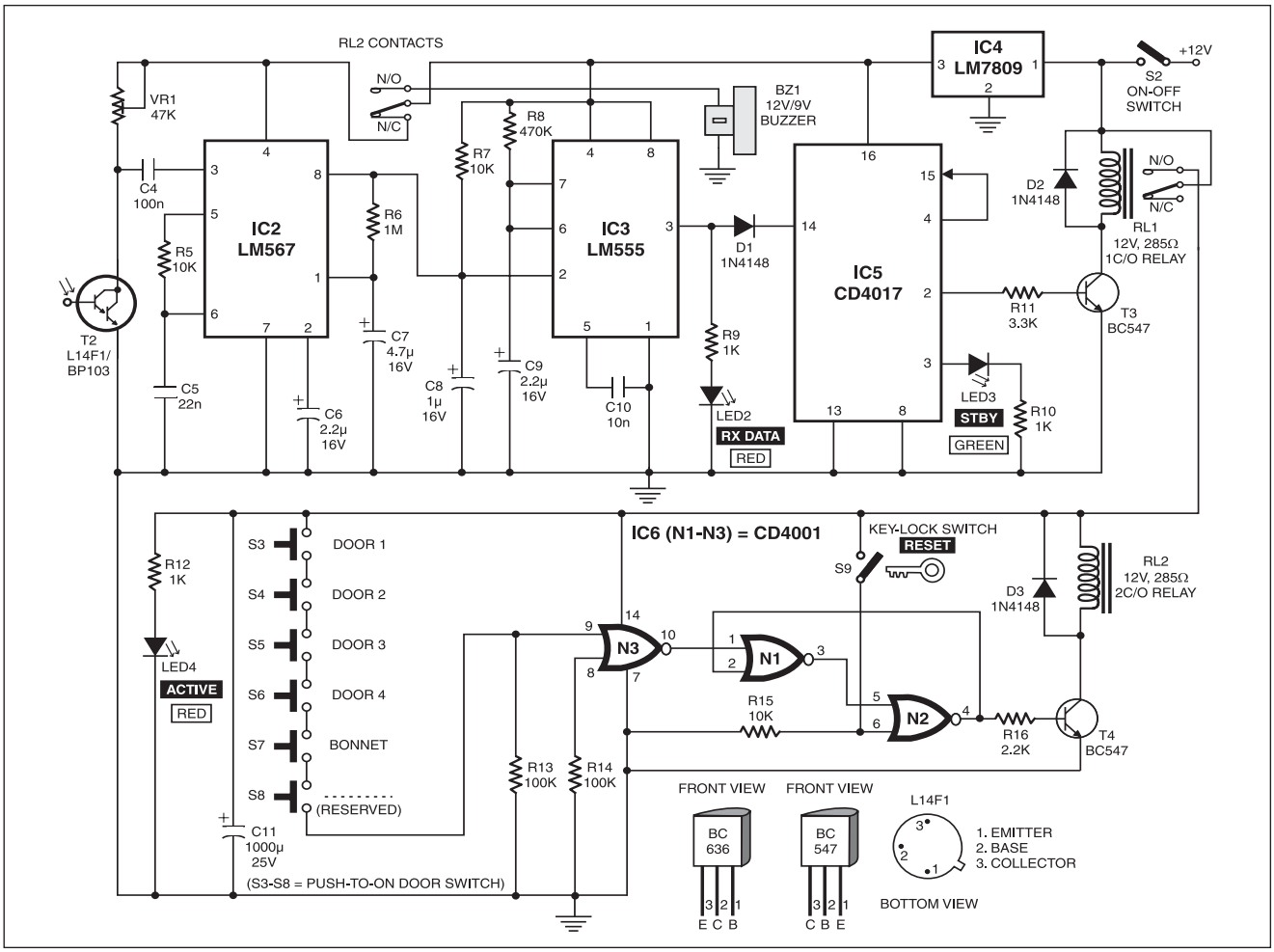


Fig. 2: Circuit diagram of the base unit

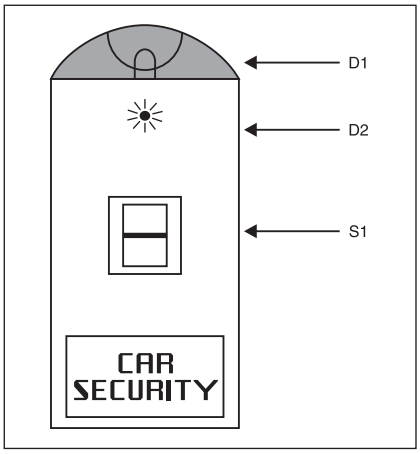


Fig. 3: Layout of remote control handset

Resistor R1 and capacitor C3 determine the output frequency at pin 5 of IC1.

For compactness, use a 9V PP3/6F22 battery for powering the remote control circuit. LED1 is used as a simple response indicator. (***EFY Lab note.*** Typical applications of LM567, courtesy National Semiconductor, are reproduced in Fig. 7 for the benefit of readers.)

The base unit

Fig. 2 shows the circuit diagram of the base unit. When switch S2 is turned on to connect 12V DC from the vehicle to relay RL1 and 9V regulator IC4, the regulated 9V DC is applied to the front end of the circuit. As a result, green LED (LED3) lights up to indicate the standby condition.

When photo transistor T2 receives a valid (4.5kHz) tone from the remote control handset, pin 8 of IC2 changes from high to low state to trigger the monostable (IC3). Consequently, red LED2 (labeled as RX Data) lights up briefly and the decade counter (IC5) is clocked to change its output state at pin 2 (Q1) to high. As a result, relay RL1 energises, while standby condition indicator LED3 goes off. This condition is maintained until the photodetector (IC2) receives the next IR burst from the remote control.

Thus after receiving the first burst from the remote control unit, the main power (+12V) is extended to the rest of the circuit (the bottom section in Fig. 2) via normally open (N/O) contacts of relay RL1. As a result, red LED4 lights up to indicate that the security controller is in active state.

Normally all the door switches, including the reserved one, are in closed state and hence the output of gate N3 of IC6 (quad two-input NOR gate) is in low logic state. Please ensure that any of these switches, if not in use, should be shorted using jumpers. However, when any of the

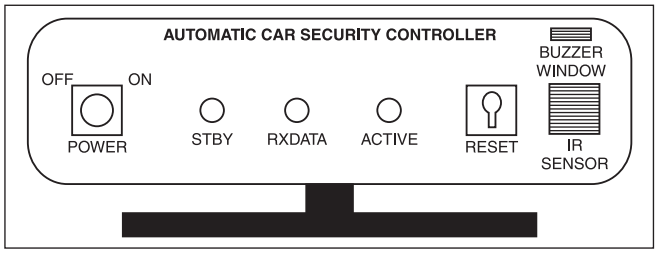


Fig. 4: Base unit enclosure with front-panel layout

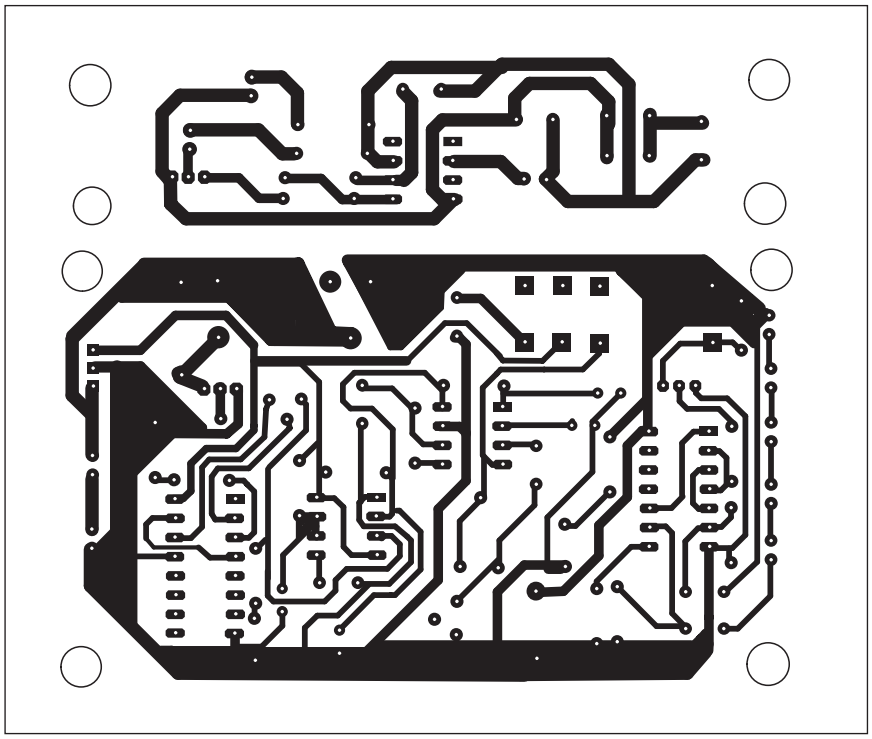


Fig. 5: Actual-size, single-side PCB layout for the car security system with remote control

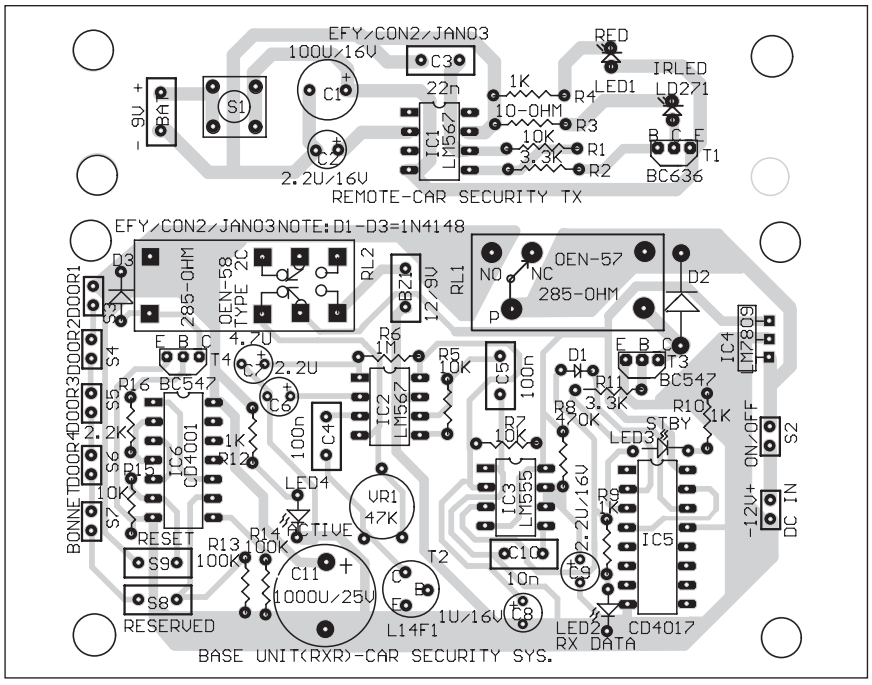


Fig. 6: Component layout for the PCB

PARTS LIST

Semiconductors:

IC1, IC2	-	LM567 tone decoder
IC3	-	LM555 timer
IC4	-	LM7809 9V regulator
IC5	-	CD4017 decade counter
IC6	-	CD4001 quad NOR gate
T1	-	BC636
T2	-	L14F1 Photo Darlington/ BP103 phototransistor
T3, T4	-	BC547 npn transistor
D1-D3	-	1N4148 switching diode
IRLED	-	LD271 infrared LED
LED1, LED2, LED4	-	Red LED
LED3	-	Green LED

Resistors (all ¼-watt, ±5% carbon, unless stated otherwise):

R1, R5, R7, R15	-	10 kilo-ohm
R2, R11	-	3.3-kilo-ohm
R3	-	10-ohm
R4, R9, R10, R12	-	1 kilo-ohm
R6	-	1-mega-ohm
R8	-	470-kilo-ohm
R13, R14	-	100-kilo-ohm
R16	-	2.2-kilo-ohm
VR1	-	47-kilo-ohm preset

Capacitors:

C1	-	100µF, 16V electrolytic
C2, C6, C9	-	2.2µF, 16V electrolytic
C3, C5	-	22nF ceramic disk
C4	-	100nF ceramic disk
C7	-	4.7µF, 16V electrolytic
C8	-	1µF, 16V electrolytic
C10	-	10nF ceramic disk
C11	-	1000µF, 25V electrolytic

Miscellaneous:

S1	-	Push-to-on tactile switch
S2	-	On/off toggle switch
S3-S8	-	Push-to-on door switch
S9	-	Key-lock type switch
*RL1	-	12V, 285-ohm 1C/O relay
*RL2	-	12V, 285-ohm 2C/O relay

** refer text*

switches (for example, door 1 switch S3) is opened, the output state of gate N3 changes from low to high to enable/set the bistable latch comprising gates N1 and N2. As a result, relay RL2 activates to sound the buzzer (BZ1). Relay RL2 being a two-changeover relay, the second pair of relay contacts can be suitably interfaced to any warning device such as an emergency beeper or a high-power signaling device or a wireless alert unit, as desired.

Push-to-on switch S9 is the reset switch for the bistable. Ideally, it should be a key-lock type switch, so any unauthorised person not possessing the key cannot turn the warning device off by resetting the bistable.

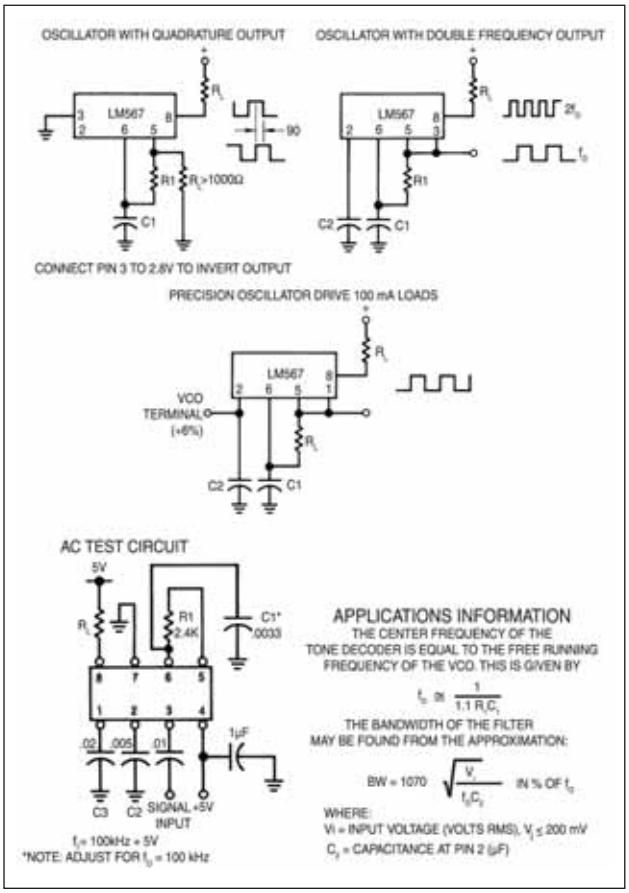


Fig. 7: Typical application circuits of IC LM567

The owner (holder of the remote control handset) can disable the security function at any time (but not after a burglary attempt) by using the remote control unit, to send another burst.

Figs 3 and 4 show the proposed layouts of the remote control and base unit, respectively. An actual-size, single-side PCB layout comprising both the remote control and base unit circuits is shown in Fig. 5 with its component layout in Fig. 6. The remote Tx part can be cut out for use in the remote. Relays RL1 and RL2 (OEN make, series 58, type 1C and 2C with coil voltage of 12V and 285-ohm resistance) can be directly mounted on the PCB.

The circuit can be remotely operated from a distance of up to 1 metre using the reflector and lense arrangement. However, to increase the range of remote control, you may:

1. Slightly decrease the value of 10-ohm resistor R3 to increase the operating current of IRLED LD271.

2. Replace phototransistor BP103 with a Darlington phototransistor (L14F1, L14F2, 2N5777, etc) as shown in Fig. 2.

3. Add a suitable transistor-based preamplifier between phototransistor T2 and tone decoder LM567 (IC2). ❑

Readers' comments:

I have the following doubts regarding the construction circuit 'Car Security System With Remote Control'.

Q1. The circuit uses L14F1 phototransistor as the IR beam sensor. Is there any effect of sunlight, car headlight, or streetlight on the functioning of the circuit?

Q2. Can I use an IR module in place of the phototransistor? If yes, what are the modifications required?

Q3. The circuit is powered directly from the car battery, which provides a high current (15-30 amps). There is no arrangement in the circuit to protect the delicate ICs, transistors, and 285-ohm relay from it.

D. Mohan Kumar
Palakkad, Kerala

The author T.K. Hareendran replies:

A1. The circuit uses modulated IR beam for control function, so light rays from headlights/streetlights will not affect the operation of the circuit.

A2. Readily available IR receiver modules for remote control applications are designed to work with the 30-40kHz range. In my prototype, the modulating frequency is only 4.5 kHz, so direct substitution is not possible. If you use such a module after proper modification in transmitter and receiver circuits, it will work. However, IR bursts from TV/video remote control handsets may affect the switching functions. This may harm the gadgets.

A3. You can directly connect the base unit to the car battery. Use of a current limiter/ constant-current generator is not very crucial.

A TUTORIAL ON 89C51 DEVELOPMENT KIT

**S. ANANTHI, K. PADMANABHAN
P. ARVIND, M. SHYAM, AND
M. SAKTHIVEL**

The 8051 family of microcontrollers introduced by Intel, and now being offered by a host of manufacturers such as Atmel, Philips, and Dallas, have proved to be very simple and useful for almost any application. The Atmel 89C51 and 89C52 happen to be the workhorses today. They contain internal flash EEPROM so that the program may be stored internally in the chip. Programmers for the device are easily available in the market. Some dealers too, who market these ICs, offer programming facility at a reasonably low price. So, if one develops code for his pet application (be it a coffee maker machine or a fan control or a digital capacitance meter), it is very simple to get the microcontroller programmed for use in the actual circuit board and get the thing going.

For simple programs using just a few input or output port bits, it is easy to write the code, fuse it into the IC, and try it in the circuit. If it does not work, the program can be erased and the modified program can be fused once again with the help of a programming tool for making a fresh trial. After several such attempts (provided the microcontroller remains intact), the circuit can be made ready in a proper working order. A good programming knowledge will reduce the trial time considerably.

Although several good books covering the software aspects of the 89C5X family of microcontrollers are available in the market for writing one's own application programs, there is no easy alternative to hands-on learning experience. A program development kit would be of immense help in the development of application programs.

The present project is much more than a simple kit. It provides an inexpensive platform for testing code and running the programs, including the use of interrupts and timers, which are available within the 89C51 microcontroller. Only after fully testing the program code and satisfying oneself that all the functions needed for the application are working without any ambiguity, one should go for programming that code inside the IC. This eliminates unnecessary trials and errors, especially

Working model of 89C51 microcontroller development board

PARTS LIST

For 89C51 development board

Semiconductors:

IC1	- 89C51 microcontroller
IC2	- 74LS373 8-bit latch
IC3	- 2764 EPROM (8k x 8-bit)
IC4	- 6264 RAM (8k x 8-bit)
IC5	- 28C64 EEPROM (8k x 8-bit)
IC6	- 74LS138 address decoder
IC7	- MAX232, RS-232 level converter
IC8	- 82C55 digital I/O interface
IC9	- 74LS240 octal inverting buffer/driver
IC10	- 74LS00 quad 2-input NAND
IC11	- 74LS04 hex inverter
LED1-LED8	- Red LED
LCD module	- 16 x 1-line type (HITACHI/ BEL Make)

Resistors (all ¼-watt, ±5% carbon, unless stated otherwise):

R1	- 4.7-kilo-ohm
R2	- 10-kilo-ohm
R3-R10	- 120-ohm
VR1	- 10-kilo-ohm preset

Capacitors:

C1, C2	- 33pF ceramic disk
C3-C10	- 100nF ceramic disk
C11-C14	- 22μF, 16V electrolytic
C15	- 1μF, 10V electrolytic

Miscellaneous:

S1	- Push-to-on tactile switch
Xtal	- 11.09MHz crystal
	- Two 40-pin IC bases
	- Three 28-pin IC bases
	- Two 20-pin IC bases
	- Two 16-pin IC bases
	- Two 14-pin IC bases
	- Berg stick connectors, pins, and a short jumper
	- IBM PC-AT keyboard connector (5-pin DIN female)
	- PC-AT keyboard
	- 5V, 1A regulated power supply

For capacitance meter application

IC1	- CD4081 CMOS quad AND gate
IC2	- CD4066B quad bilateral switch
R1	- 10-kilo-ohm
R2	- 100-kilo-ohm
R3	- 1-mega-ohm
	- Two 14-pin IC bases
	- General-purpose PCB

TABLE I

Special Functions of Port 3 Signals

P3.0	Pin 10	RXD (serial input port)
P3.1	Pin 11	TXD (serial output port)
P3.2	Pin 12	INT0 (external interrupt 0)
P3.3	Pin 13	INT1 (external interrupt 1)
P3.4	Pin 14	T0 (timer 0 external input)
P3.5	Pin 15	T1 (timer 1 external input)
P3.6	Pin 16	WR (external data memory write strobe)
P3.7	Pin 17	RD (external data memory read strobe)

when the program comprises several modules/parts, with multiple tasks involving timer interrupt and external interrupt routines and so on. In such cases it will be necessary to test each part of the program independently. The main features of the development kit include:

1. It employs 101-key AT-style keyboard for data entry and instructions, which is connected to the board using AT-style 5-pin DIN connector.

2. Provision is made for serial communication with the PC through COM1 or COM2 port using a 9-pin 'D' connector.

3. A 16-character x 1-line LCD module is used for alphanumeric display of data and address as well as text messages depending upon the application.

Description

The main board comprises the 89C51 microcontroller that contains the monitor program for this board to work. Fig. 1 shows the pin-out details of IC 89C51. The internal block diagram of IC 89C51, which is useful for frequent references to various ports, interrupts, etc, is given in Fig. 2. The circuit diagram of the main development board is shown in Fig. 3.

There is a provision in the PCB for 5-pin DIN type socket on this board to take the IBM PC AT keyboard, which is used for code entry. An extra 28-pin socket is provided on the board for plugging an EEPROM for continuous use while developing the programs is also feasible. An Atmel 28C64A or 28C64B can be used for this purpose. Since it would take several days for developing the proposed application program, this EEPROM will hold its data without a battery back-up.

For display, an alphanumeric 16x1 LCD module is provided, which can be used as an output unit in an application. For example, if your application needs to show some text and numbers, the program could be developed to display them on the LCD module.

In this board, the 89C51 (IC1) has been wired as a microprocessor to be optionally used with the external memory interface. However, since the program for this unit itself is internal to the 89C51 (preprogrammed into the device), pin 31 is to be held at high logic. This pin labelled EA (external access), if held at logic 0, accesses the external program memory.

On the PCB, a shorting jumper is provided so as to select either the internal or the external mode. Here it is held high in order to use the internal program memory for use with a preprogrammed 89C51 chip. In case an ordinary 8051 chip is fitted, it will use external EPROM monitor program and so the pin 31 jumper will be strapped to logic 0 and a preprogrammed 28C64 should be used in memory socket 1 (for IC3).

An 89C51 is provided with four ports: port 0, port 1, port 2, and port 3. This circuit uses an external memory in addition to the internal program memory available within the chip. So, the address and data bus will be required to access the external memory as well. For this, port 0 provides address bits A0-A7 and data bits D0-D7 combined as time-multiplexed signals with the ALE (address latch-enable) signal pulse coming out from

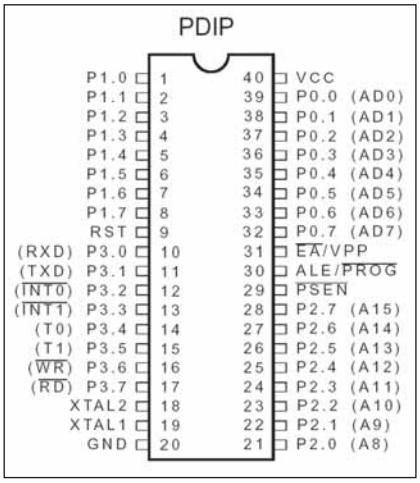


Fig. 1: Pin configuration of IC 89C51

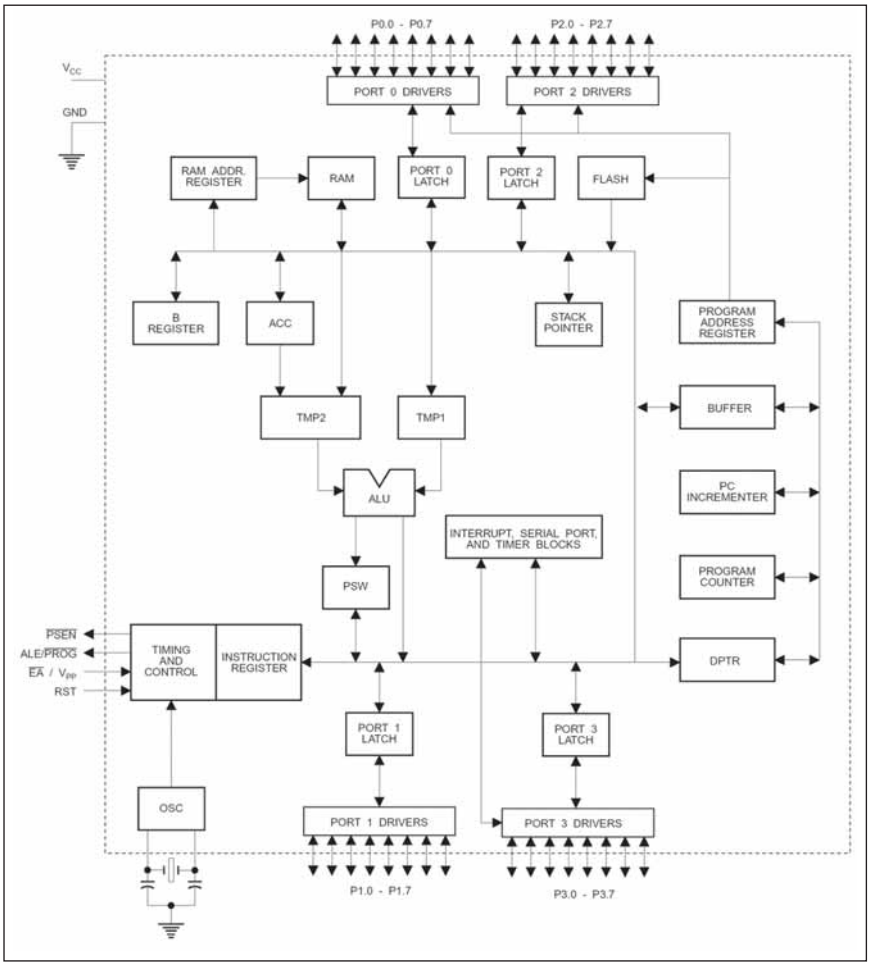


Fig. 2: Internal functional block diagram of IC 89C51

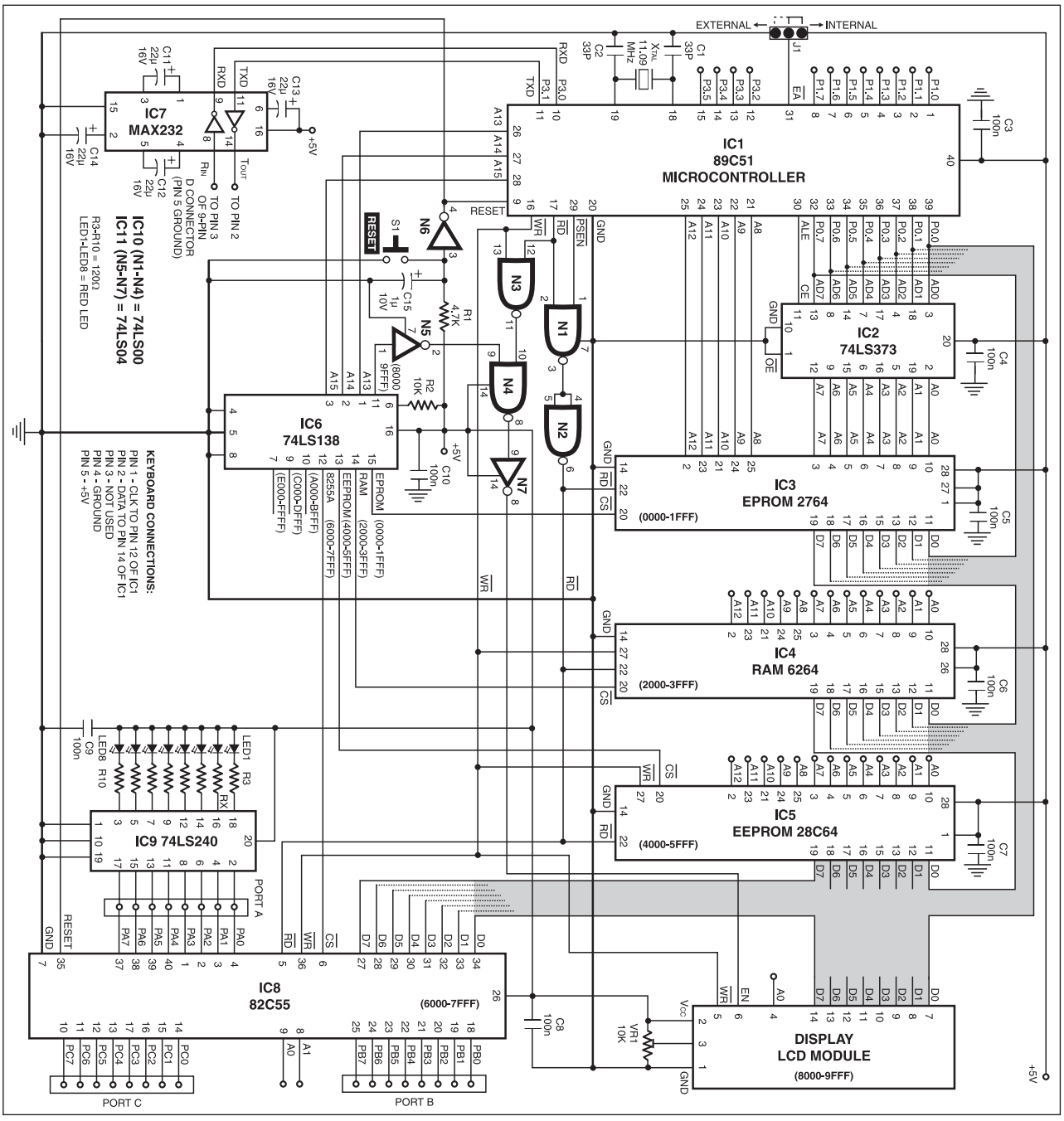


Fig. 3: Circuit diagram of main 89C51 development board

pin 30 of the IC. Using this ALE signal, the low-order 8-bit address (A0-A7) from port 0 is latched into an external 8-bit latch 74LS373 (IC2). Higher-order address lines (A8 to A15) are obtained from port 2 (pins 21 to 28 of IC1). The 8-bit port 1 (pins 1 to 8 of IC1) is available as a separate input/output port.

In this circuit, because of usage of external memory, all ports except port 1, i.e. ports 0, 2, and 3, are used for generation of the requisite address and data signals. Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. It also serves various special functions (shown in Table I).

For accessing the external memory in groups of 8k, a 3-line-to-8-line address decoder 74LS138 (IC6) is employed. Address lines A13, A14, and A15 from IC 89C51 are given to its input pins 1, 2, and 3, respectively. The decoded outputs from pin 15 down to 7 provide the chip address-select signals as given in the table along side.

IC pin No.	Address select range	Comments
Pin 15	0000 to 1FFF	Selects socket 1 (IC3) for monitor EPROM
Pin 14	2000 to 3FFF	Selects second socket (IC4) i.e. RAM 6264
Pin 13	4000 to 5FFF	Selects third socket (IC5) for EEPROM 28C64
Pin 12	6000 to 7FFF	Selects 82C55 (IC8)
Pin 11	8000 to 9FFF	Selects LCD module
Pin 10	A000 to BFFF	Not used
Pin 9	C000 to DFFF	Not used
Pin 7	E000 to FFFF	Not used

Memory signals. Address lines A0 to A12 are connected to the three sockets meant for memory (EPROM/RAM/EEPROM). Pin 22 of these sockets are connected to the combination of PSEN and read signals via NAND gates N1 and N2. The PSEN signal is issued out of the 89C51 whenever the program memory external to the IC is selected. Thus, while executing the program code, if the address of the program happens to be the address noted above, the PSEN signal is to be used as the read signal for reading that memory. Also, the read memory signal meant for external memory read operation appearing at pin 17 of the 89C51 is to be used for reading the external data memory. In this circuit, both program as well as data memory are combined into one and the same IC. So, $\overline{\text{PSEN}}$ and $\overline{\text{RD}}$ signals from IC1 are AND-ed in a 74LS00 gate, inverted, and used as $\overline{\text{RD}}$ signal for pin 22 of all the three sockets for the memory.

The LCD module. A simple 16-character display of a single row is connected to the circuit to show address and data at any memory location. Then, programs can be entered from the keyboard, viewed on the display, and then executed from any selected address location. This will be clearer as we proceed further.

The LCD requires an interface to its internal microcontroller (part of LCD module) through an 8-bit data bus. There are a set of registers internal to the LCD module and these are selected through address line A0. The chip-enable signal connected to pin 6 of the module is an active-high signal. The WRITE signal from pin 16 of the 89C51 is connected to it to enable write operations. Internally, the LCD appears to the 89C51 as a set of register addresses, as under:

LCD Module Operation

Address select	Read	Write
Base+0	Status reg.	Command reg.
Base+1	Display data reg.	Display data reg.

Thus, if one writes into the command register, the data on the data bus is written into the internal command register, which is required for setting up the mode of operation of the LCD, address setting, clearing the display, etc.

If one writes into the data register, the actual character is written in, which causes that ASCII character to be shown on the LCD screen at the current cursor position.

The selection of cursor and its movement (auto increment) are controlled by the command code suitably written into the command register beforehand.

Instructions for programming the LCD modules have already appeared in Electronic Projects Vol-18, Electronic Projects Vol-19 and Electronic Projects Vol-23.

As far as the interface connections are concerned, the data bus goes to pins 7 through 14 of the LCD. Address line A0 goes to pin 4, the write signal goes to pin 5, and the device-select signal goes to pin 6.

In order to properly operate the LCD, the device-select signal is generated from a combination of address-select signal (8000-9FFF), i.e. pin 11 of IC 74LS138 and read and write signals. The required gating is obtained using ICs 74LS00 (gates N3 and N4) and 74LS04 (gates N5 and N7).

There are 14 pins on an LCD module with no backplane yellow light. The LCD module with backplane lighting uses two extra pins for the supply for the backplane LEDs. A Berg connector is useful for fixing the LCD module directly on the board or through a flexible wire connector to a panel.

An 8255 interface. Since ports 0, 2, and 3 are not available on this board for I/O purpose, the 8255 digital interface is added to the unit to support three additional ports for program development. After development of the program, when a dedicated 89C51 board is used for the application, ports 0, 2, and 3 will become available on the chip.

Some small extra code is needed while using the external 8255 programmable interface IC in developing the code. For instance, if an industrial safety control is to be rigged up, using input switches from proximity sensors and the like, the need for extra inputs while testing the program under development is met by the 8255. If extra outputs for relays and lights are required, these are also available via other port of the 8255. Ports A, B, and C of this IC can be set up individually as inputs and outputs, but not for bit-wise operation. Where bit-wise inputs (one bit as input and the next bit as output) are needed, pins 1 to 8 of port 1 of IC1 are to be used by the application program, which are already there on the 89C51.

An 8255 or 82C55 (CMOS version of the IC) can be used. Address lines A0 and A1 given to this IC enable the selection of its four internal registers. Pin 12 of IC 74LS138, which selects 6000H address group, is connected to chip-select pin 6 of IC 82C55. Therefore the four internal registers are at address values 6000 through 6003 Hex. Read and Write signals from the 89C51 microcontroller are connected to read- and write-select pins 5 and 36 of IC 82C55. The data bus is connected to data input pins 34 down to

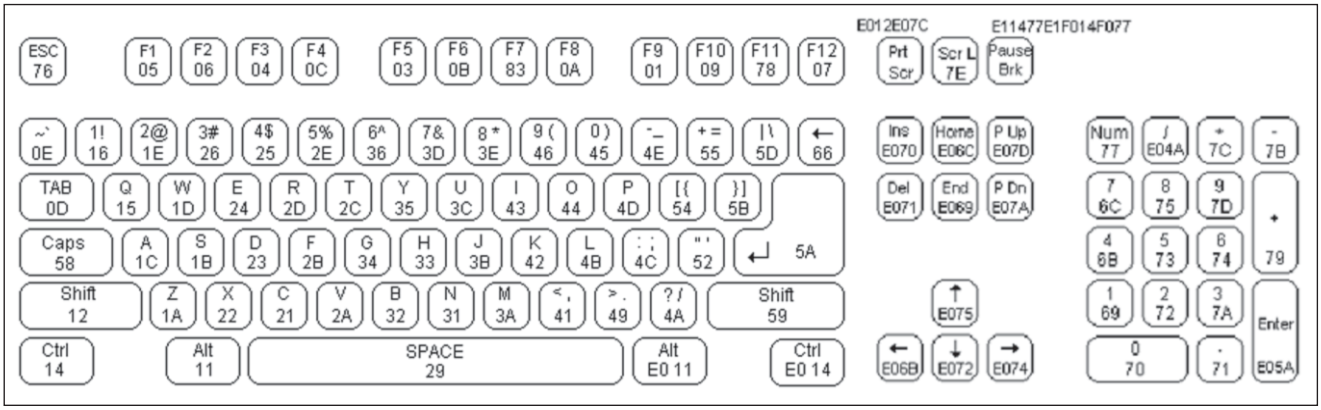


Fig. 4: AT-type IBM PC keyboard showing scan codes for various keys

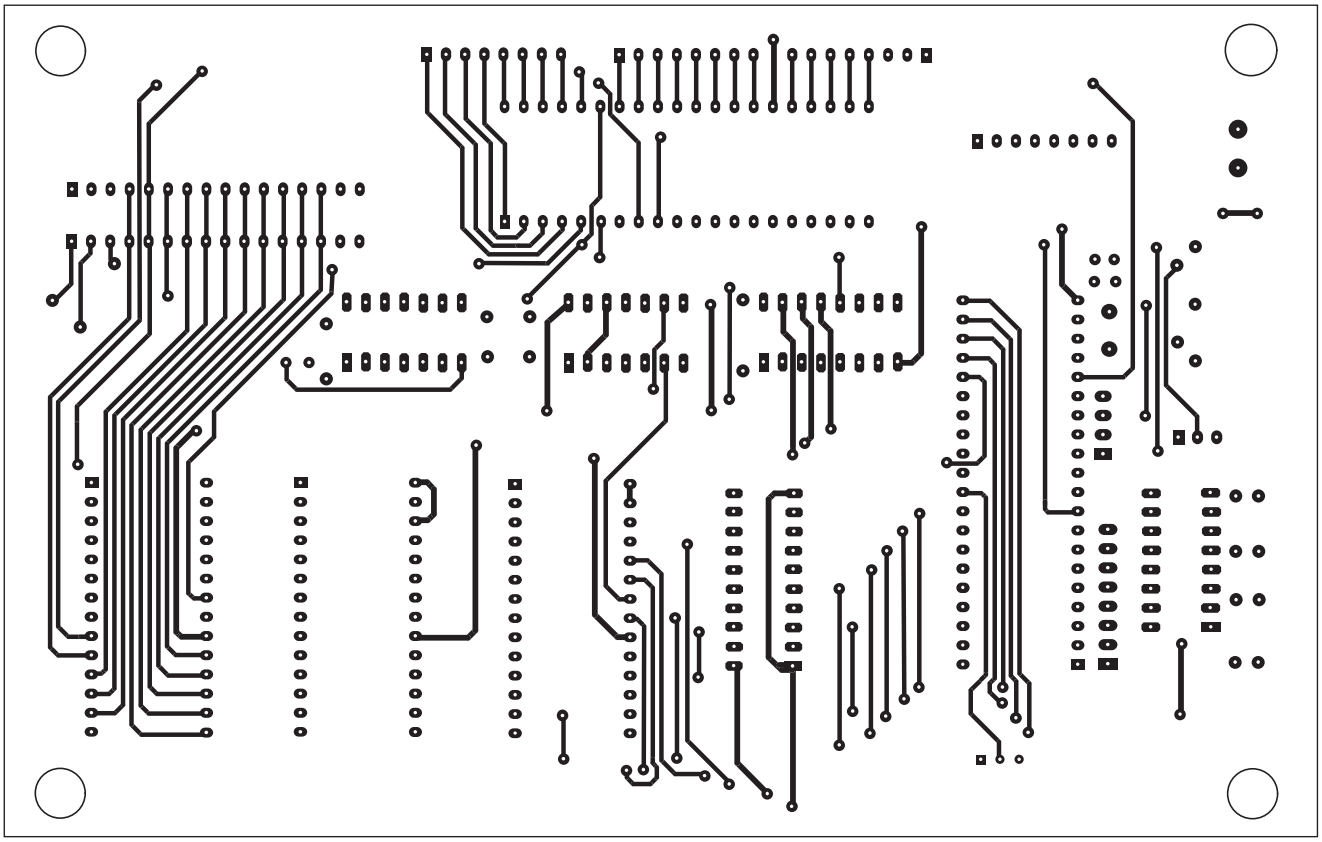


Fig. 5: Actual-size, component-side PCB layout for the 89C51 development board

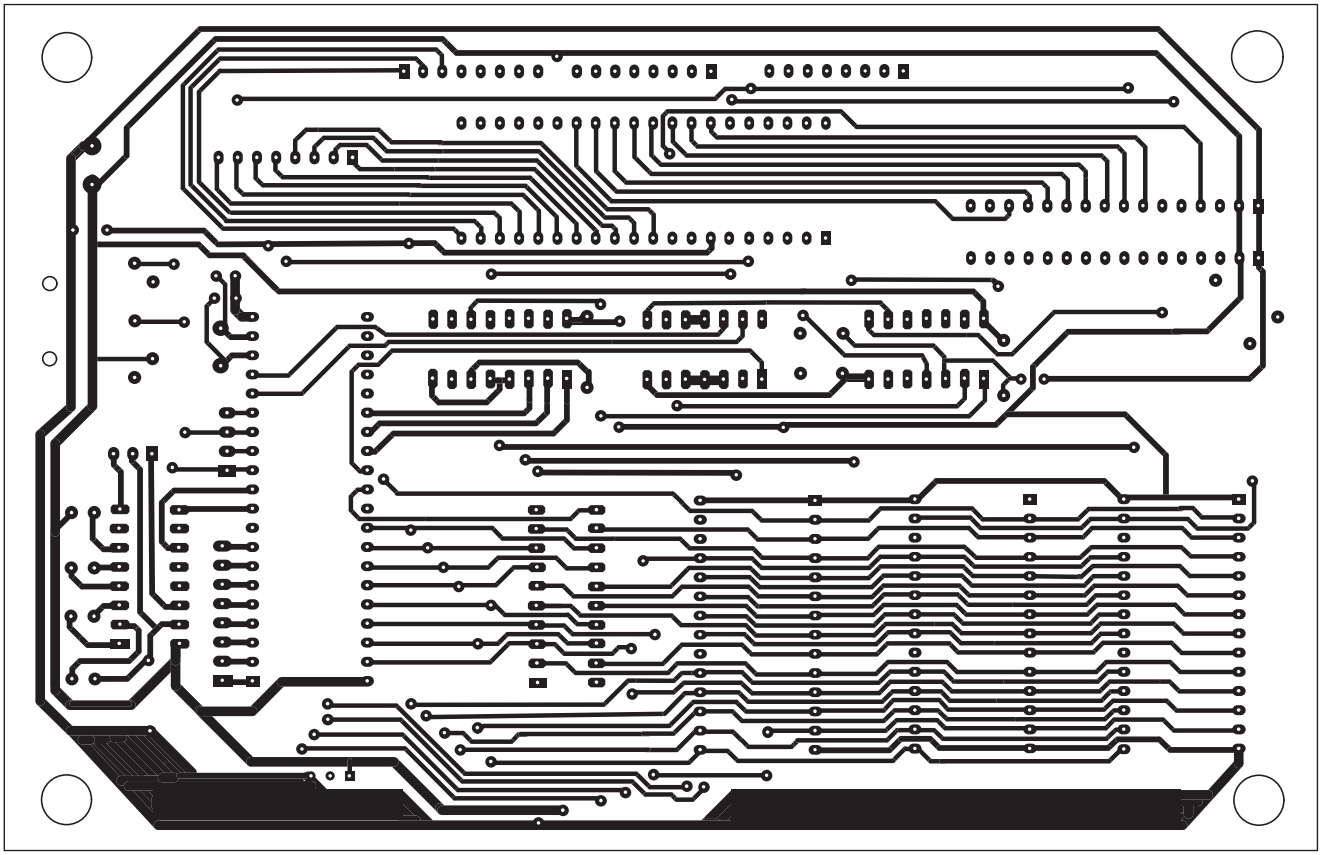


Fig. 6: Actual-size, solder-side PCB layout for the 89C51 development board

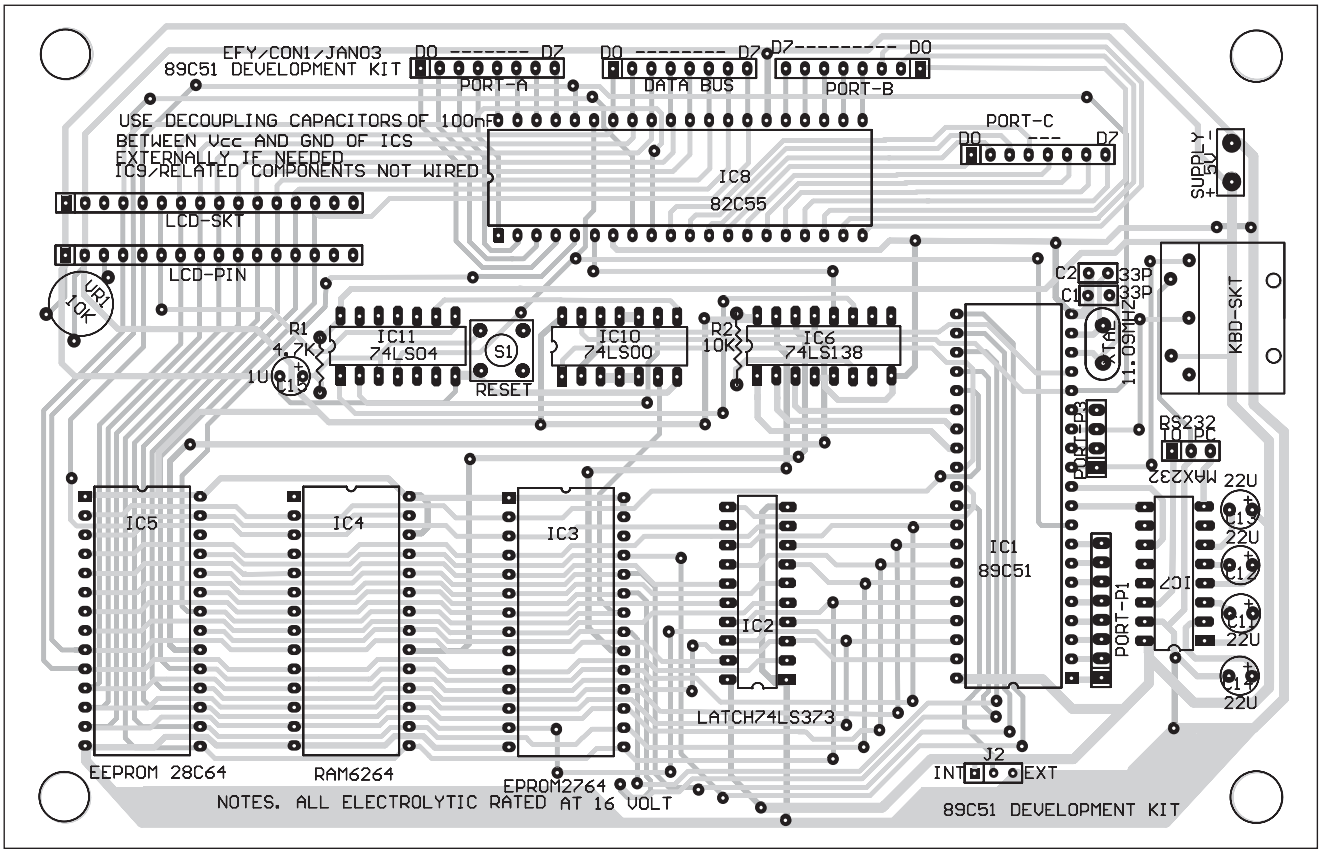


Fig. 7: Component layout for the PCB

27 (D0 through D7).

Ports A, B, and C are configurable as either input or output. The port pins are brought out to connectors, from which one can hook up a buffer IC such as 74LS240 and wire up the LEDs for any indication. 8-way DIP switches can be used for grounding the input port to input any number from 0 to 255 or for connecting bit-wise inputs as in any limit switch or proximity switch application.

Keyboard interface. For easy program code entry, an IBM PC AT keyboard is interfaced to the circuit. The PC keyboard has two pulse inputs for clock and serial data, which are given to pins 12 and 14, respectively, of the 8051. Data and clock signals are used in a software program to read the scan code given out when any key is pressed. Upon pressing a key, the scan code of that key outputs serially via the data pin, while the clock pin sends the timings of data bits. So, each bit has to be read during the high-clock period. As the bits stream out of the keyboard connector, pin 12 of IC1 receives clock signal, while pin 14 of IC1 receives data signal.

The IBM PC scan codes are converted to the corresponding key codes in ASCII value. For example, the scan code for letter A is 1C hex, and this is converted to ASCII 41 hex. Scan codes for the keys are shown in Fig. 4.

The software for the keyboard in this application takes into consideration the keys used for entry of hexadecimal numbers 0 through 9 and A through F as also the keys used for high address selection, low address selection, incrementing and storing, incrementing, decrementing, register viewing, and program execution. The detailed usage of these keys is as follows:

- S key is used for incrementing an address.
- Enter key is used for storing the current data at the current address and later incrementing the address.
- Backspace key is used for decrementing an address.
- G key is used for executing a program from the current displayed address.
- H key is for making the data field value as the high address.
- L key is for making the data field value as the low address.
- R key shows contents of the internal register of the 89C51.
- T key enters data into the internal register and increments the address of the internal register, and shows data there.

For entering the commands, the AT-type PC keyboard is interfaced in a very simple way to the 89C51. Pins 12 (P3.2) and 14 (P3.4) are used as clock and data input pins, respectively. (The same pins are also used as interrupt 0 and timer 0 input pins.) These are the only two connections for the keyboard other than +Vcc (5V) and ground.

The software interprets the key pressed and takes appropriate action for the keys used in this kit. When the kit starts working upon pressing reset button, the display shows the first RAM socket address and the contents of the memory there. For example, the display shows:

20 00 44

This is the first address of the RAM socket where one would begin to enter the program codes, at 2000H. The 44 at the end is data presently stored at this address location.

Enter the new data, say, 90, by pressing 9 followed by 0 on the keyboard and observe the display. The display shows:

20 00 49 ... after pressing key 9 on

TABLE II
AT Keyboard Connector Pin Assignments

AT Computer		
Signals	**DIN41524, female at Computer, 5-pin DIN 180°**	**6-pin mini DIN PS2 style female at Computer**
Clock	1	5
nc	3	2, 6
+5V	5	4

the keyboard

20 00 90 ... after pressing key 0 on the keyboard

If you want to access the EEPROM, the high address is to be set to 40. Enter 40 in the data field and transfer it to the high address field by pressing H key. Thus 20 00 40 on the display becomes:

40 00 FF

where FF indicates the blank data of a fresh EEPROM IC fixed into the socket. Then, when a low address of, say, 20, is to be set, the number is first entered in the data field as:

40 00 20

followed by depression of L key, which shows the display as:

40 20 FF

selecting the memory address as 4020H.

On entering any data, say, 74H, the display will show:

40 20 74

which can be stored by pressing Enter key of the keyboard.

Thus 74H gets stored at 4020H. The address is automatically incremented to show:

40 21 FF

So, one can keep on entering data one byte after another and the same gets stored.

When an EEPROM IC is fixed in any of the two sockets (RAM or EPROM) at addresses 2000 to 3FFFH and 4000H to 5FFFH, the data can be kept stored even after the power is off, because the EEPROM can be written in this kit *in situ* without using any external programmer.

If we want to view the EEPROM contents one by one after having written a set of bytes into locations, say, 4020H to 40FFH, it is necessary to set the address to 4020H to start with. The display now shows:

40 20 74

Then, instead of using Enter key to increment the address, use S key, which just shows the contents of any address and doesn't store data from data field into the current address. So to just check up/verify the previously entered data from address 4020 onwards, press S key. The display shows:

40 21 80
40 22 90
40 23 60
40 24 03

one by one data at these memory locations. This is the function of S key.

R key is the register entry key, which moves the data field into the middle field, and that becomes the register address. So, any internal register can be viewed. Data can also be entered directly into any register address by using T key, which moves the data in the data field into that register and then increments the register address as shown below:

20 40 30

Now R key is pressed to get:

20 30 AE ...showing that register 30 holds AE hex.

Enter 5 and 5 to get:

20 30 55

Then on pressing T:

20 31 xx

shows that 55 has been entered into register address 30 and then the current data xx or whatever is present in register address 31hex appears.

In these operations of R and T keys, the first field (20) has no significance.

Backspace key can be used for decrementing addresses when the external memory is used for program entry into the RAM or EEPROM socket. It does not work with register entry.

The number keypad of the PC's keyboard is also used for number entry.

Monitor program

The monitor program (refer Appendix A) is simple and fully commented. It uses the program from its location 0200H onwards to decipher the scan codes and convert them to codes 0 to 9, A to F, as well as the keys for control action (such as Enter key for storing and incrementing, R key for register entry, and so on). Thus any one wanting to use other keys can include additional key codes. Key depressions other than those used for this program are treated as null, which returns just FFhex from the routine 'convert'.

RS-232 communication interface

This is necessary if one wants to interface the board to a simple computer through a communication port, say, COM2. The computer is used for program entry in Assembly language (of 89C51) and converting the same into the codes that have to be actually entered into the RAM or EEPROM on the board. Some vendors supply software for downloading the program into the RAM from the PC and executing from the PC itself. (This is considered superfluous by most users because the learner always questions what is the use of the 89C51 board when the computer itself can do much better.) The W key is used for downloading the program code from the PC. We shall explain it in detail later.

As far as possible, it is advisable to develop programs on the board using direct code entry. If the program is lengthy, assemble the program using a cross-assembler on a PC. After getting the code from there, transfer the same into the board's memory through the RS-232 port. From then onwards, the program can be executed using the G key from the keyboard (of the kit) after setting the start address of that program.

Actual-size, component-side and solder-side PTH PCB layouts for the 89C51 development kit circuit are shown in Figs 5 and 6, respectively. The component layout (silk screen) for the PCB is shown in Fig. 7.

There is a two-way jumper provision, which is to be selected by the user. If the jumper shorts the middle and left pins on the PCB, the 89C51 internal program is selected as the program memory. In that case, the user inserts a preprogrammed 89C51. If the middle and right pins are shorted by the jumper, the 8051 mode with the external program memory access is selected. In that case, the user inserts in the first socket (meant for IC3) a pre-

programmed EPROM or EEROM containing the monitor program (refer Appendix A).

The keyboard connector is at the middle-right of the PCB. The IBM PC AT keyboard, which is available for around Rs 200 only, is to be connected to this socket. The keyboard socket provides 5V power for the keyboard to function. Clock and Data signals received from the keyboard are extended to pins 12 and 14 of the microcontroller. Refer Table II for keyboard connector pin assignments.

The LCD is to be connected via the 16-pin connector on the top left of the PCB. We have provided for two types of connectors, one for pins on the LCD to mate with a 16-pin connector with 2.54mm pitch socket, and the other a Relimate connector pin housing or Berg strip from which a wired connection can be made to an LCD module that can be mounted outside the PCB (say, in a box or cabinet). If the first socket is used, a pin-soldered LCD module can snugly fit onto the socket and the unit will be compact.

A contrast control potmeter or preset is connected via the 3-pin connector on the middle left of the board. Its contrast is adjusted such that the LCD characters are properly visible in the presence of ambient light. Electroluminescent backlit LCD modules are also available in the market. These have 16 pins, with pins 15 and 16 meant for the 5V needed for the backlighting of internal LEDs.

A Reset pushswitch (S1) is provided on the board. A suitable 7805 IC regulated 5V power supply is needed for the unit. A three-pin connector on the top right of the board is meant for the RS-232 computer cable connection. The connector pins are TXD, RXD, and Ground, which have to be connected to the COM port of the PC. Pin TXD will go to pin RXD of the COM port and vice versa. Thus pins 2, 3, and 5 of the COM port are to be connected by a three-wire cable. A modem cable can be used as well.

Checking for board faults

Using a continuity tester, such as the one published in Electronic Projects Vol-17, it is possible to check all connections to data and address lines as per the circuit diagram. The +5V and ground connections to all ICs must be checked. After connecting the 5V supply, first observe for crystal oscillations at pin 18 of the 89C51 or 8051, as the case may be. Then on pressing Reset switch S1, pin 9 of the microcontroller should go high. A logic probe can be used if a CRO is not available.

After a reset, the address and data lines must be pulsing. This can be checked with the help of the probe. The LCD would show the address and data in the RAM socket. If there is no RAM fixed on the second 28-pin socket, it will show FF:

20 00 FF

The display may show a random value if a 6264 RAM is fixed on the second socket. It is advisable to use the third socket for fixing an EEPROM (IC 28C64). This will be written by any data entry to locations 4000 to 5FFF hex.

Thus, since the 28C64 will hold data even after switch off, program development can be done using this IC, and the address of the application program could be 4000H onwards.

APPENDIX 'A' : MONITOR PROGRAM LISTING

```
2500 A.D. 8051 CROSS ASSEMBLER   -   VERSION 3.41f

INPUT  FILENAME :  KTTT1.ASM
OUTPUT FILENAME :  KTTT1.OBJ

Line Addr  Code label      Mnemonics                    Comments
1
2    0000                  .ORG 0
3    0000  01 25           RESET:  AJMP MONI
4    0003                  .ORG 0003     ;upon ext.interrupt on pin 12 jump to 2F03
H                                                       address
5    0003  02 2F 03        JMP 2F03H  ; EXTERNAL INT. VECTOR 0
6
7    000B                  .ORG 000BH  ;when internal Timer 0 overflows, jum to
                                                        2F2B H address
8    000B  02 2F 2B        JMP 2F2BH  ; TO TIMER/COUNTER INTERRUPT  '0'
9    000F                  .ORG 000FH
10   000F  02 2F 43        JMP 2F43H
11
12   0013                  .ORG 0013H   ;upon ext.interrupt on pin 13 jump to 2F63
H                                                       address
13   0013  02 2F 63        JMP 2F63H   ;EXT. INT. 1 ADDRESS
14
15   001B                  .ORG 001BH  ;upon timer 1 overflow, jump to 2F8B H
                                                        address
16   001B  02 2F 8B        JMP 2F8BH    ; EXT. TIMER COUNTER 1 INT. VEC.
17
18   0023                  .ORG 0023H  ;upon serial interrupt  jump to 0023 H
                           address
19
20   0023  01 FC           AJMP 00FCH    ;SERIAL PORT INTERRUPT VECTOR
21   0025                  .ORG 25H
22   0025                  MONI:
23   0025  75 B0 FF        ST:     MOV P3,#FFH  ;set port 3 for secondary function
                                                        use.
24
25   0028  75 81 60        MOV SP,#60H   ;set stack pointer to 60H in internal RAM
26   002B  11 DE           ACALL INIT_LCD  ;initialise the LCD display
27   002D  90 20 00        MOV DPTR,#2000H ; to point to first RAM memory loca
                                                        tion and
28   0030  E0              SC2:      MOVX A,@DPTR    ; get the data from that
                                                        address
     into A
29   0031  FA              SC3:      MOV R2,A       ; put it in R2 register
30   0032  31 30           SC1:      ACALL DISPLAY    ;DISPLAYS DPH,DPL, R2
31   0034  11C0            ACALL KBD   ; await and get a key pressing from user
32   0036  EF              MOV A,R7    ;get the keycode from reg.7 where it got
                                                        stored
33   0037  12 02 00        CALL CONVERT      ;TO ACTUAL KEY CODES 0 to F
and                                                     control keys
34                                                      ;40 to 4c
35   003A  C3              CLR C     ;clearing the carry flag since a subtraction
                                                        follows
36   003B  FB              MOV R3,A ; save keycode in register 3
37   003C  94 40           SUBB A,#40H ; is it command code (is it greater than 40?)
38   003E  50 07           JNC D  ;if so,jump to point D to check for various
                                                        commands
39   0040  EA              MOV A,R2   ; so we got a key 0 to F.  Shift it to left
40   0041  C4              SWAP A
41   0042  54 F0           ANL A,#F0H  ; keep the upper nibble which now has the
                                                        key value
42   0044  4B              ORL A,R3   ; combine with the value in display kept in R3
43   0045   01 31          JMP SC3    ; jump to SC3, where it is moved to R2 for
                                                        showing on LCD
44   0047  EB              D:      MOV A,R3   ; here we check which command it is
45   0048  B4 41 04        CJNE A,#41H,E ; is it 41 which is set for high addr.
                                                        setting
                                                        key
46   004B  8A 83           MOV DPH,R2  ; if so, move the value of R2 to the high ad.
                                                        ptr.
47   004D  01 30           JMP SC2      ; go to look for key again
48   004F  EB              E:      MOV A,R3
49   0050  B4 42 04        CJNE A,#42H,F  ; compare with 42 which is set for low
                                                        addr. set key
50   0053  8A 82           MOV DPL,R2    ; if so, move R2 to data pointer low
51   0055  01 30           AJMP SC2
52   0057  EB              F:      MOV A,R3
53   0058  B4 44 08        CJNE A,#44H,G    ;44 is decrement address key
54   005B  15 82           DEC DPL         ; so decrement data pointer
55   005D  70 02           JNZ Q
56   005F  15 83           DEC DPH
57   0061  01 30           Q:      AJMP SC2      ;then, go to show the same at SC2
                                                        and read kbd further
58   0063  EB              G:  MOV A,R3
59   0064  B4 47 05        CJNE A,#47H,H  ;47 control code is  assigned for store the
                                                        data+increment
```

```
0067  EA        MOV A,R2      ;get the data field byte in A
0068  F0        MOVX @DPTR,A  ;write that into the current address
0069  A3        INC DPTR      ; move to next address
006A  01 30     AJMP SC2      ;go and show the next address and
                              data,await kbd.
006C  EB        H:  MOV A,R3
006D  B4 43 05  CJNE A,#43H,K ;43 control code is "Go" key to run
                              program at current addr.
0070  E4        CLR A
0071  75 D0 08  MOV PSW,#8    ; set program status word for user
                                    register bank
0074  73        JMP @A+DPTR   ; go to execute program at currently
                              shown address
0075  B4 48 03  K:  CJNE A,#48H,J   ;here we check if the code is S
                              key, just increment
0078  A3        INC DPTR      ; increment address
0079  01 30     AJMP SC2      ;go and show the next address and
                              data,await kbd.
007B  B4 4B 04  J:  CJNE A,#4BH,J1  ;small enter key for code seeing at
                              address
007E  E4        CLR A         ; clear A for next instrn.
007F  93        MOVC A,@A+DPTR ; get code into A from current
                                    address
0080  01 31     AJMP SC3      ;Show it and go to await keybd.
0082  21 00     J1: AJMP 100H
0090            .ORG 090H     ;routine to enter a command into LCD
0090  C0 83     CMD:    PUSH DPH      ;save values of Data pointer
                                    since we disturb it inside
0092  C0 82     PUSH DPL
0094  90 80 00  MOV DPTR,#8000H ;set data pointer to LCD display
                                    address
0097  F0        MOVX @DPTR,A  ;write accumulator value into LCD
                                    command register
0098  11 F7     ACALL DELAY   ;wait for LCD to act
009A  D0 82     POP DPL
009C  D0 83     POP DPH       ;  retrieve earlier data pointer value
009E  22        RET           ; return


009F  C0 83     LC_wr:  PUSH DPH  ;this routine writes into the LCD
                                    a data value
00A1  C0 82     PUSH DPL      ; save data pointer on stack
00A3  C0 E0     PUSH A        ;push to stack the byte to be written to LCD
00A5  90 80 00  MOV DPTR,#8000H  ;point to <LCD>
00A8  E0        KC:     MOVX A,@DPTR
00A9  20 E7 FC  JB ACC.7,KC   ;busy is checked on LCD status
00AC  D0 E0     POP A         ;get the byte
00AE  90 80 01  MOV DPTR,#8001H ; 8001 is the data register of the
                                    LCD
00B1  F0        MOVX @DPTR,A  ; write into that register
00B2  11 F7     ACALL DELAY   ; wait for LCD to act
00B4  D0 82     POP DPL       ; retrieve data pointer back to earlier value
00B6  D0 83     POP DPH
00B8  22        RET           ; return

00C0            .ORG C0H      ; this routine reads a code sent by IBM PC
                              AT
                              keyboard
00C0  7B 08     KBD:    MOV R3,#8     ; 8 is count of bits per code
00C2  7F 00     MOV R7,#0     ; R7 picks the serially received bits from
kbd
00C4  A2 B2     KP1:    MOV C,P3.2    ; check the clock bit
00C6  40 FC     JC KP1
00C8  A2 B2     K4:  MOV C,P3.2   ; high to low got
00CA  50 FC     JNC K4        ;
00CC  A2 B2     K5:     MOV C,P3.2    ; low to high got, so start bit is over
00CE  40 FC     JC K5
00D0  A2 B4     MOV C,P3.4    ; read data bit
00D2  EF        MOV A,R7      ; pack into R7
00D3  13        RRC A         ; using shifting through carry flag
00D4  FF        MOV R7,A
00D5  A2 B2     K6:     MOV C,P3.2    ; look for next clock bit
00D7  50 FC     JNC K6
00D9  DB F1     DJNZ R3,K5    ; like this for 8 clock bits
00DB  11 F7     ACALL DELAY
00DD  22        RET           ; return  with code in A and R7


                INIT_LCD:     ;this routine initialises the LCD module for
                              cursor mode
                              ; and clears the display, no shift of display
00DE  74 38     MOV A,#38H
00E0  11 90     ACALL CMD     ; set the function code 38for  our LCDs
00E2  74 0E     MOV A,#0EH
00E4  11 90     ACALL CMD     ; cursor set etc.
00E6  74 06     MOV A,#06H
00E8  11 90     ACALL CMD     ; shift mode
00EA  74 80     MOV A,#80H
00EC  11 90     ACALL CMD     ; address of display points to leftmost
                              character slot
00EE  74 01     MOV A,#1
00F0  11 90     ACALL CMD  ; clear display now
00F2  22        RET


00F7            .ORG 0F7H     ;a delay is required for LCD after each
                              instruction

00F7  7D 80     DELAY:  MOV R5,#80H   ; .1 ms
00F9  DD FE     DJNZ R5,$
00FB  22        RET
00FC            .ORG 0FCH
00FC  02 2E 00  SERINT:  JMP 2E00H  ; this goes to RAM address when
                                    a serial port interrupt
                ; happens, provided the Serialinterrupt is enabled.
                ; used in programs for user to employ int. mode serial
                  transfer.
                ; serial port interrupt is not used by (this) monitor pgm,
0100            .ORG 100H
                ; the follg. two keys are used fore internal register access.
                ; so, you can read 89c51's internal registers and write to
                  them directly
0100  B4 49 08  CJNE A,#49H,L1      ; register key is "R"
0103  EA        MOV A,R2
0104  F9        MOV R1,A              ; used to set the register address
                                    in display
0105  E7        MOV A,@R1
0106  FA        MOV R2,A
0107  89 82     MOV DPL,R1
0109  01 32     AJMP SC1
010B  B4 4A 0B  L1:  CJNE A,#4AH,L2   ;store in internal register -
                                    control
                                    code 4a, key T
010E  EA        MOV A,R2
010F  A9 82     MOV R1,DPL   ; pressing T key stores data field into
                              register
0111  F7        MOV @R1,A  ; writes data into register
0112  05 82     INC DPL
0114  09        INC R1        ;increment the register address
0115  E7        MOV A,@R1
0116  FA        MOV R2,A
0117  01 31     AJMP SC3   ; go back to display and read keyboard
                              further
                ; the follg. is the Download program code from PC key "W"
0119  B4 4C 02  L2:  CJNE A,#4CH,L3   ; is it W key
011C  41 E1     AJMP SERIN        ; go to read serial data

011E  01 31     L3:  AJMP SC3         ; if the key pressed does not match
                                    with any of the above
                      ; keys, ignore the key and go to read afresh the kbd.
0130            .ORG 130H
                    ; this routine shows DPH,.DPL and R2 on LCD with
                      spaces in between
0130  74 01     DISPLAY:  MOV A,#1  ;clear display if not cleared earlier
0132  11 90     ACALL CMD

0134  74 F0     MOV A,#F0H
0136  55 83     ANL A,DPH    ; pick most significant  4 bits of DPH
                ; i.e., in 20, it picks 2
0138  C4        SWAP A       ; by swap, 20 becomes 02
0139  78 0F     MOV R0,#0FH  ; 0F is the first internal location for stor-
ing
                                    first character
013B  F6        MOV @R0,A  ;  save in memory address (int.)
013C  18        DEC R0       ; now point to one less address
013D  74 0F     MOV A,#0FH
013F  55 83     ANL A,DPH
0141  F6        MOV @R0,A  ; get the least significant 4 bits of DPH and
                                    write there
0142  18        DEC R0
0143  76 10     MOV @R0,#10H  ; write the space code
0145  18        DEC R0
0146  74 F0     MOV A,#F0H      ; like the above, repeat for DPL
0148  55 82     ANL A,DPL
014A  C4        SWAP A
014B  F6        MOV @R0,A
014C  18        DEC R0
014D  74 0F     MOV A,#0FH
014F  55 82     ANL A,DPL
0151  F6        MOV @R0,A
0152  18        DEC R0
0153  76 10     MOV @R0,#10H   ; space
0155  18        DEC R0
0156  74 F0     MOV A,#F0H     ; like that repeat for R2 register value
0158  5A        ANL A,R2
0159  C4        SWAP A
015A  F6        MOV @R0,A
015B  18        DEC R0
015C  74 0F     MOV A,#0FH
015E  5A        ANL A,R2
015F  F6        MOV @R0,A
                ; by now the values of hex codes for address and data
                ; have been moved into the register addresses 0F -08 hex
```

```
210  0160   LCD_DISP:
211  0160   C0 83       PUSH DPH
212  0162   C0 82       PUSH DPL
213  0164   78 0F       MOV R0,#0FH    ; now start taking each of the hex codes
214  0166   E6          LD1:    MOV A,@R0
215  0167   31 74       ACALL ASCII_CONV  ; and convert them to ascii code
                                  e.g., 2 -> 32
216  0169   11 9F              ACALL LC_wr     ; write to LCD
217  016B   18          DEC R0
218  016C   B8 07 F7    CJNE R0,#7,LD1  ; repeat for 8 such code
219  016F   D0 82       POP DPL
220  0171   D0 83       POP DPH
221  0173   22          RET
222                     ASCII_CONV:    ; converts to ascii code from hex.
223  0174   C3          CLR C
224  0175   54 1F       ANL A,#1FH
225  0177   B4 10 03    CJNE A,#10H,A1  ; if code is 10, it is space code 20H
226  017A   74 20       MOV A,#20H
227  017C   22          RET
228
229  017D   FB          A1:MOV R3,A
230
231  017E   94 09       SUBB A,#09H  ; number keys 0 to 9
232  0180   40 04       JC NUMKEY
233  0182   EB          A_F:  MOV A,R3    ; for code A to F
234  0183   24 37       ADD A,#37H   ;37+A=41H, A
235  0185   22          RET
236  0186   EB          NUMKEY:  MOV A,R3
237  0187   24 30       ADD A,#30H  ; numbers have ascii code 30 to 39
238  0189   22          RET
239  0200               .ORG 200H
240                     CONVERT:  ; this subroutine compares all the scan codes
                                  used by us
241                     ; and converts them to codes and control key codes for
242                     ; use in the monitor program above
243  0200   B4 45 03    CODECHK:CJNE A,#45H,K1
244  0203   74 00                MOV A,#0     ;"0" KEY
245  0205   22                  RET
246  0206   B4 16 03    K1:           CJNE A,#16H,K2
247  0209   74 01                  MOV A,#1     ;"1" KEY
248  020B   22                    RET
249  020C   B4 1E 03    K2:           CJNE A,#1EH,K3
250  020F   74 02                  MOV A,#2     ;"2" KEY
251  0211   22                    RET
252  0212   B4 26 03    K3:     CJNE A,#26H,K41
253  0215   74 03              MOV A,#3     ;"3" KEY
254  0217   22                RET
255  0218   B4 25 03    K41:  CJNE A,#25H,K51
256  021B   74 04              MOV A,#4     ;"4" KEY
257  021D   22                RET
258  021E   B4 2E 03    K51:  CJNE A,#2EH,K61 ;"5" KEY
259  0221   74 05              MOV A,#5
260  0223   22                RET
261  0224   B4 36 03    K61:  CJNE A,#36H,K7  ;"6" KEY
262  0227   74 06              MOV A,#6
263  0229   22                RET
264  022A   B4 3D 03    K7:   CJNE A,#3DH,K8   ;"7" KEY
265  022D   74 07              MOV A,#7
266  022F   22                RET
267  0230   B4 3E 03    K8:   CJNE A,#3EH,K9
268  0233   74 08              MOV A,#8     ;"8" KET
269  0235   22                RET
270  0236   B4 46 03    K9:   CJNE A,#46H,KA   ;"9" KEY
271  0239   74 09              MOV A,#9
272  023B   22                RET
273  023C   B4 70 03    KA:   CJNE A,#70H,KB   ;"0" KEY
274  023F   74 00              MOV A,#0
275  0241   22                RET
276  0242   B4 69 03    KB:   CJNE A,#69H,KD   ;"1" KEY
277  0245   74 01              MOV A,#1
278  0247   22                RET
279  0248   B4 72 03    KD:   CJNE A,#72H,KE   ;"2" KEY
280  024B   74 02              MOV A,#2
281  024D   22                RET
282  024E   B4 7A 03    KE:   CJNE A,#7AH,KF   ;"3" KEY
283  0251   74 03              MOV A,#3
284  0253   22                RET
285  0254   B4 6B 03    KF:   CJNE A,#6BH,KG   ;"4" KEY
286  0257   74 04              MOV A,#4
287  0259   22                RET
288  025A   B4 73 03    KG:   CJNE A,#73H,KH   ;"5" KEY
289  025D   74 05              MOV A,#5
290  025F   22                RET
291  0260   B4 74 03    KH:   CJNE A,#74H,KI   ;"6" KEY
292  0263   74 06              MOV A,#6
293  0265   22                RET
294  0266   B4 6C 03    KI:   CJNE A,#6CH,KJ   ;"7" KEY
295  0269   74 07              MOV A,#7
296  026B   22                RET
297  026C   B4 75 03    KJ:   CJNE A,#75H,KK   ;"8" KEY
298  026F   74 08              MOV A,#8
299  0271   22                RET
300  0272   B4 7D 03    KK:   CJNE A,#7DH,KL   ;"9" KEY
301  0275   74 09              MOV A,#9
302  0277   22                RET
303  0278   B4 5A 03    KL:   CJNE A,#5AH,KM   ;"ENTER" KEY
304  027B   74 47              MOV A,#47H
305  027D   22                RET
306  027E   B4 33 03    KM:   CJNE A,#33H,KN  ;"H" KEY
307  0281   74 42              MOV A,#42H
308  0283   22                RET
309  0284   B4 4B 03    KN:   CJNE A,#4BH,KO  ;"L" KEY
310  0287   74 41              MOV A,#41H
311  0289   22                RET
312  028A   B4 66 03    KO:   CJNE A,#66H,KP  ;bACKSPACE
313  028D   74 44              MOV A,#44H   ;TO DECREMENT
314  028F   22                RET
315  0290   B4 1B 03    KP:   CJNE A,#1BH,KQ
316  0293   74 48              MOV A,#48H     ; INCREMENT ONLY
317                             ; KEY IS S KEY
318  0295   22                RET
319  0296   B4 2C 03    KQ:   CJNE A,#2CH,KR
320  0299   74 4A              MOV A,#4AH     ;REGISTER STORE
321  029B   22                RET          ; IS T KEY
322
323  029C   B4 2D 03    KR:   CJNE A,#2DH,KT
324  029F   74 49              MOV A,#49H     ; R KEY FOR REGISTER
325  02A1   22                RET          ;ACCESS
326  02A2   B4 1C 03    KT:   CJNE A,#1CH,KS
327  02A5   74 0A              MOV A,#0AH     ; KEY A ;
328  02A7   22                RET
329  02A8   B4 32 03    KS:   CJNE A,#32H,KU
330  02AB   74 0B              MOV A,#0BH     ; KEY B ;
331  02AD   22                RET
332  02AE   B4 21 03    KU:   CJNE A,#21H,KV
333  02B1   74 0C              MOV A,#0CH     ; KEY C ;
334  02B3   22                RET
335  02B4   B4 23 03    KV:   CJNE A,#23H,KW
336  02B7   74 0D              MOV A,#0DH     ; KEY D ;
337  02B9   22                RET
338  02BA   B4 24 03    KW:   CJNE A,#24H,KX
339  02BD   74 0E              MOV A,#0EH
340  02BF   22                RET          ; e KEY
341  02C0   B4 2B 03    KX:   CJNE A,#2BH,KY
342  02C3   74 0F              MOV A,#0FH
343  02C5   22                RET          ; F KEY
344
345  02C6   B4 7B 03    KY:   CJNE A,#7BH,KZ
346  02C9   74 4B              MOV A,#4BH     ;SMALL ENTER KEY
347  02CB   22                RET
348
349  02CC               KZ:
350  02CC   B4 34 03           CJNE A,#34H,KZ1  ;"g"
351  02CF   74 43              MOV A,#043H     ;GO KEY
352  02D1   22                RET
353  02D2   B4 2A 03    KZ1:  CJNE A,#2AH,KZ2   ; B KEY EXTRA
354  02D5   74 0B              MOV A,#0BH
355  02D7   22                RET
356  02D8   B4 1D 03    KZ2:  CJNE A,#1DH,LZ3 ; W IS DOWNLOAD KEY
357  02DB   74 4C              MOV A,#4CH
358  02DD   22                RET
359  02DE   74 FF       LZ3:  MOV A,#0FFH     ; if not any of the above keys,
                                  fill with FF hex.
360  02E0   22                 RET
361
362  02E1   75 98 50    SERIN:      MOV SCON,#50H    ; set serial port
363  02E4   75 89 20    MOV TMOD,#20H   ; timer 1 to generate baud rate clock
364  02E7   75 8D FD    MOV TH1,#0FDH   ; value to suit 9600 baud
365  02EA   D2 8E       SETB TR1     ; start timer 1
366  02EC   30 98 FD    RXCK: JNB 98H,$       ;check if serial data awaiting, by
flag
                                  98 testing
367  02EF   C2 98       CLR 98H       ; if set,data is there. clear it for next use
368  02F1   E5 99       MOV A,SBUF     ; take the data from serial buffer into A
369  02F3   12 03 0B    CALL ASCI_HEX   ; convert to hexcode
370  02F6   54 0F       ANL A,#0FH
371  02F8   C4          SWAP A
372  02F9   FD          MOV R5,A
373  02FA   30 98 FD    JNB 98H,$     ; read next data serially
374  02FD   C2 98       CLR 98H
375  02FF   E5 99       MOV A,SBUF
376  0301   12 03 0B    CALL ASCI_HEX   ; convert it to hex
377  0304   54 0F       ANL A,#0FH
378  0306   4D          ORL A,R5        ; pack the two into one byte
379  0307   F0          MOVX @DPTR,A   ; save in memory
380  0308   A3          INC DPTR
381  0309   41 EC       JMP RXCK       ; keep on doing downloading into memory
                                  from serial port
382
383                     ;as before ascii to hex conversion is done below
384  030B   C3          ASCI_HEX:    CLR C
385  030C   FF          MOV R7,A
```

```
386  030D  94 40     SUBB A,#40H
387  030F  40 05     JC ZTO9
388  0311  C3        CLR C
389  0312  EF        MOV A,R7
390  0313  94 37     SUBB A,#37H
391  0315  22        RET
392  0316  C3        ZTO9:     CLR C
393  0317  EF        MOV A,R7
394  0318  94 30     SUBB A,#30H
395  031A  22        RET
396
397
398
399
400
401  031B  END
402
403
404  031B            .END
```

SYMBOLIC REFERENCE TABLE

A1	017D	ASCII_CONV	0174	ASCI_HEX	030B	A_F	0182
CMD	0090	CODECHK	0200	CONVERT	0200	D	0047
DELAY	00F7	DISPLAY	0130	E	004F	END	031B
F	0057	G	0063	H	006C	INIT_LCD	00DE
J	007B	J1	0082	K	0075	K1	0206
K2	020C	K3	0212	K4	00C8	K41	0218
K5	00CC	K51	021E	K6	00D5	K61	0224
K7	022A	K8	0230	K9	0236	KA	023C
KB	0242	KBD	00C0	KC	00A8	KD	0248
KE	024E	KF	0254	KG	025A	KH	0260
KI	0266	KJ	026C	KK	0272	KL	0278
KM	027E	KN	0284	KO	028A	KP	0290
KP1	00C4	KQ	0296	KR	029C	KS	02A8
KT	02A2	KU	02AE	KV	02B4	KW	02BA
KX	02C0	KY	02C6	KZ	02CC	KZ1	02D2
KZ2	02D8	L1	010B	L2	0119	L3	011E
LCD_DISP	0160	LC_wr	009F	LD1	0166	LZ3	02DE
MONI	0025	NUMKEY	0186	Q	0061	RESET	0000
RXCK	02EC	SC1	0032	SC2	0030	SC3	0031
SERIN	02E1	SERINT	00FC	ST	0025	ZTO9	0316

LINES ASSEMBLED : 404 ASSEMBLY ERRORS : 0

Now let's go through various applications of the development kit.

Auto-ranging capacitance meter

The 'Z-80 Based Auto-ranging LCD Capacitance Meter' project published in May 2002 issue was meant to measure the capacitance values in the range of 500 pF to 400 μF and show the results on LCD screen. The same application is possible using this development kit in conjunction with the hardware circuit shown in Fig. 8, which can be easily assembled on a general-purpose PCB and interfaced to the development kit circuit of Fig. 3 (in Part I of the article). The application program has been rewritten for use with this board. It starts from location 4000H (of EEPROM 28C64) and the same is given in Appendix 'B' with complete listing and comments.

The procedure for measurement of the value of unknown capacitor is as follows:

1. Reset the kit using switch S1.

2. Set the address to 4000 using H and L keys.

3. Press G key on the kit's keyboard and read the value on LCD screen

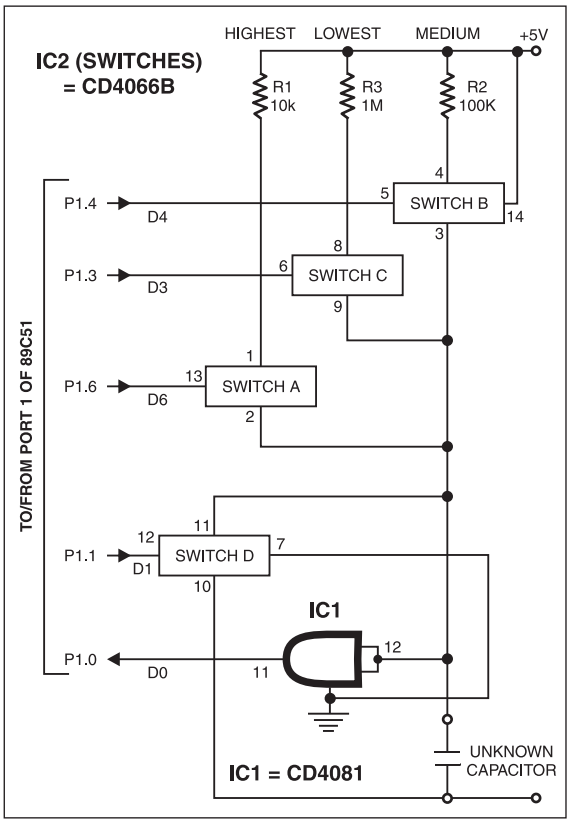


Fig. 8: Circuit diagram of auto-ranging capacitance meter

An abridged schematic showing the interface connections of the development board to LCD module, PC, and the keyboard is shown in Fig. 9. It will be helpful in conceptualisation of the working of various applications covered hereunder.

LCD applications

The circuit has a single-row LCD that can display up to 16 characters. The displayed characters can be made to move from right to left to provide a continuous-moving message display. There is also a provision in these LCD modules to show some graphic characters as well. The method of doing this varies from one type of module to another. To ensure precise working of all the programs given in this part, use a Hitachi make or equivalent single-row LCD module.

Many articles on programming LCD modules have appeared in EFY. The 8085 kit published in Nov. '99 issue of EFY (or Electronics Projects Vol. 20) also uses one such display, where details of the LCD are given. The LCD module comprises a built-in microcontroller (referred to as LCD controller/driver). Thus one actually sends characters to be displayed on LCD screen via its inbuilt microcontroller. Initiaisation of the LCD module, including address setting and display selection, are required before characters are actually entered. The monitor program has already set these things (refer routine at addresses 00DE in Appendix 'A' Monitor Program Listings for INIT_LCD). Thus, as far as the user is concerned, he can just write to the LCD after setting the address to which the cursor is to point.

Two subroutines are available in the monitor program for accessing the LCD, which can be called from there. These are:

1. CMD routine at address 0090H for writing instruction into LCD Command register.

2. LC_WR routine at address 009FH for writing data to LCD data display RAM (DD-RAM).

The CMD routine is used for writing commands for positioning the cursor point, cursor blinking, left shifting of the display on each character-write operation, etc. The address where one wishes to write to the LCD memory is also set by one of these commands. The LC_WR routine is the one that actually writes to the LCD. Once the command for a certain display operation is entered, one can enter characters one after another using the LC_WR routine. However, it first requires clearing of the display using the command 01 as follows:

DISPLAY	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16					
LINE 1	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15
LINE 2	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55

```
20 00  74 01      MOV A,#01
       12 00 90   CALL CMD
       80 FE      SJMP $ ;$ means here
       itself
```

The last instruction is a one-instruction infinite loop. Execute this program after entering the seven bytes from 2000 onwards. You will note that the display has cleared and a cursor is seen at the left end.

The programs for LCD can be developed easily if you know how the display characters are numbered internally. There are only 16 characters on the LCD as seen, but there are 40 memory locations in a RAM within the LCD module itself. This is called DD-RAM. Address map for a HD44780 based 16-character x 1-line LCD module is shown here:

The shaded area (00 to 15) is the visible display. It measures 16 characters per line x 2 lines. The number in each box is the DD RAM address that corresponds to the screen position.

The command 06 hex shifts the cursor to right after a write to a location. The command for left shifting after a write operation is 04 hex. Thus if 04 hex is entered, the cursor moves to left instead of right. The address of DDRAM is incremented or decremented accordingly. The commands for cursor and display shifting are listed below:

Command 04	Shifts cursor left
Command 06	Shifts cursor right
Command 07	Shifts the display also to left

When 07 is entered, the display itself shifts to left. That is, if the 16 characters viewed on the display are considered to be an arc of a circle with 40 letters (1 to 40) like a clock's dial, the display shift moves the DD addresses on the view towards the left. This command is used for showing a continuous display (up to 40 characters) by writing to the LCD (DD RAM) one after the other. The program for writing to the first 16 display positions on the LCD is:

```
20 00  74 06      MOV A,#06  ; To shift cursor
                  right after each entry
20 02  12 00 90   CALL CMD
20 05  74 01      MOV A,#01; To clear display
                  with cursor returning to
                  extreme left edge of display
20 07  12 00 90   CALL CMD
20 0A  90 20 40   MOV DPTR, # MESSAGE
20 0D  E0         K1: MOVX A,@DPTR ; start
                  reading from message table
                  (loc.2040)
20 0E  B4 FF 02   CJNE A,#FFH,K2 ;
                  Check for FF hex (i.e. end of
                  message)
20 11  80 06      SJMP END
20 13  12 00 9F   K2: CALL LC-WR ;WRITES
                                  TO DISPLAY
20 16  A3         INC DPTR
20 17  80 F4      SJMP            K1
20 19  80 FE      END: SJMP $ ;Loop here
                       infinitely (i.e. halt)
2040   MESSAGE:   DB 41H, 42H, 43H, 44H,
                  45H, 46H, 47H, 48H, 49H, 4AH, 4BH,
                  4CH, 4DH, 4EH, 4FH, FFH :ASCII code
                  of 15 characters ending with FF; DB is
                  define byte (assembler directive)
20 40  41 42 43 44 45 46 47 48
2048   49 4A 4B 4C 4D 4E 4F  FF
```

Program execution. Any program from the development kit can be executed by entering (via the kit's keyboard) hex value of the MSB of address (20 in this case) and pressing H key, then entering the LSB (00 in this case) and pressing L key, and pressing G key for execution.

After executing the program from 2000H, we see ABCDEFGHIJKLMNO_ on the display, with the cursor at the 16th position. Add one more ASCII code 50 (for P) to the message table as under:

```
20 40  41 42 43 44 45 46 47 48 49 4A 4B 4C
       4D 4E 4F 50  FF
```

Now on running the program again, it shows ABCDEFGHIJKLMNOP. The cursor has gone to the right of 16th character, hence it is invisible. Add two or three more characters to the table and again run the program. Still the display shows only 16 characters (ABCDEF GHIJKLMNOP). In order to show a message containing more than 16 characters, we have to either clear the display and start again from the extreme left, or let the display scroll to left further by 16 characters.

Scrolling display. Rewrite the program as follows:

```
20 00  74 06      MOV A,#06
20 02  12 00 90   CALL CMD
20 05  74 01      MOV A, #01
20 07  12 00 90   CALL CMD
20 0A  90 20 40   MOV DPTR, # MESSAGE
20 0D  7C 11      MOV R4,#11H  ; FIRST
                  16+1 TO COUNT
20 0F  E0         K1: MOVX  A,@DPTR
20 10  B4 FF 02   CJNE  A,# FFH, K2
20 13  80 13 SJMP END
20 15  DC 02 DJNZ R4,K2  ; if count not over,
       go to K2
20     80 06      SJMP F1  ; if >count, go to F1
20 19  12  00  9F  K2:  CALL  LC_WR  ;
       WRITES  TO DISPLAY
20 1C  A3         INC DPTR
20 1D  80 F0 SJMP K1
20 1F  80 FE F1: SJMP $  ; halt here
2040   MESSAGE: DB 41H, 42H, 43H, 44H,
       45H,  46H, 47H, 48H, 49H, 4AH,
204A   DB 4BH, 4CH, 4DH, 4EH, 4FH, 50H,
       51H, 52H, 53H, 54H, 55H, FFH
20 40  41 42 43 44 45 46 47 48
2048   49 4A 4B 4C 4D 4E 4F
204F   50 51 52  53 54 55 FF
```

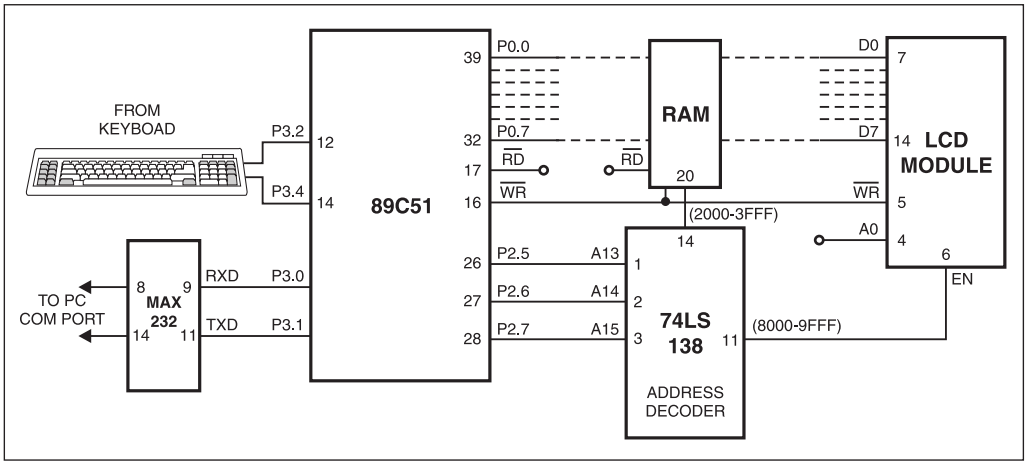


Fig. 9: Microcontroller application through PC

Modify the program after the display of 16 characters, i.e. at 201F, to add the display shift command as under:

```
20 1F  74 07 F1:  MOV A,#7  ;COMMAND TO
       SHIFT DISPLAY LEFT
20 21  12 00 90   CALL CMD ;AFTER EACH
       LCD WRITE OPERATION
20 24  7C 18 MOV R5,#18H  ; FOR ANOTHER
       24  CHARACTERS
20 26  80 E7 SJMP K1
20 28  80 FE END: SJMP $
```

Now the display shows the extra characters (EFGHIJKLMOPQRSTU), but the movement is so rapid that one cannot see the characters as they move. It stops at U, because there is an FF after U (55H).

Speed control. For moving-type display, the writing is to be slowed down. For this, after every write operation, a time delay has to be called. This can be interposed after the instruction CALL LC-_WRITE at location 20 19H. The jump addresses also will have to be changed since we have used relative jump instructions at three points in this program. To introduce a delay without much alteration in the program, the call to LC_WRITE may be made at a different address. Then, the delay can be added there. For example, call address at 20 19 for LCD_WRITE may be changed to 20 80 as under:

```
20 80  12 00 9F    CALL LC_WRITE
20 83  78 FF       MOV R0,#FFH
20 85  00 C0:      NOP ;double loop time delay
20 86  79 FF       MOV R1,#FFH
20 88  00       C1: NOP
20 89  D9 FD       DJNZ R1,C1
20 8B  D8 F8       DJNZ R0,C0
20 8D  22          RET
```

The double-loop time delay is needed because a single loop even with FFh for the number cannot make the display movement slow enough. The change is to be made at location 2019 H is 12 20 80 in place of 12 00 9F.

Now, after execution, the program works with a slow movement.

Continuous display of more than 40 characters. If you want the characters to move continuously with more than 40 characters in a message, the above program will fail. That is, because the DDRAM is having only 40 locations. After 40th character is written, the display repeats from the first character written earlier. Therefore the fresh characters coming after the 40th character will be missed.

Therefore after the 40th character, we have to set the address back to location 1. Then the rotation of the characters to the left will appear continuous. It is advisable to start writing always from the right end of the display and then, after the 40th character, reset the address back to 1. The program for continuous moving message display is given below:

```
1  2000              ORG 2000H
2  2000   90 20 34   MOV DPTR, 4#MESSAGE
          ;POINT TO MESSAGE TABLE
          WITH FF AT END
3  2003   7C 18      MOV R4,# 18H ; TO
          COUNT 24 CHARACTERS MORE
          IN DDRAM (LCD)
4  2005   74 07      MOV A,# 07H ; Set LEFT
          SHIFT DISPLAY MODE
5  2007   12 00 90   CALL CMD
6  200A   74 90      MOV A,#90H ; SET ADD-
          RESS TO BE 10H OR 16
7  200C   12 00 90   CALL CMD
8  200F   12 20 20   X: CALL CHARACTER ;
          GETS ASCII CHARACTER FROM
          TABLE
9  2012   12 00 9F   CALL  LC_WRITE
                     ; WRITES TO LCD DISPLAY
10 2015   DC F8      DJNZ R4, X ;18H OR 24
          TIMES
11 2017   7C 28      MOV R4,#28H  ; NOW
          40 TIMES
12 2019   74 80      MOV A,#80H ; SET
          ADDRESS TO BE 0
13 201B   12 00 90   CALL CMD ; WRITE TO
          lcd
14 201E   80         EF SJMP X; REPEAT
       DISPLAYING EACH KEY PRESSING
15 2020   E0         CHARACTER : MOVX
          A,@DPTR
16 2021   B4 FF 02   CJNE A,#FFH, MORE
17 2024   80 FE      OVER:  SJMP $
18 2026   12 20 2B   MORE: CALL DELAY
19 2029   A3         INC DPTR
20 202A   22         RET
21
22 202B   7E FF DELAY:  MOV R6, #FFH
23 202D   7D FF      D1: MOV R5,# FFH
24 202F   DD FE      DJNZ R5,$
25 2031   DE FA      DJNZ R6, D1
26 2033   22         RET
27 2034   MESSAGE:
28 2034   41 42 43 44          DB 41H, 42H,
          43H, 44H, 45H, 46H, 47H, 48H, 49H,
          4AH, 4BH, 4CH, 4DH, 4EH, 4FH
   2038   45 46 47 48
   203C   49 4A 4B 4C
   2040   4D 4E 4F
29 2043   4F 50 51 52     DB 4FH, 50H, 51H,
          52H, 53H, 54H, 55H, 56H, 57H, 58H,
          59H, 5AH, 5BH,  5CH, 5DH, 5EH
   2047   53 54 55 56
   204B   57 58 59 5A
   204F   5B 5C 5D 5E
30 2053   5F 60 61 FF   DB 5FH, 60H, 61H,
          FFH
31 0090             .ORG 90H
32 0090   22        COMMAND:    RET
33 009F             .ORG 9FH
34 009F   22        LC_WRITE:   RET
35
36 00A0             .END
```

The above program can display any length of message continuously in a moving fashion on the LCD. The message table must be entered along with the program.

Keyboard-based moving message display. For this, it is advisable to have a keyboard to enter the data for the message using the keyboard routine included in the firmware (monitor program) at 00C0H.

Keyboard reading routine. Understanding the keyboard routine is desirable. The keyboard scan code is collected by this routine starting at 00C0H. The PC AT keyboard is used here and the keyboard connector cable has Clock and Data lines that connect to pins P3.2 (timing pulse) and P3.4 (data bit pulse), respectively, of 89C51. The routine first looks at the P3.4 for the timing pulse to go low, which happens when data is sent from the keyboard upon a keypress by the user. The following two instructions check for the P3.2 to go low in an endless loop until a key is pressed:

```
BACK:   MOV C, P3.2
        JC BACK
```

The next two instructions look for a low-to-high transition on the timing bit:

```
BACK1:  MOV C,P3.2  ; READ THE
        CLOCK PULSE BIT AGAIN
        JNC BACK 1
```

After the timing bit has become high, the following instruction waits for it to go low again:

```
BACK2:  MOV C, P3.2
        JC BACK2
```

Thus, the start bit of the keycode serial output is bypassed (which comprises high to low, low to high, and back to low transitions) for the second bit in the serial stream. This is the beginning of the first data bit. The following four instructions read the data bit information (high or low) from pin 14 into the carry flag:

```
MOV  C,P3.4  ; READ THE DATA BIT
             AT PIN 14  IN  CARRY FLAG
MOV  A,R7    ; READ THE PAST DATA
```

```
                    STORAGE REGISTER
RRC A           ; PUT THE CARRY
MOV  R7,A       ; REPLACE IN THE STORAGE
                  REGISTER
```

Here, register R7 is used to collect the carry bit into it by the process of shifting, bit after bit from left to right using the RRC instruction.

The following two instructions look at the timing bit again:

```
BACK3 MOV C,P3.2  ; READ TIMING BIT
                    INTO CARRY FLAG
JNC BACK3
```

Then we have to repeat reading of the data, until eight bits are collected, and get back to the previous instruction at BACK2: DJNZ R3, BACK2. Here, R3 is used for counting up to 8, for which it has to be previously set to 8. The routine ends with a call to a time delay and exits as under:

```
CALL DELAY
RET
```

There is probably no simpler routine developed to read and interface LCD module to the 89C51 !

Putting it all together will need the following two instructions at the beginning:

```
MOV  R7,#00   ;CLEAR THE  R7 REGISTER
                WHICH  COLLECTS DATA
MOV  R3,#8    ; COUNTER FOR 8 BITS TO
                BE COLLECTED
```

Display from keyboard. Now it is easy to develop the routine to read every key depression and display it on the LCD. We first try to change the message-picking operation using the keyboard routine (reading operation) to get every character. So, we replace the CALL CHARACTER line (at location 2020) in the program by CALL KBD. Thus we get:

```
200F    20 00 C0   CALL KBD  ; CALLS THE
                     KBD ROUTINE
```

Now on running the program with this change, we don't get the keys right: when A is typed, some strange character appears on the LCD! This is because the keyboard scan code that it sends is not the ASCII code of the key that is pressed. The PC keyboard has a scan code for each key (refer Fig. 4 in Part I). And each key has an ASCII code. For example, the key Q has an ASCII code of 51 hex. So, after the keyboard routine, we have to use another routine to convert the code into ASCII code. This requires a look-up table (LUT) as mentioned in the following brief routine:

```
KBDASCII:
C: CALL KBD  ;CALL THE BASIC KEYBO-
               ARD ROUTINE (at 00C0)
CJNE R7,#F0H,C ;HAS KEY BEEN RELEASED?
               LOOP BACK  IF NO
CALL KBD     ; CALL THE SCAN CODE
               READING ROUTINE AGAIN
CJNE R7,#FFH,C1 ; FFH DENOTES NULL KEY
               SO LOOP BACK
SJMP C       ;  TO C AGAIN, IF IT IS FF,
               OTHERWISE
C1:  CALL TABLOOK  ; LOOK UP TABLE
MOV R7,A     ;WRITE DATA INTO R7
RET
TABLOOK:  MOVC  A,@A+DPTR ; GET THE
            CODE FROM THE TABLE DOWN
RET
TABLE:
     FFH,FFH,FFH.... ETC.
```

The above program has several points to ponder over. The keyboard routine basically collects an 8-bit key scan code. The PC keyboard outputs the scan code for a key not only when the key is pressed but also when it is released. On releasing a key, F0H code is first sent, followed by the scan code. Thus, the scan code after a key release is good enough for us. After the first call to keyboard routine, the code is checked for F0H.

If F0H is not present, we go back and scan again for this key release code. When we get F0H code, the next code must be a valid scan code. If it is not a valid scan code, it will be FFH. The same is ignored, and we go back to point C again. On the other hand, if the code following F0H is a valid scan code, a table loop-up program (TABLOOK) is called. This program reads the table and gets the contents in accumulator and also saves them in register R7. This serves as a basis to read from a keyboard and display it on the LCD. The routine for the same is given below:

```
1  2000 .ORG  2000H
2  00 30 COUNT .EQU 30H
3  2000 75 30 18    MOV COUNT,#18H
4  2003 74 07       MOV A,#07H  ;LEFT
                    SHIFT DISPLAY MODE
5  2005 12 00 90    CALL 0090H  ;COM-
                    MAND
6  2008 74 90       MOV A,#90H  ;SET
                    ADDRESS TO BE 10H
OR 16
7  200A 12 00 90    CALL 90H  ; COMMAND
8  200D 12 20 25    X: CALL KBD_ASCI
          ;KEYBD ROUTINE GIVES
          ASCII CODE
9  2010 B4FF02      CJNEA,#FFHX1
          ;WRITE TO LCD IF VALID KEY
          CODE, NOTE IF FFH
10 2013 80 F8    SJMP X  ;OTHERWISE
          READ KEYBOARD FOR ANO-
          THER VALID KEY.
11 2015 12 00 9F    X1: CALL 009FH
          ;LC_WRITE ;WRITES TO LCD
          DISPLAY
12 2018 D5 30 F2    DJNZ COUNT,X
13                  ; 24 TIMES
14 201B 75 30 28    MOV COUNT,#28H ;
                    NOW 40 TIMES
15 201E 74 80       MOV A,#80H ; SET
                    ADDRESS TO BE 0
16 2020 12 00 90    CALL 0090H
          ;COMMAND ; WRITE TO LCD
17 2023 80 E8       SJMP X  ;REPEAT
          DISPLAYING EACH KEY
          PRESSING
18
19 2025 12 00 C0    KBD_ASCI:  CALL
          C0H ;CALL KEYBOARD ROUT-
          INE IN MONTIOR PROGRAM
20 2028 BF F0 FA    CJNE R7, # F0H,
          KBD_ASCI  ;HAS KEY
          RELEASE CODE COME? (F0H)
21 202B 12 00 C0    CALL C0H ;CALL KBD
22 202E BF FF 02    CJNE R7,# FFH,
                    C1
23 2031   80 F2     SJMP KBD_ASCI
24 2033   12 20 38  C1: CALL TABLE_
                    LK
25 2036   FF        MOV R7,A
26 2037   22        RET
27
28 2038   00        TABLE_LK:
                    NOP
29 2039   83        MOVC A,@A+PC
30 203A   22        RET
31
32 203B             TABLE:
33 203B   FF FF FF FF      CODES:DB
          -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
          -1, -1, "9", "“, -1, -1, -1, -1, -1, -
          1, "q"
   203F   FF FF FF FF
   2043   FF FF FF FF
   2047   39 60 FF FF
   204B   FF FF FF FF
   204F   71
34
35 2050   31 FF FF FF  DB "1", -1, -1, -1,
          "z", "s", "a", "w", "2", -1, -1, "c",
          "x", "d", "e", "4"
   2054   7A 73 61 77
   2058   32 FF FF 63
   205C   78 64 65 34
36 2060   33 FF FF 20  DB "3", -1, -1, " ",
          "v", "f", "t", "r", "5", -1, -1, "n",
          "b", "h", "g", "y"
   2064   76 66 74 72
```

```
    2068    35 FF FF 6E
    206C    62 68 67 79
37  2070    36 FF FF FF  DB "6", -1, -1, -1,
                ”m”, ”j”, ”u”, ”7", ”8", -1, -1, ”,”,
                ”k”, ”i”, ”o”, ”0"
    2074    6D 6A 75 37
    2078    38 FF FF 2C
    207C    6B 69 6F 30
38  2080    39 FF FF 2E   DB “9”, -1, -1, ”.”
                ,”/”, ”l”, ”;”, ”p”, ”-”, -1, -1, -1,
                ””””, -1, ”[“, ”=”
    2084    2F 6C 3B 70
    2088   2D FF FF FF
    208C   27 FF 5B 3D
39  2090    FF FF FF FF  DB  -1, -1, -1, -1,
                0dh, ”]”, ”\”, -1, -1, -1, -1, -1, -1,
                -1, -1
    2094    0D 5D 5C FF
    2098    FF FF FF FF
    209C    FF FF FF
40
41
42  209F                .END
```

So, after this program is entered and executed, we can continue typing on the keyboard of the 89C51 kit, with letters displayed on the LCD module. Only lower-case letters have been used in the above table. If lower-case ASCII code is replaced with upper-case ASCII code, we get only capital letters. To be able to use both upper- and lower-case letters, we have to sense the shift key and accordingly look up either of the two tables. This makes the program much more complicated and hence we stop at this stage.

Serial communication

The serial port pins TXD and RXD (P3.1 (pin 11) and P3.0 (pin 10), respectively) of the 89C51 can be employed for data transfer in the asynchronous mode from any other computer or microcontroller. The serial port in 89C51 is controlled by a register called SCON at address 98H. The bit-wise functions (SCON.7 through SCON.0) of this register are given below:

Thus, to set up for any of the above modes, SCON must be written with a data byte. For example, it is 52H for our 8-bit UART mode, which is Mode 1 for the 89C51.

Generating the baud rate clock by using timer. For the clock of the serial data bits, the timer in the chip is used. The timer clock is programmable. So, depending upon the crystal used for the 89C51, the baud rate timing will vary. (Normally, the crystal used is of 12 MHz.) However, baud rates are generally in standard numbers (300, 1200, 2400, 9600, 11200, and so on), which are available in a PC for serial data communication via serial ports (COM1, COM2, COM 3,…). So, we have to use only one of these values for the baud rate.

Exact generation of a baud rate, say, 9600, will be possible only if the crystal is chosen to suit the timing. For this, a 11.059MHz crystal is used. We have selected and used a baud rate of 9600 for communicating from and to the PC via its serial port.

There are two timers (timer 0 and timer 1) in the 89C51. Timer 1 is to be programmed as an 8-bit timer, with a clock frequency input of crystal frequency prescaled by 12. Thus, clock to timer 1 is 1 MHz with a 12MHz crystal. Timer 1 is programmed to work as an auto-reload timer of eight bits. Thus, if its register TH1 is written with a number, say, 250, it will overflow after 15 timer clocks, or 15 microseconds. So, we get timing events every 15 microseconds. This is only a rough description for easy understanding.

So, at every timer 1 interrupt due to overflow, the bit is transmitted/received. For baud rate of 9600, we can get the reload value into the timer-high register TH1:

Baud rate= 1/32×overflow rate of timer 1

$$\text{Overflow rate} = \frac{\text{Oscillator frequency}}{12\times(256-\text{TH1})}$$

We get TH1=FD hex as the reload value into the timer-high register TH1. To get a more accurate baud rate of 9600, use an 11.09MHz crystal, as recommended by Intel in its application note.

Timer 1 is to be started by setting timer 1 start bit at address location TR1 (8D hex). TR1 is the sixth bit in register TCON. The byte address of the timer control register TCON is 88H. The bit address of TR1 is therefore TCON.6 or 88 + 6 = 8E hex. We start timer 1 by setting bit 8E hex.

Serial port initialisation. As per the above arguments, the following code is to be written to initialise the serial port for data communication at a baud rate of 9600 on our kit for use with a PC's COM port:

```
1 MOV SCON, #52H ; SET UP SERIAL PORT
                   MODE 1
2 MOV TMOD,#20H  ; SET UP THE TIMER
                   1 FOR AUTO RELOAD
                   8-bit
3 MOV  TH1,#FDH  ; VALUE FOR timer
                   high register
4 SETB TR1       ; Starts timer 1
```

Once these instructions are executed, the serial port is ready for use because

SCON: Serial Port Control Register (Bit Addressable)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI
SM0	SCON.7	Serial Port mode specifier. (1)					
SM1	SCON.6	Serial Port mode specifier. (1)					
SM2	SCON.5	Enables the multiprocessor communication feature in modes 2 and 3. In mode 2 or 3, if SM2 is set to 1, then RI is not activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2=1, then RI is not activated if a valid stop bit was not received. In mode 0, SM2 should be 0.					
REN	SCON.4	Set/cleared by software to enable/disable reception.					
TB8	SCON.3	The 9th bit that is transmitted in modes 2 and 3. Set/cleared by software.					
RB8	SCON.2	In modes 2 and 3, is the 9th data bit that was received. In mode 1, if SM2=0, RB8 is the stop bit that was received. In mode 0, RB8 is not used.					
TI	SCON.1	Transmit interrupt flag. Set by hardware at the end of the 8th bit time in mode 0 or at the beginning of the stop bit in the other modes. Must be cleared by software.					
RI	SCON.0	Receive interrupt flag. Set by hardware at the end of the 8th bit time in mode 0 or halfway through the stop bit time in the other modes (except see SM2). Must be cleared by software.					

Note (1).

SM0	SM1	Mode	Description	Baud Rate
0	0	0	Shift register	Fose/12
0	1	1	8-bit UART	Variable
1	0	2	9-bit UART	Fose/6.4 OR Fose/3.2
1	1	3	9-bit UART	Variable

the baud rate clock has now been generated.

Downloading from the PC. First,we shall discuss the program that is used to enter a program code from a PC onto the RAM chip of our 89C51 development board. The PC is convenient for developing the code for any Assembly language program.

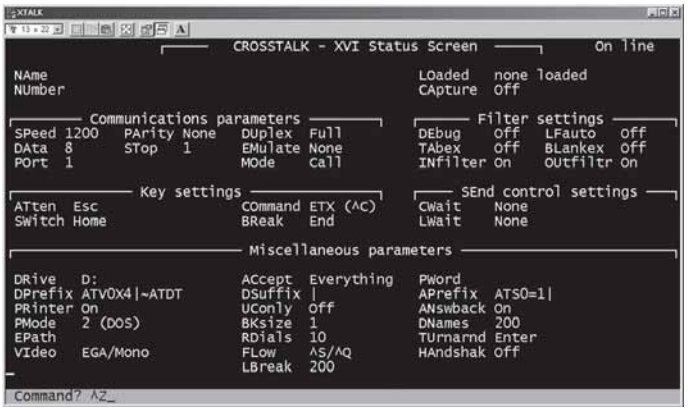
The PC serial port (COM1 or COM2) is connected to the development kit using a 3-wire cable by making use of MAX-232 chip on the kit. After wiring the PC's COM port to the kit, we can start communication between the PC and the development kit as explained in succeeding paragraphs.

Downloading 8-bit data from the PC to the kit. After initialising the serial port for mode 1 for a baud rate of 9600, we read the serial port interrupt flag 98H (a bit address in the 89C51 internal RAM). The instruction (30 98 FD MORE: JNB RI,$) keeps looking at 98H and jumps only when this bit is set, which happens upon reception of a byte through RXD pin 10. Then the next instruction (C2 98 CLR RI) clears bit RI, so the next data byte can be checked similarly.

Once a data byte has been received into the SBUF buffer, it is moved into the accumulator, and then stored into the memory using instruction MOVX. The memory pointer DPTR is incremented. Then the routine loops back to read more data. The complete routine is given below:

```
1  2700   .ORG 2700H
2
3  2700   90 20 00   MOV DPTR,#2000H
          ;PROGRAM TO DOWNLOAD AT
          2000H ONWARDS
4  2703   75 98 52   INI: MOV SCON,#52H
          ;SET SERIAL MODE 1
5  2706   75 89 20   MOV TMOD,#20H   ;
          TIMER 1 NON GATED AUTO
          RELOAD
6  2709   75 8D FD  MOV TH1,#FDH
          ;COUNT FOR BAUD RATE 9600.
7  270C   D2 8E      SETB TR1   ;START
          TIMER 1
8  270E   30 98 FD  MORE:     JNB RI,$
          ;LOOK FOR RECEIVE DATA INTER
          RUPT FLAG SET
9  2711   C2 98      CLR RI     ; CLEAR RI
          FLAG
102713   E5 99      MOV A,SBUF ;MOV
          FROM SBUF TO A
112715   F0         MOVX @DPTR,A ;WRITE
          INTO RAM starting LOC. 2000H
122716   A3         INC DPTR  ;INCREMENT
132717   80 F5      SJMP MORE
```

We can execute this program from address 2700H in the kit. Then, in order to be able to send data from the PC, we must have a serial communication program, such as CROSSTALK. On running XTALK.EXE on the PC by entering XTALK at DOS prompt >, a status window as shown in the following screenshot appears with a help screen:



At the bottom, there is a line for entry of commands. In that, enter the number of COM port. For example, for COM port 3, type PO using the PC's keyboard and press Enter, then type 3 and Enter. Now enter the baud rate by typing SP and pressing Enter, followed by typing 9600 and pressing Enter. This sets the speed to 9600 bauds. Now type GO LO and press Enter.

The Xtalk program is now able to send and receive data. We can type 1234567890 and so on after executing the above-mentioned program on the 89C51 board from location 2700H. The data entered from the PC will go into successive memory locations starting 2000 onwards. Now terminate the send/receive session by pressing Escape key on the PC.

To come out of the XTALK session, type the command QUIT (or QU) via the PC's keyboard and press Enter to come to DOS. Now, on the 89C51 development board, press Reset switch and examine memory locations 2000 H onwards. You will observe data 31, 32, 33, etc on the LCD screen at successive locations. These are ASCII codes for 123... etc, as entered via PC. The program will function properly only when we send alphabets or string characters.

Sending the code or program file from the PC. The code for any 8-bit microprocessor comprises two nibbles. For example, the code for instruction INC DPTR is A3. We will spell it as A and 3. So, if we send these characters from the PC using XTALK or any other serial communication program, it will send only its ASCII code comprising seven bits and will clip the eighth bit. So, A3 (10100011) will go only as 23H (0100011).

In order to send the code correctly, we have to split the code in ASCII (for each hex digit) into two ASCII bytes for each code byte. Thus A and 3 will be sent as 41H for A and 33H for 3, in that order. Therefore after an Assembly language program is built and its code bytes are known in binary form as a file, the contents of that file must be converted into an ASCII file. Such an ASCII file will have double the size of the input binary file.

Thus, the process of program development boils down to:

1. Development of the source program in Assembly code (with extension .ASM) using any text editor or wordprocessor (in non-document mode). The same is to be converted into its object code file (with extension .OBJ) using X8051.exe or other suitable cross compiler.

2. Conversion of the above code into binary format (with .TSK extension) using the LINK151.exe program.

3. Conversion of the binary file into ASCII file using the program BIN4ASC.exe. It will convert the .TSK file formed in the previous step into an ASCII file with .ASC extension.

(***Note***. X8051.exe, LINK151.exe, and BIN4ASC.exe have been included in CD.

The final ASCII file with .ASC extension is to be sent from the PC using the XTALK communication program (SE command). For this, the program needs to be modified at 2700H, because we now send each byte in two halves. The modified program for downloading an ASCII converted code file from the PC to the kit (starting location 2000H) is given below:

```
1  2700   .ORG 2700H
2
3  2700   90 20 00   MOV DPTR,#2000H
          ;PROGRAM TO DOWNLOAD AT
          2000H ONWARDS
4  2703   75 98 52   INI:  MOV SCON, #52H
```

```
                    ;SET SERIAL MODE 1
5  2706  75 89 20   MOV TMOD,#20H
                    ;TIMER 1 NON GATED AUTO
                    RELOAD
6  2709  75 8D FD   MOV TH1,#FDH   ;THE
                    COUNT FOR BAUD RATE 9600
                    WITH 11.09 MHz CRYSTAL
7  270C  D2 8E      SETB TR1  ;START
                    TIMER 1
8  270E  30 98 FD   MORE:     JNB RI,$
                    ;LOOK FOR RECEIVE DATA
                    INTER-RUPT FLAG SET
9  2711  C2 98      CLR RI     ; CLEAR
                    THAT FLAG
10 2713  E5 99      MOV A,SBUF ;MOV
                    FROM SBUF TO A
11 2715  12 27 2D   CALL ASCIIHEX  ;CON-
                    VERT THAT 7 BIT DATA IN ASCII
                    FORMTO HEX FORM
12 2718  54 0F      ANL A,#0FH     ;SKIP
                    OFF UPPER NIBBLE
13 271A  C4         SWAP A         ;BRING
                    THAT TO LOWER NIBBLE POSITION
14 271B  FD         MOV R5,A       ;SAVE IN
                    R5 REGISTER
15 271C  30 98 FD   JNB RI,$       ;LOOK
                    FOR SECOND PART OF  8 BIT
                    CODE FROM SERIAL DATA
16 271F  C2 98      CLR RI ; CLEAR SERIAL
                    RECEIVER INTERRUPT FLAG
17 2721  E5 99      MOV A,SBUF   ;MOVE
                    SECOND BYTE CODE IN ASCII
                    FORM NTO AC CUMULATOR
18 2723  12 27 2D   CALL ASCIIHEX ;CON-
                    VERT IT TO HEX FORM
19 2726  54 0F      ANL A,#0FH   ; PICK
                    THE LOWER NIBLE AND
20 2728  4D         ORL A,R5     ; ADD IT
                    WITH THE UPPER NIBBLE
                    ALREADY IN R5
21 2729  F0         MOVX @DPTR,A  ;
                    MOVE THIS 8 BIT CODE BYTE
                    INTO 2THE ADDRESS
22 272A  A3         INC DPTR      ; FOR
                    STORING AND INCREMENT THE
                    ADDRESS
23 272B  E1 0E      JMP  MORE     ; GO
                    AND TAKE THE NEXT DATA BYTE
                    FROM SERIAL INPUT
24 272D  ASCIIHEX:
25 272D  C3         CLR C
26 272E  FF         MOV R7,A       ;SAVE
                    A IN R7
27 272F  94 40      SUBB A,#40H    ; ASCII
                    CODES FOR A TO f HAVE VALUES
                          > 4O
28  2731 40 05      JC Z_9         ; IF LESS
                    THAN 40, IT MUST BE NUMBERS 0
                                   TO 9
29 2733  C3         CLR C
30 2734  EF         MOV A,R7     ; GET
                    THE DATA BACK IN A
31 2735  94 37      SUBB A,#37H   ; SUB
                    TRACT 37H FROM DATA ( A TO f)
                    TO GET IN HEX
32 2737  22         RET
33 2738  C3         Z_9: CLR C
34 2739  EF         MOV A,R7   ;GET DATA
35 273A  94 30      SUBB A,#30H   ; ASCII
                    FOR 0 TO 9 IS 30 TO 39, SUBTRACT
                    30
36 273C  22         RET
37
38 273D             .END
```

At line No. 18, the program calls the subroutine ASCIIhex. This converts the ASCII into a single nibble. For example, the 41 code of A is converted into 0A hex. Now, when a second byte is received, that too is converted similarly. Combining the two parts in register R5 completes the code byte. This is done in lines 13 to 20 in the above program.

This program, starting at location 2700H, is useful for transferring any developed code into the kit's memory and later executing it at 2000H. It is to be entered by hand via the kit's keyboard. After this is loaded at 2700H, it is also executed from that address. Since file transfer from the PC to the kit will occupy the address 2000H onwards, we've used 2700H for the program. This program can be included as part of the monitor routine at a different address inside the ROM space of the 89C51, if desired.

Program to send data from 89C51 to the PC. The previous program checked for data received into the kit, picked it up one by one, and saved it. The following program will send data from the kit's memory or keyboard to the PC:

```
Send data Program.
75 98 52  INI:    MOV SCON,#52H
                  ;SET SERIAL MODE 1
75 89 20          MOV TMOD,#20H   ;
                  TIMER 1 NON GATED
                  AUTO RELOAD
75 8D FD          MOV TH1,#FDH   ;THE
                  COUNT FOR BAUD RATE
                  9600 WITH 11.09 MHz
                  CRYSTAL
D2 8E     SETB TR1 ;START
                  TIMER 1
```

The above is the initialisation part of the serial port, where no interrupts are enabled.

Now, for sending a byte, the bit 99H (TI Flag, SCON.1) has to be polled using the following instructions:

```
30 99 FD   MORE:JNB
TI,$  ; $ DENOTES LOOP
UNTIL TI BIT SET
    C2 99  CLR TI  ; CLEAR THE TI BIT FOR
                   NEXT USE
    F5 99  MOV SBUF,A ; MOVE DATA INTO
                   SERIAL PORT BUFF
```

Note that in this case, the data is moved into the SBUF, while the previous program for downloading from the PC had the opposite instruction MOV A,SBUF. The bit here is TI or 99H, while it was 98H (RI) earlier. Whatever data (byte) is moved into SBUF, it is transmitted via the TXD pin. We continue to do this one character after another using instruction
02 20 0E JMP MORE

The above sequence of instructions works. But we have not yet configured the PC for receiving the data that is being transmitted from the kit. So, we run the XTALK on the PC, initialise its port and baud rate (9600), and then GO LO. After this, we observe that the character sent from kit's keyboard is printed on the PC's monitor screen continuously and at a fast rate.

This suffices to demonstrate the transmit action of the above program. But, when we want to send data from RAM into PC, we need to increment the RAM address every time and then pick that data before transmitting. So, the routine is to be modified as under:

```
1  2000   .ORG 2000H
2  2000   90 22 00   MOV DPTR,#2200H
                     ;DATA PART STORED FROM HERE
3  2003   75 98 52   MOV SCON, #52   ; SE-
                     RIAL PORT MODE 1
4  2006   75 89 20   MOV TMOD,#20H   ;
                     TIMER 1 8 BIT AUTO RELOAD MODE
5  2009   75 8D FD  MOV TH1,#FDH    ;
                     timer overflow after 3 clocks =9600
                     BR
6  200C   D2 8E      SETB TR1 ; start timer
                     1
7  200E   30 99 FD  MORE:     JNB TI, $
                     ;POLL THE Transmit interrupt flag bit
8  2011   C2 99      CLR TI     ; CLEAR
                     THE BIT FOR FUTURE USE
9  2013   12 20 1A   CALL DATA  ; PRO-
                     GRAM GETS ONE BYTE FROM ram
10 2016   F5 99      MOV SBUF,A ; WRITE
                     TO TRANSMIT PORT
11 2018   80 F4      SJMP MORE
12 201A   E0  DATA:MOVX A, @DPTR ; GET
                     A BYTE FROM ram
13 201B   A3         INC DPTR
14 201C   22         RET
15
16 201D   .END
```

The above program can be either entered from the keyboard on the kit or downloaded via the PC, if the downloading program is already stored there at 2700H and is ready to use. For downloading, the program must be converted into binary form with the LINK151.exe and then to ASCII file with BIN4ASC.exe. The resultant .dat file from the output of the BIN4ASC program can be sent (using SE command) through XTALK for downloading.

This program, on execution by GO command from 2000H on the kit, starts transmitting junk characters! We see a host of junk characters on the XTALK screen. That is because we haven't written any data at the specified RAM address of 2200H. If we run the program after entering there ASCII codes like 30, 31, 32, 33,...41, 42,..., etc, it will write 012...AB... on the computer screen.

Sending characters from the kit's keyboard to the PC's screen. As another improvement in programming techniques, the following program uses the keyboard on our development kit to send data. So, the DATA calling routine in the previous program is now a call to the keyboard-ASCII routine.

The basic keyboard routine at 00C0H in the monitor program, as discussed earlier, was meant to look for only a single scan code. However, for this program, we have to check for the key release scan code of F0H and then scan once more to get the actual keypressed scan code. Additionally, we also convert the data received from the keyboard into its ASCII code by looking up the conversion table (described earlier).

The program is as follows:

```
1  2000  .ORG 2000H
2  2000  75 98 52     MOV SCON, #52    ; SERIAL PORT MODE 1
3  2003  75 89 20     MOV TMOD,#20H    ; TIMER 1 8 BIT AUTO RELOAD MODE
4  2006  75 8D FD     MOV TH1,#FDH     ; timer overflow after 3 clocks =9600 BR
5  2009  D2 8E        SETB TR1  ; start timer 1
6  200B  30 99 FD     MORE: JNB TI,$   ; POLL THE Transmit interrupt flag bit
7  200E  C2 99        CLR TI     ; CLEAR THE BIT FOR FUTURE USE
8  2010  12 20 17     CALL KEY_ASCI  ; PROGRAM GETS ONE BYTE FROM KEYBD
9  2013  F5 99        MOV SBUF,A ; WRITE TO TRANSMIT PORT
10 2015  80 F4        SJMP MORE
11 2017  12 00 C0     KEY_ASCI: CALL 00C0H ; KBD CALL IN MONITOR PROGRAM
12 201A  BF F0 FA     CJNE R7, #F0H, KEY_ASCI ; KEY RELEASE CODE GOT?
13 201D  12 00 C0     CALL 00C0H  ; AGAIN GET THE SCAN CODE FOR KEY PRESS
14 2020  BF FF 02     CJNE R7,#FFH,C1
15 2023  80 F2        SJMP KEY_ASCI  ; NOT A VALID KEY, SO SCAN AGAIN
16 2025  12 20 2A     C1:    CALL TABLE_LK ; TABLE FOR SCAN CODE TO ASCII CODE
17 2028  FF           MOV R7,A      ;SAVE IN REGISTER R7
18 2029  22           RET
19 202A  00           TABLE_LK:    NOP
20 202B  83           MOVC A,@A+PC ; READ CODE FROM THE LIST BELOW
21 202C  22           RET
22 202D  FF FF FF FF  TABL: DB -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, "9", """, -1, -1, -1, -1, -1, -1, -1, "q"
   2031  FF FF FF FF
   2035  FF FF FF FF
   2039  39 60 FF FF
   203D  FF FF FF FF
   2041  71
23
24 2042  31 FF FF FF  DB "1", -1, -1, -1, "z", "s", "a", "w", "2", -1, -1, "c", "x", "d", "e", "4"
   2046  7A 73 61 77
   204A  32 FF FF 63
   204E  78 64 65 34
25 2052  33 FF FF 20  DB "3", -1, -1," ", "v", "f", "t", "r", "5", -1, -1, "n", "b", "h", "g", "y"
   2056  76 66 74 72
   205A  35 FF FF 6E
   205E  62 68 67 79
26 2062  36 FF FF FF  DB "6", -1, -1, -1, "m", "j", "u", "7", "8", -1, -1, ",", "k", "i", "o", "0"
   2066  6D 6A 75 37
   206A  38 FF FF 2C
   206E  6B 69 6F 30
27 2072  39 FF FF 2E  DB "9", -1, -1, ".", "/", "l", ";", "p", "-", -1, -1, -1, """, -1, "[", "="
   2076  2F 6C 3B 70
   207A  2D FF FF FF
   207E  27 FF 5B 3D
28 2082  FF FF FF FF  DB -1, -1, -1, -1, 0dh, "]", "\", -1, -1, -1, -1, -1, -1, -1, -1
   2086  0D 5D 5C FF
   208A  FF FF FF FF
   208E  FF FF FF
29
30 2091              .END
```

The program is easily downloadable from the PC to the kit. With this program the message entered via the kit's keyboard appears on the XTALK screen.

Two-way communication between the kit and the PC (a terminal). The next step is two-way communication between the PC and the kit. What we type on the kit goes to the PC screen. (The same may be saved in a file since the XTALK can capture data and save in file.) Also, what we type on the PC keyboard is shown as a continuous moving display on the LCD module.

For simultaneous reception and transmission of data between the PC and the kit, the following modification to the previous keyboard program is required:

```
29  20D0  A2B2      B1: MOV C,P3.2
30  20D2  40 07     JC A1
31  20D4  7F 00     MOV R7,#0
32  20D6  7B 08     MOV R3,#8
33  20D8  02 00 08  JMP 0008H
34  20DB  30 98 F2  A1:    JNB 98H,B1
35  20DE  C2 98     CLR RI
36  20E0  E5 99     MOV A,SBUF
37  20E2  E0        MOVX A,@DPTR
38  20E3  12 00 9F  CALL LC_WR
39  20E6  01 D0     JMP B1
40  009F
```

In the two-way communication LCD terminal program given below, we first check for the timing (clock) bit from the keyboard. Instead of just looping and waiting for the clock's low transition, the program goes to check the receive bit from the serial interface during this waiting time. So, a jump to A1 takes place if no low clock is sensed.

We basically check whether any receive interrupt flag is set, meaning that data has been collected in the serial buffer. If no serial byte is waiting in the buffer, we jump back to B1 to look at the keyboard clock. Thus, when no key is pressed or no serial data is input via RXD pin, the program jumps back and forth between A1 and B1.

Thus, we take care of the key depression on the keyboard as well as the

received data, whichever and whenever it occurs. And if the receive data byte is ready at B1, the RI flag is set. We clear the same and read the SBUF in the subsequent instructions. The data received is displayed by a call to LCDWRITE routine.

The program then jumps back to the keyboard at B1. The LCDWRITE subroutine includes the counter for 40 characters in DDRAM of LCD module so that a continuous display is possible with left shifting of the display at each entry. This has been explained earlier. The complete program includes the COMMAND and basic LC_WRITE routines also, so it is self-contained with no need for the monitor routines to be called. It can be directly incorporated in a dedicated 89C51 internally for making an LCD computer terminal.

LCD TERMINAL PROGRAM

```
1    2000    .ORG 2000H
2    2000    90 24 00        MOV DPTR,#2400H
3    2003    75 98 52        MOV SCON,#52H
4    2006    75 89 20        MOV TMOD,#20H
5    2009    75 8D FD        MOV TH1,#FDH
6    200C    D2 8E           SETB TR1
7
8    200E    00              A1:     NOP
9    200F    00              NOP
10   2010    00              NOP
11   2011    7E 30           MOV R6,#30H
12   2013    7C 18           MOV R4,#18H         ;no. of char. till shift (40 -16)
13   2015    74 07           MOV A,#07H          ;LCD SHIFT LEFT MODE
14   2017    12 20 C9        CALL CMD
15   201A    74 90           MOV A,#90H          ;LCD ADDRESS AT LAST
                             POSITION
16   201C    12 20 C9        CALL CMD
17   201F    12 20 45        K1: CALL KBD_ASCI_REC
18   2022    30 99 FD        JNB 99H,$   ;KEY DATA GOT, CHECK TRANS
                             MIT FLAG BUSY
19   2025    C2 99           CLR 99H     ;CLEAR FLAG
20   2027    EF              MOV A,R7    ;GET DATA FROM KEYBOARD
21   2028    F5 99           MOV SBUF,A  ;TRANSMIT THE KEY ASCI CODE
22   202A    80 F3           SJMP   K1
23
24   202C    KBD_REC:
25
26   202C    A2 B2           B1:  MOV C,P3.2   ; LOOK FOR KEYBOARD
                                                CLOCK LOW
27   202E    40 07           JC  A21    ; IF NO CLOCK LOW, CHECK RE
                                                CEIVE DATA SERIAL
28   2030    7F 00           MOV R7,#0
29   2032    7B 08           MOV R3,#8
30   2034    02 20 FA        JMP KBD4  ;00C8H    ;KBD ROUTINE IN MONI
                             TOR
31   2037    30 98 F2        A21:  JNB 98H,B1   ;IF NO RECEIVE DATA GO
                                                TO POLL KBD
32   203A    C2 98           CLR RI         ;CLEAR RECEIVE INTERRUPT
                                                FLAG
33   203C    E5 99           MOV A,SBUF     ;PICK DATA INTO ACCUMULA
                             TOR
34   203E    F0              MOVX @DPTR,A ;WRITE RECD. DATA IN
                                                MEMORY
35   203F    A3              INC DPTR       ;  INCREMENT DATAADDRESS
36   2040    12 20 BC        CALL LCDWRITE    ;WRITE ASCII CODE RECD.
                                                IN LCD
37   2043    01 2C           JMP B1       ;GO TO KEYBOARD SCAN AGAIN
38
39   2045    KBD_ASCI_REC:
40   2045    11 2C           CALL KBD_REC
41   2047    BF F0 FB        CJNE R7,#F0H,KBD_ASCI_REC
42   204A    11 2C           CALL KBD_REC
43   204C    BF FF 02        CJNE R7,#FFH,C1
44   204F    80 F4           SJMP KBD_ASCI_REC
45   2051    12 20 56        C1:      CALL TABLE_LOOK
46
47   2054    FF              MOV R7,A
48   2055    22              RET
49   2056                    TABLE_LOOK:
50   2056    83              MOVC A,@A+PC
51   2057    22              RET
52   2058                    TABLE:
53   2058    FF FF FF FF     CODES:  DB -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,"9","“",-
1,-1,-1,-1,-1,-1,"q"
     205C    FF FF FF FF
     2060    FF FF FF FF
     2064    39 60 FF FF
     2068    FF FF FF FF
     206C    71
54
55   206D    31 FF FF FF     DB          “1”,-1,-1,-1,”z”,”s”,”a”,”w”,”2",-1,-
1,”c”,”x”,”d”,”e”,”4"
     2071    7A 73 61 77
     2075    32 FF FF 63
     2079    78 64 65 34
56   207D    33 FF FF 20     DB      “3”,-1,-1,”     “,”v”,”f”,”t”,”r”,”5",-1,-
1,”n”,”b”,”h”,”g”,”y”
     2081    76 66 74 72
     2085    35 FF FF 6E
     2089    62 68 67 79
57   208D    36 FF FF FF     DB “6”,-1,-1,-1,”m”,”j”,”u”,”7","8",-1,-1,”,”,”k”,”i”,”o”,”0"
     2091    6D 6A 75 37
     2095    38 FF FF 2C
     2099    6B 69 6F 30
58   209D    39 FF FF 2E     DB “9”,-1,-1,”.”,”/”,”l”,”;”,”p”,”-”,-1,-1,-1,””,-1,”[“,”=”
     20A1    2F 6C 3B 70
     20A5    2D FF FF FF
     20A9    27 FF 5B 3D
59   20AD    FF FF FF FF DB  -1,-1,-1,-1,0dh,”]”,”\”,-1,-1,-1,-1,-1,-1,-1,-1
     20B1    0D 5D 5C FF
     20B5    FF FF FF FF
     20B9    FF FF FF
60
61   20BC                    LCDWRITE:
62   20BC    12 20 D8        CALL LC_WRITE
63   20BF    DC 07           DJNZ R4,RET1
64   20C1    7C 28           MOV R4,#28H
65   20C3    74 80           MOV A,#80H    ;SET ADDRESS OF LCD AS FIRST
66   20C5    12 20 C9        CALL CMD  ;WRITE COMMAND TO LCD MOD
                                                ULE
67   20C8    22              RET1:       RET
68
69   20C9    C0 83           COMMAND:      PUSH DPH
70   20CB    C0 82           PUSH DPL
71   20CD    90 80 00        MOV DPTR,#8000H
72   20D0    F0              MOVX @DPTR,A
73   20D1    31 25           ACALL DELAY
74   20D3    D0 82           POP DPL
75   20D5    D0 83           POP DPH
76   20D7    22              RET
77
78
79   20D8    C0 83           LC_WRITE:     PUSH DPH
80   20DA    C0 82           PUSH DPL
81   20DC    C0 E0           PUSH A
82   20DE    90 80 00        MOV DPTR,#8000H
83   20E1    E0              KC:      MOVX A,@DPTR
84   20E2    20 E7 FC        JB ACC.7,KC
85   20E5    D0 E0           POP A
86   20E7    90 80 01        MOV DPTR,#8001H
87   20EA    F0              MOVX @DPTR,A
88   20EB    31 25           ACALL DELAY
89   20ED    D0 82           POP DPL
90   20EF    D0 83           POP DPH
91   20F1    22              RET
92
93
94   20F2    7B 08           KBD:      MOV R3,#8
95   20F4    7F 00           MOV R7,#0
96   20F6    A2 B2           KP1:      MOV C,P3.2
97   20F8    40 FC           JC KP1
98   20FA    KBD4:
99   20FA    A2 B2           K4:   MOV C,P3.2
100  20FC    50 FC           JNC K4
101  20FE    A2 B2           K5:      MOV C,P3.2
102  2100    40 FC           JC K5
103  2102    A2 B4           MOV C,P3.4
104  2104    EF              MOV A,R7
105  2105    13              RRC A
106  2106    FF              MOV R7,A
107  2107    A2 B2           K6:      MOV C,P3.2
108  2109    50 FC           JNC K6
109  210B    DB F1           DJNZ R3,K5
110  210D    31 25           ACALL DELAY
111  210F    22              RET
112
113                          ;ORG E0H
114  2110                    INIT_LCD:
115  2110    74 38           MOV A,#38H
116  2112    11 C9           ACALL CMD
```

```
117  2114   74 0E          MOV A,#0EH
118  2116   11 C9          ACALL CMD
119  2118   74 06          MOV A,#06H
120  211A   11 C9          ACALL CMD
121  211C   74 80          MOV A,#80H
122  211E   11 C9          ACALL CMD
123  2120   74 01          MOV A,#1
124  2122   11 C9          ACALL CMD
125  2124   22             RET
126
127
128
129  2125   7D 80          DELAY:   MOV R5,#80H
130  2127   DD FE          DJNZ R5,$
131  2129   22             RET
132  212A                  END
```

After downloading the above program or entering it at 2000H, it can be run using 'G' key. Then, whatever is typed at the kit, goes to PC and whatever is typed on PC, appears on LCD screen. Of course, only lower case characters from kit, but both upper and lower case from PC can be seen on LCD screen.

APPENDIX 'B' : CAPACITANCE METER PROGRAM LISTING

```
APPENDIX 'B' Capacitance Meter Program Listing
                        1
                        2           $MOD52
4000                    3           ORG 4000H
                        4           ; ON ENTRY DPTR CONTAINS THE
VALUE OF THE TIME COUNT FOR CAP.CHARGING
4000 900000             5           MOV     DPTR,#0000
4003 7403               6           MOV     A,#03   ;
4005 F590               7           MOV     P1,A
4007 1242F0             8           LCALL   DLY
400A 7401               9            MOV     A,#01
400C F590               10          MOV     P1,A
400E 7441               11          MOV     A,#41H
4010 F590               12          MOV     P1,A
4012 A3                 13          KK:  INC     DPTR
4013 3090FC             14          JNB     P1.0,KK
4016 1243F7             15          CALL CHECK
4019 4009               16          JC NXTRNG
401B 124200             17          CALL MULPACK
401E 124313             18          CALL DISPFIL1
4021 024300             19          JMP DISPLAY_C
                        20
4024 900000             21          NXTRNG:MOV DPTR,#0000H
4027 7403               22          MOV A,#3
4029 F590               23          MOV P1,A
402B 1242F0             24          LCALL DLY
402E 7401               25          MOV A,#1
4030 F590               26          MOV P1,A
4032 7411               27          MOV A,#11H
4034 F590               28          MOV P1,A
4036 A3                 29          KK1:  INC DPTR
4037 3090FC             30          JNB P1.0,KK1
403A 1243F7             31          CALL CHECK  ; CHECKS IF TIME IS
LESS THAN 120 MICROSECS
403D 400B               32          JC NXTRNG2  ;GO TO NEXT RANGE
SELECT RESISTANCE
403F 124200             33          CALL MULPACK      ;MULTIPLY BY
CONSTANT AND PACK DIGITS
4042 1242C0             34          CALL DISPFIL2   ;FILL DIGITS  IN
ASCII CODE
4045 024300             35          JMP DISPLAY_C ;DISPLAY IT
4048 80FE               36          SJMP $
                        37
404A 1243C0             38          NXTRNG2:CALL INIT
404D 7409               39          MOV A,#09H
404F F590               40          MOV P1,A
4051 A3                 41          KK2:  INC DPTR
4052 3090FC             42          JNB P1.0,KK2
4055 1243F7             43          CALL CHECK
4058 400B               44          JC NXTRNG3      ;TOO LOW A CAP
VALUE, GO TO NEXT RANGE
405A 124200             45          CALL MULPACK
405D 124338             46          CALL DISPFIL3
4060 024300             47          JMP DISPLAY_C
4063 80FE               48          SJMP $
4065 904180             49          NXTRNG3:MOV DPTR,#MESLOW
4068 7401               50          MOV A,#1
406A 120090             51          CALL 0090H ;call command
406D 7C10               52          MOV R4,#10H
406F E0                 53          KS:  MOVX A,@DPTR
4070 12009F             54          CALL 009FH ;LC WR DDRAM
4073 A3                 55          INC DPTR
4074 DCF9               56          DJNZ R4,KS
4076 80FE               57          SJMP $
4180                    58          ORG 4180H
4180 4F50454E           59          MESLOW:                   DB
4FH,50H,45H,4EH,20H,20H
4184 2020
4186 43204F52           60          DB
43H,20H,4FH,52H,3CH,31H,30H,30H,70H,46H,46
418A 3C313030
418E 70462E
4200                    61          ORG 4200H
4200 7400               62          MULPACK:MOV     A,#00
4202 F540               63          MOV     40H,A
4204 F541               64          MOV     41H,A
4206 F542               65          MOV     42H,A
4208 0583               66          INC     DPH
420A 858230             67          MOV     30H,DPL  ;USE TEMP REGIS-
TERS 30,31 TO MOVE THE VALUE
420D 858331             68          MOV     31H,DPH  ;MULTIPLICATION
BY THE VALUE OF  17 WITH THE TIME COUNT (1.44*12 us
                                    )
4210 D53005             69          K:   DJNZ 30H,A1
4213 124250             70          LCALL ADD1
4216 8005               71          SJMP B1
4218                    72          A1:
4218 124250             73          LCALL ADD1
421B 80F3               74          SJMP K
421D D531F0             75          B1:   DJNZ 31H,K
4220 805E               76          SJMP PACK_ASCII
4222 08                 77          INC R0
4223 E0                 78          MOVX A,@DPTR
4224 A3                 79          INC DPTR
4225 742E               80          MOV A,#02EH
4227 F0                 81          MOVX @DPTR,A
4228 A3                 82          INC DPTR
4229 E6                 83          MOV A,@R0
422A F0                 84          MOVX @DPTR,A
422B A3                 85          INC DPTR
422C 08                 86          INC R0
422D B838ED             87          CJNE R0,#56,B1
4250                    88          ORG 4250H
4250                    89          ADD1:
4250 E540               90          MOV A,40H
4252 2406               91          ADD A,#06
4254 D4                 92          DA A
4255 F540               93          MOV 40H,A
4257 E541               94          MOV A,41H
4259 3400               95          ADDC A,#00
425B D4                 96          DA A
425C F541               97          MOV 41H,A
425E E542               98          MOV A,42H
4260 3400               99          ADDC A,#00
4262 D4                 100         DA A
4263 F542               101         MOV 42H,A
4265 22                 102         RET
4280                    103         ORG 4280H
4280                    104         PACK_ASCII:
4280 7842               105         MOV  R0,#42H   ; FROM LOCATIONS
42,41,40
4282 7950               106         MOV R1,#50H
4284 E6                 107         KD:  MOV A,@R0
4285 C4                 108         SWAP A
4286 540F               109         ANL A,#0FH
4288 4430               110         ORL A,#30H
428A F7                 111         MOV @R1,A
428B E6                 112         MOV A,@R0
428C 540F               113         ANL A,#0FH
428E 4430               114         ORL A,#30H
4290 09                 115         INC R1
4291 F7                 116         MOV @R1,A
4292 09                 117         INC R1
4293 18                 118         DEC R0
4294 B83FED             119         CJNE R0,#03FH,KD
4297 22                 120         RET   ;   SJMP DISP_FILL
                        121         ; NOW THE ASCII CODES OF THE
DIGITS ARE IN 50,51,.... 57H
42C0                    122         ORG 42C0H
42C0                    123         DISPFIL2:
42C0 7850               124         MOV   R0,#50H   ;POINT TO ASCCI
CODE AREA
42C2 9020F0             125         MOV     DPTR,#20F0H
42C5 E6                 126         MOV     A,@R0
42C6 08                 127         INC     R0
42C7 F0                 128         MOVX    @DPTR,A
42C8 A3                 129         INC     DPTR
42C9 742E               130         MOV     A,#2EH
42CB F0                 131         MOVX    @DPTR,A
42CC A3                 132         INC     DPTR
42CD                    133         K1:
42CD E6                 134         MOV     A,@R0
42CE F0                 135         MOVX    @DPTR,A
42CF A3                 136         INC     DPTR
42D0 08                 137         INC     R0
42D1 B856F9             138         CJNE    R0,#56H,K1
42D4 74E4               139         MOV     A,#0E4H
42D6 F0                 140         MOVX    @DPTR,A
```

```
42D7 A3          141        INC    DPTR
42D8 7406        142        MOV    A,#406H  ;'F'
42DA F0          143        MOVX   @DPTR,A
42DB 22          144        HERE:  RET
                 145        ; NOW AREA 27F0H ONWARDS
HOLDS THE ASCII CODES
42F0             146        ORG 42F0H
42F0             147        DLY:
42F0 7EFF        148        MOV    R6,#0FFH
42F2 00          149        L1:    NOP
42F3 00          150        NOP
42F4 00          151        NOP
42F5 00          152        NOP
42F6 DEFA        153        DJNZ   R6,L1
42F8 22          154        RET
4300             155        ORG 4300H
4300             156        DISPLAY_C:
4300 9020F0      157        MOV DPTR,#20F0H
4303 7401        158        MOV A,#01
4305 120090      159        CALL 0090H  ;call command
4308 7C0A        160        MOV R4,#0AH
430A E0          161        KD2:  MOVX A,@DPTR
430B 12009F      162        CALL 009FH  ;LC WR DDRAM
430E A3          163        INC DPTR
430F DCF9        164        DJNZ R4,KD2
4311 80FE        165        SJMP $
4313             166        ORG 4313H
4313             167        DISPFIL1:      ;(10k RES RANGE)
4313 7850        168        MOV R0,#50H  ;POINT TO DECIMAL
DIGITS AFTER PACKING
4315 9020F0      169        MOV DPTR,#20F0H ;POINT TO AREA
BUFFER FOR LCD MESS DISPLAY
4318 E6          170        MOV A,@R0
4319 08          171        INC R0
431A B43002      172        CJNE  A,#30H,LEADZER   ;LEADING
ZERO SUPPRSSION
431D 7420        173        MOV A,#20H ; SPACE
431F             174        LEADZER:
431F F0          175        MOVX @DPTR,A
4320 A3          176        INC DPTR
4321 E6          177        MOV A,@R0
4322 08          178        INC R0
4323 F0          179        MOVX @DPTR,A
4324 A3          180        INC DPTR
4325 742E        181        MOV A,#2EH  ;DEC. PT
4327 F0          182        MOVX @DPTR,A
4328 A3          183        INC DPTR
4329 E6          184        KA1:   MOV A,@R0
432A F0          185        MOVX @DPTR,A
432B A3          186        INC DPTR
432C 08          187        INC R0
432D B856F9      188        CJNE R0,#56H,KA1
4330 74E4        189        MF:   MOV A,#0E4H
4332 F0          190        MOVX @DPTR,A
4333 A3          191        INC DPTR
4334 7446        192        MOV A,#46H
4336 F0          193        MOVX @DPTR,A
4337 22          194        RET
4338             195        ORG 4338H
4338 7850        196        DISPFIL3: MOV  R0,#50H  ;1 m RES
RANGE
433A 9020F0      197        MOV DPTR,#20F0H
433D 7430        198        MOV A,#30H
433F F0          199        MOVX @DPTR,A
4340 A3          200        INC DPTR
4341 742E        201        MOV A,#2EH
4343 F0          202        MOVX @DPTR,A
4344 A3          203        INC DPTR
4345 E6          204        KA2:  MOV A,@R0
4346 F0          205        MOVX @DPTR,A
4347 A3          206        INC DPTR
4348 08          207        INC R0
4349 B856F9      208        CJNE R0,#56H,KA2
434C 6130        209        AJMP MF
                 210
                 211
43C0             212        ORG 43C0H
43C0 900000      213        INIT:       MOV       DPTR,#0000H
;INITIALISE CAPACITANCE PROBE
43C3 7403        214        MOV A,#3       ;DISCHARGE CAP.
43C5 F590        215        MOV P1,A
43C7 1242F0      216        LCALL DLY      ;WAIT
43CA 7401        217        MOV A,#1     ;LET IT CHARGE
43CC F590        218        MOV P1,A
43CE 22          219        RET
43F7             220        ORG 43F7H
43F7             221        CHECK:         ;CHECK FOR UNDER
RANGE
43F7 E583        222        MOV A,DPH
43F9 B40004      223        CJNE A,#0,AA1
43FC E582        224        MOV A,DPL
43FE 9410        225        SUBB A,#10H    ;COUNT LESS THAN
16 MEANS UNDER RANGE
4400 22          226        AA1:  RET
                 227
                 228        END
                 229
                 230
                 231
                 232
                 233
                 234
```

VERSION 1.2k ASSEMBLY COMPLETE, 0 ERRORS FOUND

```
A1 ..... C ADDR   4218H
AA1 ..... C ADDR   4400H
ADD1 ..... C ADDR   4250H
B1 ..... C ADDR   421DH
CHECK ..... C ADDR   43F7H
DISPFIL1 ..... C ADDR   4313H
DISPFIL2 ..... C ADDR   42C0H
DISPFIL3 ..... C ADDR   4338H
DISPLAY_C ..... C ADDR   4300H
DLY ..... C ADDR   42F0H
DPH ..... D ADDR   0083H   PREDEFINED
DPL ..... D ADDR   0082H   PREDEFINED
HERE ..... C ADDR   42DBH   NOT USED
INIT ..... C ADDR   43C0H
K ..... C ADDR   4210H
K1 ..... C ADDR   42CDH
KA1 ..... C ADDR   4329H
KA2 ..... C ADDR   4345H
KD ..... C ADDR   4284H
KD2 ..... C ADDR   430AH
KK ..... C ADDR   4012H
KK1 ..... C ADDR   4036H
KK2 ..... C ADDR   4051H
KS ..... C ADDR   406FH
L1 ..... C ADDR   42F2H
LEADZER ..... C ADDR   431FH
MESLOW ..... C ADDR   4180H
MF ..... C ADDR   4330H
MULPACK ..... C ADDR   4200H
NXTRNG ..... C ADDR   4024H
NXTRNG2 ..... C ADDR   404AH
NXTRNG3 ..... C ADDR   4065H
P1 ..... D ADDR   0090H   PREDEFINED
PACK_ASCII ..... C ADDR   4280H
```

❑

Now let's go through some more applications of the development kit using its inbuilt timers and interrupts.

Programming of timers

Each of the two timers in the 8051 family of microcontrollers is separately usable. The clock for each time count is 1 μs with the standard 12MHz crystal. (With 4MHz crystal, it would be 3 ms and so on proportionately.) The timer counts the clock pulses (of 1 μs each) up to the full 16-bit count value of 65,535 counts and then it overflows. When it overflows, it will inform such an overflow in two ways:

1. Bit 7 of the TCON register (88H) is set to logic 1 for timer 1 overflow, while for timer 0 overflow, bit 5 of the TCON register will be set to logic 1. Bit-wise addresses and related signals of TCON register are given below:

TCON Register

Bit.	D7	D6	D5	D4	D3	D2	D1	D0
Address.	8F	8E	8D	8C	8B	8A	89	88
Signal.	TF1	TR1	TF0	TR0	IE1	IT1	IE-0	IT-0

2. If the interrupt for the timer is enabled, the program to that timer's interrupt service routine (ISR) is entered. The vector for timer 0 interrupt is having the base address as 000BH, while for timer 1, it is 001BH. These locations are usually in the internal Flash EEROM of the 89C51. At these locations, we load jump addresses (vectors) for the actual ISR in the user's program memory area. Thus if locations starting at 000BH have 02 2F 2B and that starting at 001BH have 02 2F 8B, it means that timer 0's ISR begins at address 2F2BH and timer 1's ISR begins at 2F8BH.

You have to write a program segment from 2F2B onwards, indicating action to be performed when timer 0 overflows, i.e. at the end of the predetermined time, the action to be performed; for example, switch on a relay at output P1.0 by using

TABLE I
Timer/Counter Mode Control Register Bit Functions

TMOD ADDRESS = 89H
NOT BIT ADDRESSABLE
RESET VALUE = 00H

7	6	5	4	3	2	1	0
GATE	C/T	M1	M0	GATE	C/T	M1	M0
TIMER 1				TIMER 0			

GATE GATING CONTROL WHEN SET. TIMER/COUNTER 'x' IS ENABLED ONLY WHILE 'INT$_x$' PIN IS HIGH AND 'TRx' PIN IS SET. WHEN CLEARED TIMER 'x' IS ENABLED WHEN EVER 'TRx' CONTROL BIT IS SET.

C/T TIMER OR COUNTER SELECTOR DEARED FOR TIMER OPERATION (INPUT FROM INTERMINAL SYSTEM CLOCK.)

M1	M0	OPERATING
0	0	8048 TIMER 'TLx' SERVERS AS 5-BIT PRESCALER.
0	1	16-BIT TIMER/CONTER 'THx' AND 'TLx' ARE CASCADED; THERE IS NO PRESCALER.
1	0	8-BIT AUTO-RELOAD TIMER/COUNTER 'THx' HOLDS A VALUE WHICH IS TO BE RELOADED INTO 'TLx' EACH TIME IT OVERFLOWS.
1	1	(TIMER 0) TL0 IS AN 8-BIT TIMER/COUNTER CONTROLLED BY THE STANDARD TIMER 0 CONTROL BITS. TH0 IS AN 8-BIT TIMER ONLY CONTROLLED TIMER 1 CONTROL BITS.
1	1	(TIMER 1) TIMER/COUNTER 1 STOPPED.

a SETB P1.0 instruction. The ISR terminates with a RETI (return-from-interrupt) instruction.

For putting a timer into operation, there is a set sequence of steps. First, we calculate the number to be loaded into the 16-bit timer register; for example, if it is loaded with 60,000 (decimal), it will keep counting 1μs pulses (starting with count of 60,000) until it reaches terminal count of 65,535. That means an additional time of 5536 μs will be spent until the timer register overflows. So, a timer value of 5.536 ms has been obtained by the timer.

Suppose we want a square wave having a time period of 11.072 ms, then after each overflow of the 16-bit timer register, we enter its ISR. There, we can write an instruction that toggles the port P1.0 bit. That code would be like:

```
Code     Mnemonic   Comments
B2 90    CPL P1.0   ;toggle bit 0 of port 1
```

B2 is the code for complementing a direct bit; here, the direct bit is P1.0, which is 90H.

Note that 60,000 (EA60H) must be entered in hexadecimal as a 16-bit number into the timer's register pair of TH0 and TL0, which are timer 0 high- and low-byte count registers.

So, TH0 must be loaded with EA Hex and TL0 with 60H. The required instructions are:

```
MOV    TH0,#EAH
MOV    TL0,#60H
```

Then timer 0 is started by setting a 'Start Timer 0' bit in the internal register TCON.4. Likewise, TCON.6 controls timer 1. Setting TCON.6 starts timer 1. TCON.4 and TCON.6 are to be found at internal RAM at bit addresses 8BH (i.e. 88+04) and 8DH, respectively.

The above procedure is for operating the timer as a 16-bit timer. But the timers have four modes of operation as per the state of M1 and M0 bits in TMOD register. The functions of various bits in TMOD register are shown in Table I.

Fig. 10 shows the logic states of the various control bits on a timer start action. Gate bit in the TMOD register is low if INT0 pin is not used in a timer function. If Gate bit is high, INT0 pin must be high for enabling the timer and the TR0 (setB TR0) will start timer 0. Note that Gate and TR0 are software bits, while INT0 and T0 are actual pins 12 and 14 on the 89C51.

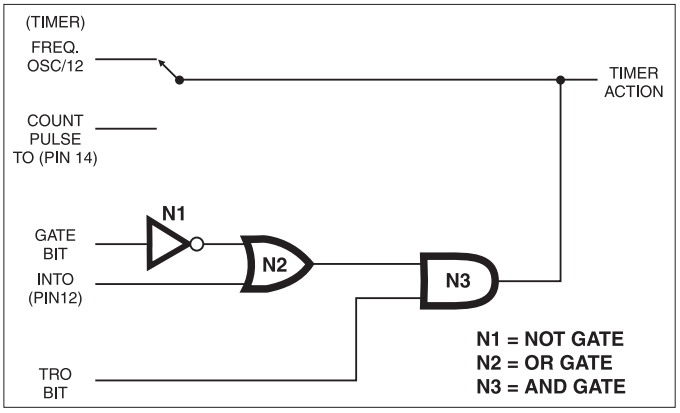


Fig. 10: Logic states of various control bits on a timer start action

Some examples of timer action

Example 1: 1ms timer for port A of 8255 on-board LEDs of Fig. 3. The program for a 1ms to 16-second counter is as follows:

```
1  2000              ORG 2000H
2  2000  90 60 03    MOV DPTR,#6003H
                     ;POINT TO 8255 CONTROL REG.
3  2003 74 80        MOV A,#80H ;TO SET
                     ALL PORTS AS OUTPUT
4  2005  F0          MOVX @DPTR,A
                     ;OUTPUT TO CONTROL REG.
5  2006  7D 01       MOV R5,#01
6  2008  75 89 29 A1: MOV TMOD,
                     #00101001B
                     ;TIMER 0 AS 16 BIT (01)

                     NON GATED
7                    ;TIMER 1 BE GATED
                     COUNTER IN MODE 2
8                    ;HERE TIMER 0 IS ALONE
                     USED.
9  200B  75 8C FC    MOV TH0,#FCH
                     ;FOR 1 ms, 3E8H subtract from
                     1000H = FC18H
10 200E  75 8A 18 MOV TL0,#18H
11 2011  43 A8 82 ORL IE,#82H
          ;ENABLE INTERRUPT FOR TIMER 0
12                   ;INT. ENABLE BIT D7
                     HIGH,D1 BIT HIGH FOR ET-0
13 2014  D2 8C       SETB TR0
                     ;START TIMER 0
14 2016  C2 8D    K: CLR TF0
                      ;CLEAR TIMER -0 OVERFLOW FLG
15 2018  80 FE       SJMP $ ;LOOP HERE
16
17 201A  01 08       AJMP A1
                     ;AFTER INTERRUPT
18
19 2F2B  ORG 2F2BH ;JUMP VECTOR for
         timer 0 interrupt
20 2F2B  ISR:
21 2F2B  0D          INC R5  ;Increment R5
22 2F2C  ED          MOV A,R5  ;move to A
23 2F2D  90 60 00    MOV DPTR,#6000H
                     ; POINT TO 8255 PORT A
24 2F30  F0          MOVX @DPTR,A
                     ; OUTPUT THERE
25 2F31  F5 90       MOV P1,A
          ;CHECKING PORT-1 IF 8255 NOT USED
26 2F33  32          RETI
                     ; Then return from ISR.
27
28 2F34  END
```

The program has to be loaded at two stretches, i.e. from 2000H to 201BH for the main program and from 2F2BH to 2F33H for timer 0 ISR. If you are assembling and downloading the above program after linking and ASCII-conversion into the file in a PC, you must wait for this program to download from 2000H to 2F34H after W key (downloading key) is pressed. Otherwise, the downloading up to 2F high address would not be completed.

By connecting an LED or logic probe at pins 1 to 8 of the IC 89C51, it is possible to see the blinking rate of the LED. Try changing the values of TH0 and TL0 from FC18 hex to 0000H. Then we get 64 ms for each overflow. That would make the last D7 bit to blink at a rate of 256x64 ms or 16 seconds.

The program uses the timer to generate 1ms timing. Register R5 increments every 1 ms. Thus, the LED on port 1 (or port A) will blink from 1 to 255 ms for its eight bits. If the D7 bit alone is observed, it will blink at 256x1x2 ms or half a second rate.

Example 2: Pulse variable control. Here is another program which generates pulses of 15 kHz (64 μs) with gaps in between. This is a pulse-train modulated (not pulse width modulated) signal. The number of pulses in a group of 16 pulses is varied from 1 to 10. (A thumbwheel switch connected to port P1.4 (pin 5) through port P1.7 (pin 8) will vary the pulses according to the number shown by thumbwheel switch.)

The remaining periods of 64 μs are free from pulses. These pulse trains are useful for small-model control motor drives. Here we need a short timing duration of 32 μs for the two half cycles of the rectangular pulse of 64μs duration. As 32 is a small number, we can use an 8-bit timer. In mode 2, auto-reloading function is available, so we need not enter the values in the timer register every time, as we did in the previous program (by jumping back to A1 point). Thus timer 0 is set in mode 2. The value of E0H corresponds to –32 (E0H+20H=100 H).

The ISR is called TIM0int. A flag bit called 'flag' is used here. If this flag is set, pin P1.0 is set to low logic. The bit is complemented at every timer interrupt, but the number of pulses to be output is limited. The reading of bits D4 to D7 from port 1 decides the number of pulses in a group of 16 time slots of 64 μs. Thus, in 64×16 μs (=1 ms), the number of output pulses is decided by this number read from port 1 bits 4 to 7.

The program is as follows:

```
1   00 30    VALUE1   EQU 30H
2   00 31    VALUE2   EQU 31H
3   00 32    FLAG     EQU 32H
4
5
6
7   2000     ORG      2000H
8   2000     MONI:
9   2000     75 81 70 ST:MOV SP,#70H
10  2003     75 B0 FF MOV P3,#FFH
11  2006     75 8A 00 MOV TL0,#00
12  2009     75 8C E0 MOV TH0,#E0H
13  200C     75 89 02 MOV TMOD,#02 ;8 BIT
                      TIMER-0 th AUTO RELOAD
14  200F 43 A8 82 ORL IE,#82H ;TIMER-0
    BIT ENABLES INTERRUPT
15  2012 7F 18    MOV R7,#24
16  2014 E5 90    KK:MOV A,P1
17  2016 54 F0    ANL A,#F0H
18  2018 C4       SWAP A
19  2019 F5 30    MOV 30H,A
20  201B 74 10    MOV A,#16
21  201D 95 30    SUBB A,30H
22  201F F5 31    MOV 31H,A
23  2021 D2 8C    SETB TR0  ; START
TIMER
24  2023 01 14    JMP KK
25
26
27  2080 ORG 2080H
28
29  2080 20 32 0C TIM0INT:JB FLAG,K1
30  2083 DF 07    DJNZ R7,K2
31  2085 D2 32    SETB FLAG
32  2087 AF 30    MOV R7,30H
33  2089 B2 90    CPL P1.0
34  208B 32       RETI
35  208C C2 90    K2:CLR P1.0
36  208E 32       RETI
37  208F DF 07    K1:DJNZ R7,K3
38  2091 C2 32    CLR FLAG
39  2093 AF 31    MOV R7,31H
40  2095 C2 90    CLR P1.0
41  2097 32       RETI
42  2098 B2 90    K3:CPL P1.0
43  209A 32       RETI
44
45  209B          END
```

Also add by hand, at 2F2BH address, the code:

```
2F2B  92 20 80   JMP 2080H
```

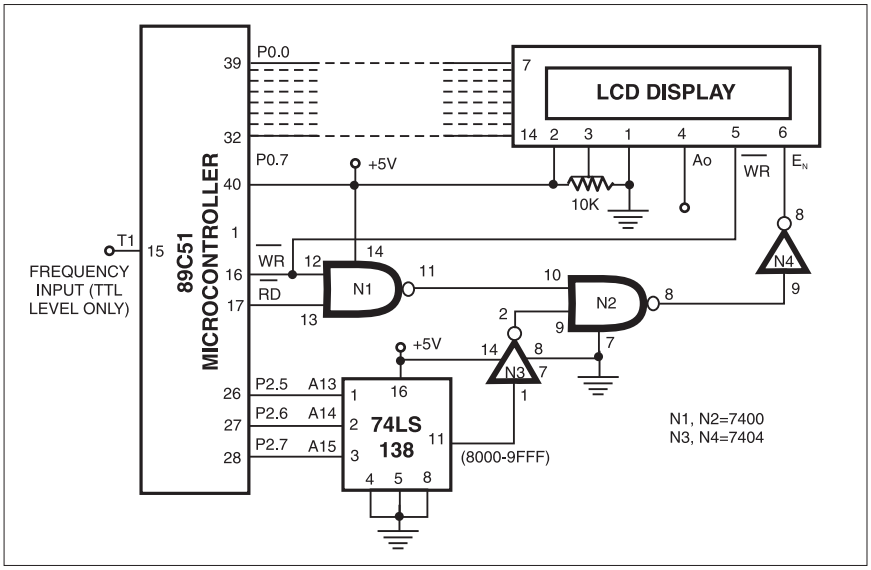


Fig. 11: Frequency counter on the 89C51 circuit

Example 3: Frequency counter program (Fig. 11). The 89C51 has two timers or counters. If one timer is configured to repeat every 1 ms, the other timer can be made to act as a counter. The counts that would accumulate during this period (1 ms) will then indicate the frequency of the signal in kilohertz. If the timer's window is varied to 10 ms, the count frequency will be in hundreds of hertz, and if it is varied to 100 ms, we get frequencies in tens of hertz, and so on. By varying this time, one can make an auto-ranging frequency counter that is basically useful for a range of 100 Hz to 20 kHz.

The counter pin must be fed with clean TTL pulses of the frequency being measured. In digital circuits this is easy. However, any analogue signal must be first converted into TTL (5V) level before it is input to the T1 (counter) pin 15 of the 89C51.

The program starts with the initialisation of the timer and the counter. Timer 0 has the timing window. Timer 1 acts as a counter, with input to T1 (pin 15).

When the time is over, which is checked by flag TF, the value of the count in TL is passed on to the LCD. The number in TL register is in binary, which is first converted into decimal form by a call to the subroutine BINBCD.

The BINBCD program is useful for converting any binary 8-bit number into BCD format. Thus FF Hex will become 255. The program uses the special instruction DIV AB to divide first by 100 and then by 10 to find the BCD digits one by one. Then, the three digits are to be displayed on the LCD. Since the LCD requires ASCII coded numbers for display, we convert 2 to 32, 0 to 30, and so on. Then, a three-digit number is displayed on the LCD.

The program can be changed for a gate period of 10 ms or 100 ms by simply altering the initial values of timer registers TL and TH in 4th and 5th lines of the program. For entering, the number 1000 for 1 ms (1 ms=1000 μs=1000 pulses), the preload value is FC18. (This is for a 12MHz crystal. For an 11.09MHz crystal, the number will be smaller by 11.09/12. This gives a value of 924 for 1 ms, which is converted into hex value (39CH) and subtracted from FFFFH to get the preload value of FC64H. FC64 is to be loaded in TH0 and TL0 in lines 4 and 5, respectively.)

The program is as follows:
FREQCTR.ASM

```
1  $MOD52
2  2000     ORG 2000H
3  2000 758951  BEG: MOV TMOD, #01010001B
                ;TIMER 1 - MODE 1 16 BIT, TIMER-
                0 COUNTER MODE 1
4  2003 758CD8 MOV TH0,#0D8H  ;TIMER
                REG.0 IS SET TO 0
5  2006  758AFE     MOV  TL0,#0FEH   ;10  ms
                    GATE TIME
6  2009 D28C   SETB TR0  ;START TIMER 0
                    (COUNT ACTION)
7  200B     758B00    MOV TL1,#00H
8  200E 758D00   MOV TH1,#00H ;ctr-1 IS
                    START-ING FROM -1
9  2011 D28E   SETB TR1 ; START COUNTER
                    TIMER-1
10 2013     308DFD  JNB TF0,$
                    ; COUNT OVER ?
11 2016     C28E      CLR TR1      ; STOP TIM- 1
12 2018     C28D      CLR TF0
13 201A     C28C      CLR TR0   ;STOP TIMER-0
14 201C     E58B      MOV A,TL1 ;read counts
15 201E     122025    CALL SHOW ; SHOWS
           THE COUNTS IN 1 MILLISEC
16 2021     117A      ACALL DELAY
17 2023     0100      AJMP BEG   ;over
18 2025     12205D    SHOW:CALL BINBCD
   ;RESULT IN DECIMAL IS IN
        ;INTERNAL RAM LOCATIONS 30-32 19
2028        7401      MOV A,#1
20 202A     120090    CALL 90H ;COMMAND
                      TO Lcd
21 202D     E530      MOV A,30H
22 202F     122070    CALL HEXASCI
23 2032     12009F    CALL 9FH
24 2035     12009F    CALL 09FH  ;Write
                      DDRAM of LCD
25 2038     E531      MOV A,31H
26 203A     122070    CALL HEXASCI
27 203D     12009F    CALL 09FH
28 2040     E532      MOV A,32H
29 2042     122070    CALL HEXASCI
30 2045     12009F    CALL 09FH
31 2048     7420      MOV A,#20H
32 204A     12009F    CALL 09FH
33 204D     744B      MOV A,#'K'
34 204F     12009F    CALL 09FH
35 2052     7448      MOV A,#'H'
36 2054     12009F    CALL 09FH
37 2057     745A      MOV A,#'Z'
38 2059     12009F    CALL 09FH
39 205C 22RET
40 205D     75F064    BINBCD:MOV B,#64H
41 2060     84        DIV AB
42 2061     F530      MOV 30H,A
43 2063     E5F0      MOV A,B
44 2065     75F00A    MOV B,#0AH
45 2068     84        DIV AB
46 2069     F531      MOV 31H,A
47 206B     E5F0      MOV A,B
48 206D     F532      MOV 32H,A
49 206F     22        RET
50 2070     HEXASCI:  ;CONVERTS A NIBBLE
          INTO ASCII CODE FOR LCD DISPLAY
51 2070     2436      ADD A,#36H
52 2072     30D602    JNB AC,SKIP
53 2075     2407      ADD A,#7
54 2077     9406      SKIP:SUBB A,#6
55 2079     22        RET
56 207A     7FFF      DELAY:MOV R7,#0FFH
57 207C     7E80      S1:MOV R6,#80H
58 207E     DEFE      DJNZ R6,$
59 2080     DFFA      DJNZ R7,S1
60 2082     22        RET
END
```

Note that this program uses only the polling of the timer flag. It doesn't use the interrupt action on timer overflow, as we have not enabled the interrupts. (The program works properly in kHz.)

Example 4: Period measurement. Signals (of course, TTL input only) of low frequency cannot be measured by using a time gate/window. In such cases the period itself can be measured to an accuracy of 1 μs. For period measurement, we can load timer 0 as above with the value '0' to start with and then start it along with counter 1. Load the timer mode register TMOD with a value of 61H. This gives mode 2 for timer 1 acting as counter of pulse transitions from high level to low level (at pin 15 of 89C51). Timer 1 functions as an 8-bit auto-reload type of counter.

In the program for period measurement, we load counter 1 register with maximum count value of FFH, because we want even the very first transition to cause the register to overflow from FF to 100, which means overflow beyond eight bits. Then, we check TF1 flag (in line 12), which gets set internally. So, if we read that TF1 bit, we know the transition has occurred. Then, we start timer 0 and count the microseconds in its TH0 and TL0 registers, which are initially set to zero (in 4th and 5th lines, respectively).

Again we start counter 1. When the next transition occurs, one period is counted on timer 0. When the count is one, we can note the timer 0 register and see the value in microseconds. (With an 11.09MHz crystal, each count is 12/11.09 μs.) But this is a 16-bit number. We have no easy method to convert it into decimal (<65,536). So, for the present we just see the value in hex itself, using the monitor LCD routine. That routine shows DPH and DPL registers. So, after transferring the timer 0 low and high registers to these, the display program is called. The program repeats indefinitely, so the period of a TTL-compatible waveform can be continuously observed.

Pin 15 of 89C51 is connected to the pulse output from, say, a function generator. Then, after executing the program, we read the value on LCD. Let it be 7 09 Hex, giving a value of 1949 μs.

The program is as follows:

```
INPUT FILENAME : PERCTR.ASM
1  2000     ORG 2000H
2
3  2000     75 89 61   MOV  TMOD,#01100001B
;TIMER 1-MODE 2 COUNTER,TIMR-0 TO MODE 1
4  2003     75 8C 00   BEG:MOV TH0,#0H
                       ;TIMER REG.0 IS SET TO 0
5  2006     75 8A 00   MOV TL0,#0H
6  2009     75 8B FE   MOV TL1,#FEH
7  200C     75 8D FF   MOV TH1,#FFH ; ctr-1 IS
```

TABLE II
Interrupt-Enable Register

IE — ADDRESS = 0A8H, BIT ADDRESSABLE — RESET VALUE = 0XO000OOB

7	6	5	4	3	2	1	0
EA	—	ET2	ES	ET1	EX1	ET0	EX0

ENABLE BIT = 1 ENABLES THE INTERRUPT
ENABLE BIT = 0 DISABLES IT

BIT	SYMBOL	FUNCTION
IE.7	EA	GLOBAL DISABLE BIT. IF EA = 0, ALL INTERRUPTS ARE DISABLED. IF EA = 1, EACH INTERRUPT CAN BE INDIVIDUALLY ENABLED OR DISABLED BY SETTING OR CLEARING ITS ENABLE BIT.
IE.6	–	NOT IMPLEMENTED. REVERSED FOR FUTURE USE.
IE.5	ET2	TIMER 2 INTERRUPT ENABLE BIT.
IE.4	ES	SERIAL PORT INTERRUPT ENABLE BIT.
IE.3	ET1	TIMER 1 INTERRUPT ENABLE BIT.
IE.2	EX1	EXTERNAL INTERRUPT 1 ENABLE BIT.
IE.1	ET0	TIMER 0 INTERRUPT ENABLE BIT.
IE.0	EX0	EXTERNAL INTERRUPT 0 ENABLE BIT.

```
                        STARTING FROM -1  so
                        that the first ;high to low
                        transition on T1 pin 15 will
                        cause over-flow, set TF1
9   200F   C2 8F        CLR TF1 ; so clear the flag
                        to start with
10
11 2011    D2 8E        SETB TR1 ; START
                        COUNTER -1
12 2013    30 8F FD     JNB TF1,$; COUNT OVER
           means edge low on input detected
13 2016    D2 8C        SETB TR0 ;START TIMER 0
14 2018    C2 8F        CLR TF1 ;CLEAR INTR.
                        FLAG of counter 1
15                      ;now start timer and count
                      microseconds till next low edge
16 201A    30 8F FD     JNB TF1,$
17 201D    C2 8C  CLR TR0 ;STOP TIMER-0 NOW
18 201F    85 8A 82     MOV DPL,TL0 ;read counts
19 2022    85 8C 83     MOV DPH,TH0 ; TO data
                        pointer as 16 bit number
20 2025    12 01 30     CALL  0130H  ;  DISPLAY
PROGRAM FOR LCD SHOWS DPH,DPL AND R2
21 2028    11 2C        ACALL DELAY
22 202A    01 03        JMP BEG   ;over
23 202C    7F FF        DELAY:MOV R7,#FFH
24 202E    7E 80        S1:MOV R6,#80H
25 2030    DE FE        DJNZ R6,$
26 2032    DF FA        DJNZ R7,S1
27 2034    22           RET
28 2035                 END
```

Using interrupts with timer

By using the timer to cause an interrupt, we run the ISR routine. We have definite (vector) addresses as designed by the makers of the 89C51, wherein the jump addresses for ISRs used by us are loaded. The same are tabulated below:

Interrupt name	Vector address	Jump address
Reset	0000	....
External Int. 0	0003	2F03
External Int. 1	0013	2F63
Timer 0	000B	2F2B
Timer 1	001B	2F8B
Serial Int.	0023	2E00

For using timer 0, we have to write the program at the branch (jump) address location for the interrupt vector at 000BH. Inside the monitor program, we have provided jumping to RAM locations outside the 89C51's internal ROM area, so the user can write programs for interrupt action as desired by him.

Example 1: Timer operation. When we want long-duration timer, it is necessary to accumulate the time counts of each overflow of the timer. In the 2-minute and 3-minute timer operation example given below, there is a need to wait for two minutes for an operation; and an indication of extra one minute is also required.

The program using timer 0 is given below. Timer 0 is operated as a 16-bit timer. Note that 16-bit timers don't have auto-reload feature, so we have to reload the timer registers every time after timer overflow. Also, the interrupt for this timer must be enabled. This enabling can be done by the following instruction:

```
ORL IE, 82H .....(A)
```

where IE refers to the interrupt-enable register at SFR address A8H. (OR operation doesn't affect any previously set interrupt.)

This SFR has eight bits with start bit 0 address of A8H and bit 7 address of AFH as shown in Table II.

In order to enable timer 0, ET0 and also EA bits for any one interrupt are set to 1. The EA bit can be used to disable all the interrupts in critical program pathways, by making the AF bit a '0'.

In the instruction (A) shown above, 82H denotes that both EA and ET0 bita are set. Another way is to use the BIT instruction direct for each bit; for example, SETB EA (code as D2 AF) and SETB ET0 (code as D2 A9). The 89C51 assembler understands what EA and ET0 labels mean.

The interrupt vector for timer 0 jumps to location 2F2BH in external RAM. There we write an additional jump to 2080H using the keyboard, from 2F2BH to 2F2DH as under:

Address	Code	Instruction
2F 2B	02 20 80	JMP 2080H

This has to be done manually, because when we download the program from the PC, it will download only in page 20 (address 2000) and the program for the timer vectors to 2F. However, if one waits for a few seconds (after pressing the W key on the keyboard of the kit and sending the program from the PC), even locations up to 2F page will get downloaded.

The program is as follows:

```
 INPUT FILENAME : 1MINTIM.ASM
1  $MOD52
2  2000    ORG 2000H
3  2000    758961   MOV TMOD, #01100001B
                    ;Set Timer-1for counter
                    &Timer-0 for timer operation.
4  2003    758CF0   BEG:MOV TH0,#0F0H
                    ;Set Timer Reg.0 to F000H,
5  2006    758A00   MOV TL0,#0H ;(gives 4ms
                    with 12MHz crystal)
6  2009    7EFF     MOV R6,#0FFH ;Interrupt
                    count location.
7  200B    D28C     SETB TR0 ;Start Timer-0.
8  200D    D2A9     SETB ET0 ;Enable Timer-
                    0 Interrupt.
9  200F    D2AF     SETB EA; Enable
                    Interrupt Global.
10 2011    7401     MOV A,#01H
11 2013    120090   CALL 90H ;To clear LCD
12 2016    7430     MOV A,#30H
13 2018    F542     MOV 42H,A ;"0"To Start
                    with secs ctr.
14 201A    12009F   CALL 09FH
15 201D    7A00     MOV R2,#0
16 201F    C28D     CLR TF0
17 2021    80FE     SJMP $
18
19 2080             ORG 2080H
20 2080    758CF2   TIM0ISR:MOV TH0,#0F2 H
21 2083    DE09     DJNZ R6,K1 ;256 Times
                      4ms gives 1 sec.
22 2085    0A       INC R2
23 2086    BA7805   CJNE R2,#078H,K1 ;120
                    sec = 2 minute.
24 2089    7432     MOV A,#32H
```

```
25 208B   12009F   CALL 9FH ;Write 1 to
                   Display.
26 208E   BAB405   K1:CJNE R2,#0B4H,K1A
                   ;After 3 minutes, write 3.
27 2091   7433     MOV A,#33H
28 2093   12009F   CALL 9FH
29 2096   32       K1A:RETI
30 2F2B            ORG 2F2BH
31 2F2B   022080   JMP 2080H ;Jump to
                   TIM0ISR subroutine.
32                 END
```

Note that in the TIM0ISR subroutine, R6 register is used for accumulating time counts of 65 ms. A full count starting with 0000 to FFFF in timer registers TH0 and TL0 causes a 65,536μs time count with the standard 12MHz crystal (and 65,536x12/11.09=70,910 μs with 11.09MHz crystal). Sixteen times the former value (or fifteen times the latter value) gives 1 second. Thereupon, we divide 1 second further by 120 to get 2 minutes and by 180 to get 3 minutes.

Example 2: Real-time clock program on LCD. The following real-time clock program is self-explanatory:

```
INPUT FILENAME : REALCLK.ASM
1                  $MOD52
2   2000           ORG 2000H
3   2000   758961  MOV TMOD, #01100001B
                   ;Set Timer-1for counter &
                   Timer-0 for timer operation
4   2003   758CF0  BEG:MOV TH0,#0F0H
                   ;Set Timer Reg.0 to F000H,
5   2006   758A00  MOV TL0,#0H ;(gives 4ms
                   with 12MHZ crestal)
6   2009   7EFF    MOV R6,#0FFH ;Interrupt
                   count location
7   200B   D28C    SETB TR0 ;Start Timer-0.
8   200D   D2A9    SETB ET0 ;Enable Timer-
                   0 Interrupt
9   200F   D2AF    SETB EA ;Enable Interrupt
                   Global.
10  2011   7412    MOV A,#12H ;Initial
                   Hour(Set)display
11  2013   F541    MOV 41H,A ;"0"To Start
                   with secs ctr
12  2015   7400    MOV A,#0 ;Iniial
                   Minute(Set) display
13  2017   F540    MOV 40H,A
14  2019   7A00    MOV R2,#0
15  201B   C28D    CLR TF0
16  201D   80FE    SJMP $
17
18  2080           ORG 2080H
19  2080   758CF2  TIM0ISR:MOV TH0,#0F2H
20  2083   DE2E    DJNZ R6,K1A ;256 Times
                   4ms gives 1 sec
21  2085   EA      MOV A,R2
22  2086   2401    ADD A,#1
```

TABLE III
Interrupt Priority Register

IP — ADDRESS = 0B8H, BIT ADDRESSABLE — RESET VALUE = XX00O000B

7	6	5	4	3	2	1	0
—	—	PT2	PS	PT1	PX1	PT0	PX0

PRIORITY BIT = 1 ASSIGNS HIGHER PRIORITY
PRIORITY BIT = 0 ASSIGNS LOWER PRIORITY

BIT	SYMBOL	FUNCTION
IP.7	—	NOT IMPLEMENTED, RESERVED FOR FUTURE USE.
IP.6	—	NOT IMPLEMENTED, RESERVED FOR FUTURE USE.
IP.5	PT2	TIMER 2 INTERRUPT PRIORITY BIT.
IP.4	PS	SERIAL PORT INTERRUPT PRIORITY BIT.
IP.3	PT1	TIMER 1 INTERRUPT PRIORITY BIT.
IP.2	PT0	TIMER 0 INTERRUPT PRIORITY BIT.
IP.0	PX0	EXTERNAL INTERRUPT 0 PRIORITY BIT.

```
23 2088   D4       DA A
24 2089   FA       MOV R2,A
25 208A   BA601D   CJNE R2,#60H,K1;If < 1
                   minute
26 208D   7A00     MOV R2,#0 ;Start with 0
                   after 60 sec
27 208F   E540     MOV A,40H
28 2091   2401     ADD A,#1
29 2093   D4       DA A
30 2094   F540     MOV 40H,A
31 2096   B46011   CJNE A,#60H,K1
32 2099   754000   HR:MOV 40H,#0
33 209C   E541     MOV A,41H
34 209E   2401     ADD A,#1
35 20A0   D4       DA A
36 20A1   F541     MOV 41H,A
37 20A3   B42404   CJNE A,#24H,K1 ;If < 24 Hour
38 20A6   7400     MOV A,#0 ;Start with 0
                   after 24 hour
39 20A8   F541     MOV 41H,A
40 20AA   854082   K1:MOV DPL,40H
41 20AD   854183   MOV DPH,41H
42 20B0   120130   CALL 0130H
43 20B3   32       K1A:RETI
44 2F2B            ORG 2F2BH
45 2F2B   022080   JMP 2080H ;Jump to
                   TIM0ISR subroutine
46                 END
```

The timings of the hour and minute need to be entered in the program itself at locations 2012 and 2016.

Use of external interrupts

The external interrupts available are very useful for certain applications when interfacing to unexpected events/inputs and actions to be taken when the external interrupt occurs. Pins 12 and 13 of 89C51 are used for external interrupt as follows:

```
Pin 12   External interrupt-0
Pin 13   External interrupt-1
```

These are pulled up high, and one has to bring the pin to low logic level to cause an interrupt. However, the interrupt will take place only if EA and EX0 or EX1 bits in the IE register (shown above) are set.

One can do polling with external interrupt by using the following instruction:

```
JNB IE0
```

where IE0 is a bit address in another TCON register. This is not to be confused with the interrupt-enable register IE1. IE0 is having the bit address as 89H and IE1 as 8BH.

In the TCON register given in the beginning, note that it contains all the flags for each of the interrupts that may be used in 89C51. For example, IE0 means that if pin 12 is brought low externally causing an external interrupt, the corresponding flag will get set, and, in addition to this, the ISR at address 0003 will also be vectored into, provided the interrupt for external interrupt-0 is enabled by an instruction using interrupt-enable IE register A8H using the instruction:

```
ORL IE,81H ;external Interrupt(EX0) is Set.
```

Thus, either polling the IE0 bit or using ISR with external interrupt method can be employed.

Interrupt-driven keyboard

The interrupt-driven keyboard unit makes use of the PCAT keyboard with CLK (pin 1) and DATA (pin 2) of the keyboard connected to pins 12 and 14 of the IC 89C51 as shown in Fig. 3 of Part I. The program for keyboard, which we have dealt with earlier, was not using the interrupt and was purely a continuously scanned routine. So, when we do a CALL KEYBOARD instruction, the program goes into that routine and comes out only after a key is pressed on the keyboard.

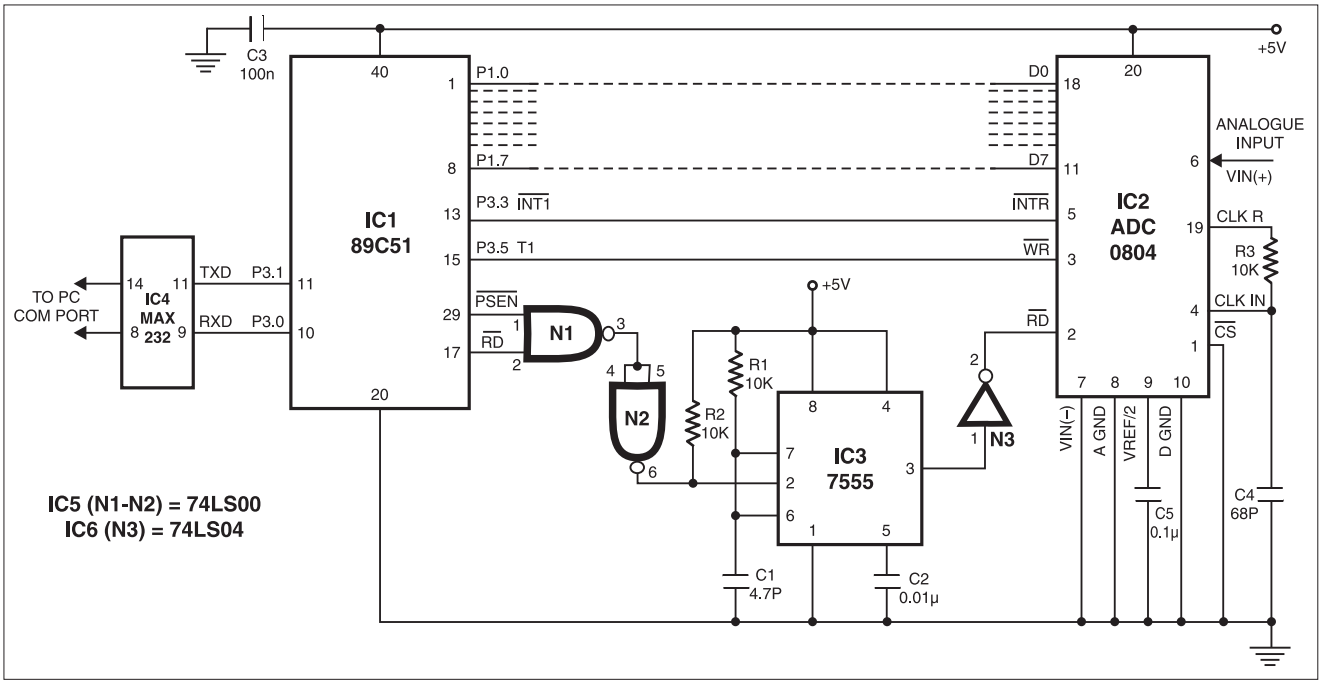


Fig. 12: Interface to the 89C51 for data transfer to computer via RS232 port

This is appropriate for an application involving data entry. But, if we had to do some other program in the meanwhile, it would be waste of time to keep on remaining inside the KEYBOARD-routine until a key gets pressed. For example, we may like to show a real-time clock on the LCD until the user presses the key. That would not be possible with the polled program for the keyboard.

Also, if we had used an 8-digit, 7-segment display on the unit for showing address and data, we should multiplex the display and refresh the writing into them continuously. That would be by a program to refresh the display and in that case, the keyboard cannot be a polled routine.

Pin 12 of 89C51 for clock input from the PC's keyboard connector is just the external interrupt INT0 pin. So, when a user presses a key, the first clock that comes can be made to cause an interrupt and thereby, if the interrupt is enabled, the program can be made to take care of the collection of the 8-bit scan code and then return from the ISR with this code. So, the time taken for the keyboard routine is just the fraction of a millisecond. Use of interrupt-driven keyboard routine is given below:

```
ORG 0003   ;Interrupt.Vector Location
JMP ISR
  ISR:  PUSH A
        PUSH PSW
  B1:   CALL KBD
        CJNE R7,#F0H,A1 ;check for key
        release scan code F0H
        CALL KBD    ;collect next scan code
 RT1:   POP PSW
        POP A
        SETB 60H
        CLR 89H  ;Tcon.1 cleared to
        re_enable interrupt
        RETI
 A1:    SJMP B1
 KBD:   MOV R3,#8 ;to collect 8 bits, a counter
                  ;is used
        MOV R7,#0 ; this collects the data bits
 KP1:   MOV C,P3.2 ;clock
        JC KP1
 K41:   MOV C,P3.2
        JNC K41
 K51:   MOV C,P3.2
        JC K51
        MOV C,P3.4 ;Data bit( At pin 14)
        MOV A,R7
        RRC A
        MOV R7,A
 K61:   MOV C,P3.2
        JNC K61
        DJNZ R3,K51
        ACALL DELAY
        RET
```

There is no change in the KBD routine that collects 8 data bits (one bit after each clock). The ISR is the vectored address to which the jump at the 0003 takes place. There, the program checks for key-release code F0 H and then acquires the subsequently coming scan code and returns.

Since our monitor vectors to 2F03H, the ISR address can be this itself or a further jump at RAM location using a jump instruction at 2F03H as under:

```
2F03     02 20 C0  JMP 20C0H
                   ; FOR JUMP TO 20C0H.
```

Keyboard time setting for real-time LCD clock

Now that the program for handling external interrupt-based keyboard has been introduced, we are free to include this into the real-time clock program for enabling key-entry based time setting. So, now we use two interrupts:

1. Timer-0 driven interrupt
2. Keyboard-driven interrupt

We can give same priority to both. In fact, when keyboard is pressed, we can give a higher priority to the keyboard-driven interrupt. When one wants to change the time, there is no need to keep the earlier timing going on. So, we load the Interrupt priority register with a higher priority for external interrupt EX-0 .One can set or clear five bits of the IP register (B8H) to make priority high or low, respectively. Upon reset, all interrupts are cleared and are at low priority. The interrupt priority register details are given in Table III.

These are bit addressable from B8H to BCH.

To let PX0 to have higher priority,

make IP register as 01 H.

Now the program that works as real-time clock with key entry for hours and minutes is given. When M and H keys are pressed, it increments minutes and hours respectively.

The program is as follows:

```
 INPUT FILENAME : REALCLK.ASM (With keyboard time set)
1            $MOD52
2   2000     ORG 2000H
3   2000     758961   MOV TMOD,#01100001B
                      ;Set Timer-1for counter
                      &Timer-0 for timer operation
4   2003     758CF0   BEG:MOV TH0,#0F0H ;Set
                      Timer Reg.0 to F000H,
5   2006     758A00   MOV TL0,#0H ;(gives 4ms
                      with 12MHZ crestal)
6   2009     7EFF     MOV R6,#0FFH ;Interrupt
                      count location
7   200B     D28C     SETB TR0 ;Start Timer-0.
8   200D     D2A9     SETB ET0 ;Enable Timer-
                      0 Interrupt
9   200F     D2A8     SETB EX0 ;Enable
                      keyboard Interrupt
10  2011     D2AF     SETB EA ;Enable Interrupt
                      Global.
11  2013     75B801   MOV IP,#01H
12  2016     7412     MOV A,#12H ;Initial
                      Hour(Set)display
13  2018     F541     MOV 41H,A ;"0"To Start
                      with secs ctr
14  201A     7400     MOV A,#0 ;Iniial
                      Minute(Set) display
15  201C     F540     MOV 40H,A
16  201E     7A00     MOV R2,#0
17  2020     C28D     CLR TF0
18  2022     80FE     SJMP $
19  2024     UPDATE:
20  2024     B43A10   K$:CJNE A,#3AH,H
                      ;Minute Set by M key
21  2027     E540     MINSET:MOV A,40H
22  2029     2401     ADD A,#1
23  202B     D4       DA A
24  202C     F540     MOV 40H,A
25  202E     B46022   CJNE A,#60H,DISP1
26  2031     754000   MOV 40H,#0
27  2034     02203A   JMP SK
28  2037     B43318   H:CJNE A,#33H,RT1 ;Hour
                      Set by H key
29  203A     E541     SK:MOV A,41H
30  203C     2401     ADD A,#1
31  203E     D4       DA A
32  203F     F541     MOV 41H,A
33  2041     122053   CALL DISP1
34  2044     B4240B   CJNE A,#24H,RT1 ;If
                      < 24 Hour
35  2047     7400     MOV A,#0H ;Start with 0
                      after 24 hour
36  2049     F541     MOV 41H,A
37  204B     7400     ZM:MOV A,#0
38  204D     F540     MOV 40H,A
39  204F     122053   CALL DISP1
40  2052     22       RT1:RET
41
42  2053     C0E0     DISP1:PUSH ACC
43  2055     854082   MOV DPL,40H
44  2058     854183   MOV DPH,41H
45  205B     120130   CALL 0130H  ;Display
46  205E     D0E0     POP ACC
47  2060     22       RET
48  2080              ORG 2080H
49  2080     758CF0   TIM0ISR:MOV TH0,#0F0H
50  2083     DE30     DJNZ R6,K1A ;256 Times
                      4ms gives 1 sec
51  2085     7EE1     MOV R6,#225
52  2087     EA       MOV A,R2
53  2088     2401     ADD A,#1
54  208A     D4       DA A
55  208B     FA       MOV R2,A
56  208C     BA601D   CJNE R2,#60H,K1 ;If < 1
                      minute
57  208F     7A00     MOV R2,#0 ;Start with 0
                      after 60 sec
58  2091     E540     MOV A,40H
59  2093     2401     ADD A,#1
60  2095     D4       DA A
61  2096     F540     MOV 40H,A
62  2098     B46011   CJNE A,#60H,K1
63  209B     754000   HR:MOV 40H,#0 ;60 sec
                      made '0'
64  209E     E541     MOV A,41H  ;increment
                      Hour
65  20A0     2401     ADD A,#1
66  20A2     D4       DA A
67  20A3     F541     MOV 41H,A
68  20A5     B42404   CJNE A,#24H,K1
69  20A8     7400     MOV A,#0H  ;greater than
                      24,made '0'
70  20AA     F541     MOV 41H,A
71  20AC     854082   K1:MOV DPL,40H
72  20AF     854183   MOV DPH,41H
73  20B2     120130   CALL 0130H ;Display
74  20B5     32       K1A:RETI
75  20C0              ORG 20C0H
76  20C0     C0E0     KBDISR:PUSH ACC
77  20C2     C0D0     PUSH PSW
78  20C4     1220DB   B1:CALL KBD
79  20C7     BFF00F   CJNE R7,#0F0H,A1
80  20CA     1220DB   CALL KBD
81  20CD     D0D0     RR1:POP PSW
82  20CF     D0E0     POP ACC
83  20D1     D260     SETB 60H
84  20D3     C289     CLR 89H
85  20D5     EF       MOV A,R7
86  20D6     1124     CALL UPDATE
87  20D8     32       RETI
88  20D9     80E9     A1:SJMP B1
89  20DB     7B08     KBD:MOV R3,#8
90  20DD     7F00     MOV R7,#0
91  20DF     A2B2     KP1:MOV C,P3.2
92  20E1     40FC     JC KP1
93  20E3     A2B2     K41:MOV C,P3.2
94  20E5     50FC     JNC K41
95  20E7     A2B2     K51:MOV C,P3.2
96  20E9     40FC     JC K51
97  20EB     A2B4     MOV C,P3.4
98  20ED     EF       MOV A,R7
99  20EE     13       RRC A
100         20EF     FF       MOV R7,A
101         20F0     A2B2     K61:MOV C,P3.2
102         20F2     50FC     JNC K61
103         20F4     DBF1     DJNZ R3,K51
104         20F6     11F9     ACALL DELAY
105         20F8     22       RET
106         20F9     7D80     DELAY:MOV R5,#80H
107         20FB     DDFE     DJNZ R5,$
108         20FD     22       RET
109         2F03     ORG      2F03H
110         2F03     0220C0   EX0INTV:JMP 20C0H
                     ;Jump KBDISR subroutine
111         2F2B     ORG      2F2BH
112         2F2B     022080   TIM0INTV:JMP
                     2080H ;Jump TIM0ISR subroutine
113                  END
```

The ISR for timer vectors to 2F2BH from the internal ROM at 000BH. Here, a further jump is made to 2080H, where timer 0 comes into action by counting the interrupt (which occurs about once every 4 ms) and updates the time-register seconds (R2) and locations 40H and 41H into which the accumulated minutes and hours are kept. The keyboard-interrupt routine, which vectors from the ROM-jump at 2F03H to 20C0H, reads the keyboard after the key-release scan code of F0Hex. This code is compared for M key scan code (3A H) for minutes setting.

For hours setting, the H key is used and hence its scan code of 33H is compared.

Upon keying M, the minutes location 40H is incremented by 1. Also, check is made to ensure that minutes is less than 60, else, it is set to zero and hours is incremented by one. Upon keying H, the hours location 41H is accordingly incremented by 1, checked for higher than 24, and reset if so. The keyboard control works only when a key is pressed and the clock continues to work until the setting is changed.

Serial interrupt-based programs. In Part II, we had seen programs for serial transmission and reception. These programs used only polling of the transmit and receive flags RI and TI in order to check for data received or transmitter buffer empty signals. However, in some

cases, it may be required to do the reception of data in the background without disturbance to the main program.

In such cases, the interrupt-driven reception is helpful. Upon reception of a data byte, there will be an interrupt to the serial interrupt vector at 0023H, which is an internal code memory area inside the 89C51. Here, a secondary jump address is stored to location 2E00H in RAM area.

So, one writes ISR from this 2E00 address, either with a jump elsewhere or exactly from 2E00H onwards. The program given here uses such a received data interrupt in the background. Any data from the PC through serial port gets stored in RAM from 2100 H onwards.

In this receive interrupt subroutine (SER_int_ISR) at 2080H, first of all, a check is made to find whether the interrupt happened due to received byte or transmit byte. Only if there is valid RI flag setting due to a reception, the data from SBUF is moved to RAM address using MOVX @DPTR,A.

If the interrupt service was not due to RI flag, a simple return takes place.

In the program, a fixed byte '3A' is continuously transmitted in the main program. This is done by polling the TI bit. The program is:

```
INPUT FILENAME : SERINTP.ASM
1  2000          ORG 2000H
2  2000  90 21 00 MOV DPTR,#2100H
3  2003  75 98 52 MOV SCON,#52H
                  ;BAUDRATE
4  2006  75 89 20 MOV TMOD,#20H ;T1 8
                  BIT AUTORELOAD
5  2009  75 8D FD  MOV TH1,#FDH ;9600
                  Buad Rate
6  200C  D2 8E   SETB TR1 ;START TIMER-1
7  200E  43 A8 90G:ORL IE,#90H ;ENABLE
                  SERIAL INT.
8  2011  74 3A    MOV A,#3AH ;JUST ANY
                  SYMBOL.
9  2013  11 19    ACALL TOUT
10 2015  01 0E    AJMP AG
11 2017  80 FE    SJMP $ ;OR ANY main
                  PROGRAM HERE
13 2019  30 99    FD TOUT:JNB TI,$ ;wait
                  for Transmit flag ready
14 201C  C2 99    CLR TI  ;clear it for next
                  transmit
15 201E  F5 99    MOV SBUF,A ;move to
                  transmit buffer (one tran-
                  smit buffer another is re-
                  ceive buffer)
16 2020  22       RET
17 2080           ORG 2080H
18 2080           SER_int_ISR:
19 2080  30 98 06 JNB RI,NO_DATA ;if not
                  received data interrupt
                  ;return with no action
20 2083  E5 99    MOV A,SBUF ;SERIAL
                  DATA INTO A
21 2085  C2 98    CLR 98H ;clear receive
                  int.flag
22 2087  F0       MOVX @DPTR,A ;SAVE
                  IN RAM & INCREMENT
                  ADDRESS
23 2088  A3       INC DPTR
24 2089  32       NO_DATA:RETI
25 2E00           ORG 2E00H
26 2E00  02 20 80 JMP 2080H ;subroutine
                  SER_inrISR.
27 2E03           END
```

On running the serial communication program XTALK.EXE on the PC and setting the baud rate (9600) and COM port (say, 1) same as in Part II, type GO LO and press Enter key. When we execute the program, local screen shows continuous line. Any key pressed on the PC keyboard is seen on the screen and stored at RAM location 2100H into ASCII converted code (only lower-case character).

Interfacing an ADC to the system

In order to interface the ADC chip, such as the ADC0804, there are two choices. We can configure the chip as an external RAM address by connecting the chip-select pin of the ADC to a normal address-select group from this circuit's 74138 decoder. Then, by connecting the Read and Write signals, one can read the ADC value, which gives the digital value of the analogue signal input. The analogue signal is in the 0-5V range.

Pin 20 is Vcc (5V) and pin 10 is ground in the ADC 0804. Pin 8 is analogue ground, which is, of course, the same as the digital ground. VI(+) and VI(-) are the pins for input signal. If VI(-) pin 7 is connected to pin 8, then, we can digitise only positive-going signals above 0 volt. Pin 6 is the input signal pin. The internal clock can be generated by connecting a resistor of 10k between pins 19 and 4, while connecting pin 4 through a capacitor of 68 pF to ground. This will generate a 100kHz internal clock, which is used for the analogue-to-digital conversion.

Pin 1 (chip-select pin), pin 2 (read active-low input), and pin 3 (write active-low input) of ADC 0804 are the interface pins to the microcontroller. When the digital conversion is over, it is signified on BUSY (pin 5). If this pin is low, it indicates conversion is completed. Pins 18 to 11 are the data pins D0 to D7 of the 8-bit conversion. These are to be read by the microcontroller.

Method 1. Interface as external memory. In this method, the chip-select pin is connected to a group-select pin from the address decoder (74LS138), say, A000H from pin 10. Read (RD) and Write (WR) signals from pins 17 and 16 are to be connected to pins 2 and 3 of the ADC chip. The DATA bus is connected to pins 18 through 11 for reading the converted data. The Busy pin (pin 5) can be connected to another pin 13 (INT1) of the 89C51. The program can first start the conversion, check the busy status at INT1 (pin 13), and then read the data when it is not busy. To start the conversion, a write instruction to that memory location is to be performed.

The program is as follows:

```
MOV DPTR,#A000H
MOVX @DPTR,A ;Any data can be written, so
A is not  specified. ;now conversion would have
started in the ADC chip. JB EX1,$ ;Wait till
the bit of pin 13 has not become low
 ; now bit is low, so conversion is over.
  MOV A,@DPTR ;READ the ADC converted
value RET      ;return from sub program.
```

Method 2. Interface through port of the microcontroller (Fig. 12). In this method, port 1 (pin 1 through pin 8) can be connected to the ADC Data (pin 18 through pin 11). Then, the chip-select pin can be grounded for the ADC. The Write signal of ADC (pin 3) can be given to the T1 (pin 15). Read signal of ADC (pin 2) is connected to the ANDed output of RD signal and PSEN signal (same as in Fig. 3 after stretching the read pulse. In this case, the initiation of the conversion is done by making the WR (pin 3) of the ADC low and then back to high. Pin port 3.5 (pin T1) has to pulse it. This causes a low-going pulse to the Write signal input of the ADC. Then, we read busy pin 5 of the ADC, which is wired to pin 13 of the 89C51. This pin is the external interrupt-1. This can be tested by a bit-test instruction.

```
SETB P3.5
NOP
CLR P3.5 ; A 2 MICROSECOND PULSE HAS
BEEN GIVEN JB EX1,$ ; If the pin 13 has
not gone low, wait here
MOV A,P1 ;READ the port-1 data connected to
```

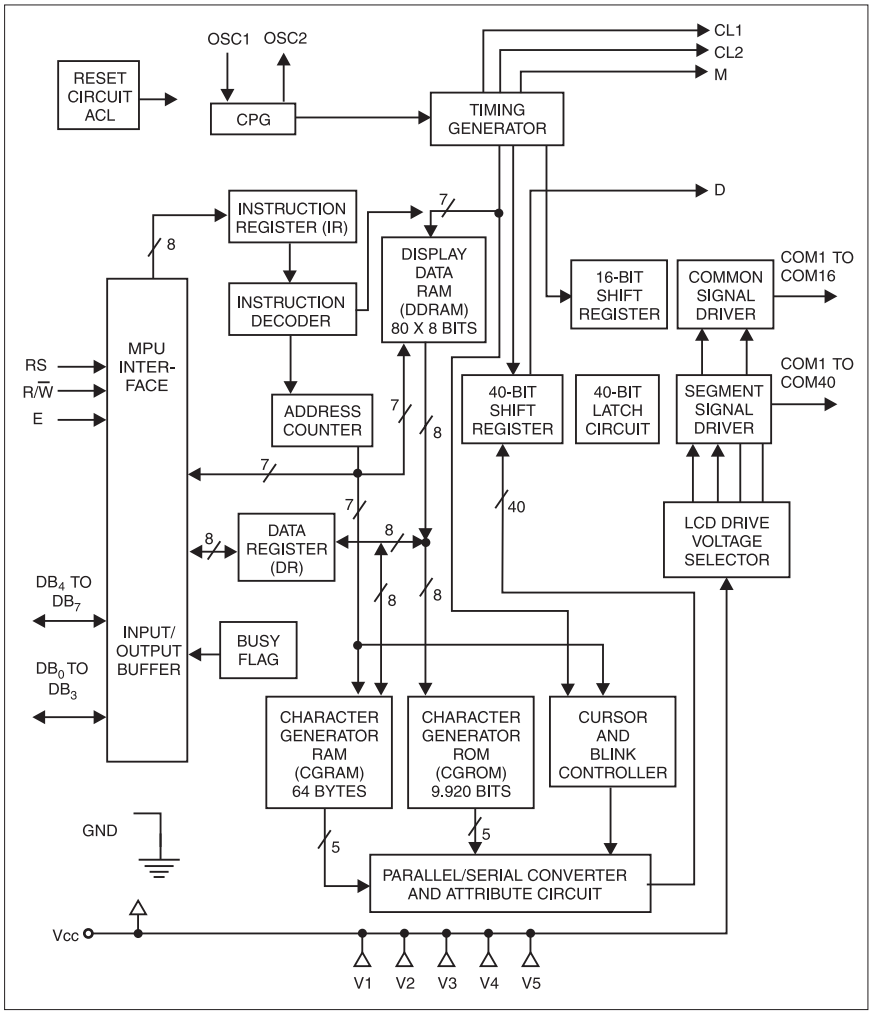


Fig. 13: Block diagram of HD44780U

ADC data pins.
This method is useful for a dedicated use of the 89C51 with no external memory as a single-chip component.

Program to read ADC and send the data to the serial port. When samples of data are to be taken periodically with a definite time interval between successive samples, it is convenient to use the 89C51's internal timers. Rather than reading the data as and when the conversion is over, it is advisable to read the data at fixed regular intervals of time to ensure that the samples are equidistant in time.

We can read the data at a slow rate, which may be sufficient for an application, instead of sampling at the fast rate of 50,000 a second, which the ADC_0804 does normally if started again after each conversion.

The program uses timer 0 for this purpose. Its ISR does the ADC reading and sending serially. The ISR of timer 0 reads the ADC data, then once more it starts the ADC by sending a write pulse on port 3.5, by setting this bit, clearing it, and then again setting it. Thus high-to-low and low-to-high transitions give a low-going pulse to the Write pin of the ADC chip. The initialised routine sets both timers as 8-bit auto-reload type. Then, the timing of timer 1 is set as per the baud rate count of 9600, which is FDH. Timer 0 will be operated once every 128 microseconds since the time of conversion of this will be 1000/128=8 kHz, which is usually enough for audio-input signals. In order to enable timer 0 interrupt, the EA and ET0 bits in the interrupt-enable register are to be set. The TI flag is initially kept set, so the transmission can start first. The timers are also started by the set-bit instructions for TR-0 and TR-1.

To send data serially, it is required to change the 8-bit data into two ASCII character codes, because most of the serial-input software on the PC read only 7-bit ASCII data easily. Hence, a small program for conversion of 8 bits into 2-byte ASCII is to be included in the program listing of ADC conversion and serial transmission to PC.

The program is given below:

```
INPUT FILENAME : ADC51.ASM
1  2000                ORG 2000H
                   ;initialise timer 0 and serial port
2  2000   75 89 22  MOV TMOD,#22H
                   ;TIMER-1 8 BIT AUTO
                   RELOAD,ALSO TIMER-0
3  2003   75 98 50  MOV SCON,#50H ;SET
                   SERIAL PORT TO MODE 1
4  2006   75 8D FD MOV TH1,#FDH ;SET
                   BAUD RATE TO 9600
5  2009   75 8C 80  MOV TH0,#80H ;once 128
                   microsecond,timer-0 interrupts.
6  200C   D2 AF      SETB EA ; GLOBAL
                   interrupt enable
7  200E   D2 A9      SETB ET0 ;TIMER 0
                   INTERRUPT
                   ;SERIAL INTERRUPT,IF NEEDED
                   FOR RECEPTION
8  2010   D2 8C     SETB TR0
9  2012   D2 8E    SETB TR1 ;START TIMERS
10 2014   D2 99      SETB TI
11 2016   80 FE      SJMP $ ;KEEP HERE
12 2018   E5 40      SEND_DATA:MOV A,40H
13 201A   30 99      FD JNB TI,$
14 201D   C2 99      CLR TI
15 201F   12 20 31   CALL HEXASCH
16 2022   F5 99      MOV SBUF,A
17 2024   E5 40      MOV A,40H
18 2026   30 99      FD JNB TI,$
19 2029   C2 99      CLR TI
20 202B   12 20 34   CALL HEXASCL
21 202E   F5 99      MOV SBUF,A
22 2030   22         RET
23                   HEXASCH:
                   ;Converts a nibble at high byte
                   position to ASCII code
24 2031   54 F0      ANL A,#0F0H ;PICK THE
                   HIGH NIBBLE PART
                   BY ANDING WITH FOH
25 2033   C4        SWAP A  ; BRING TO
                   LOW NIBBLE SIDE
26                   HEXASCL: ;Converts a
                   nibble at low byte half
                   into an aScii code
27 2034   54 0F      ANL A,#0FH
28 2036   C3         CLR C
29 2037   94 0A      SUBB A,#0AH
30 2039   40 02      JC ZTO9
31 203B   24 07      ADD A,#7
32 203D   24 3A      ZTO9:ADD A,#3AH
33 203F   22         RET
34 2080              ORG 2080H
35                   TIM0ISR:
36 2080   20 AA 0E JB EX1,RETPT ;Return if
                   adc is busy
37 2083   E5 90      MOV A,P1  ;Read adc
                   value into accumulator
```

```
38 2085   F5 40      MOV 40H,A ;SAVE IT IN
                     40H ;Start next conversion
39 2087   D2 B5      SETB P3.5 ; High to
                     write pin
40 2089   00         NOP
41 208A   C2 B5      CLR P3.5 ; LOW TO IT
42 208C   00         NOP
43 208D   D2 B5      SETB P3.5 ;HIGH AGAIN
44 208F   11 18      CALL SEND_DATA
45 2091   32         RETPT:RETI
                     ORG 2F2BH
46 2F2B   02 20 80   TIM0VECT:JMP 2080H
                     ;subroutine TIM0ISR
47 2F2C              END
```

Suggestion. For this project, use only Hitachi's HD44780 controller based on 16-character x 1-line LCD module. An internal functional block diagram of the same is included here in Fig. 13. The complete datasheet of HD44780U is in CD. ❑

Readers' comments:

Q1. My LCD shows only eight black blocks and not numbers as you have claimed. I checked continuity between all pins and found it to be fine. The tips for troubleshooting at the end of the article are of no help. I even tried to move data directly into the accumulator and then display it on the LCD but did not succeed. However, the LCD shows data if its pins are connected directly to the microcontroller and that too for a different program.

Saurabh
Through e-mail

Q2. As no number or make is suggested for the use of 16x1-line type LCD, I used the Lampex make LM16200SYBCLY. Kindly tell me whether this LCD is applicable or not. If not, suggest me an LCD which is easily available in the market.

Q3. Where can I find compilers and linkers for the kit?

Q4. Can I write the hex-dump of 89C51 development kit's monitor program directly into the EPROM as given in the program?

Sumit Kumar
Through e-mail

Q5. I have assembled the 89C51 development kit on a plate-through hole PCB procured from an agent of Kit'n'Spares. It is working in all respects (starting at address 2000 when reset, data increment, data decrement and register increment). When I press any key from 1 to F on the keyboard, the respective data appears in the data field but within a moment FF is displayed in the data field as follows:

```
2000 FF  ; on reset, FF is displayed at address 2000.
2000 F1  ;on pressing key 1 from key board
2000 FF  ;on relies key 1 from key board
2000 11    ;on pressing key 1 from key board continuously
2000 11    ;on pressing key 1 from key board continuously & enter key
2001 FF  ; address incremented to 2001 and FF is displayed in data field
2001 22  ;on pressing key 2 from key board continuously & enter key
2001 22  ;on pressing key 2 from key board  continuously & enter key
2003 FF ; address incremented to 2003
```

Now, if I change the data at various addresses in the monitor program as follows:

```
Original          Changed
02DB 74 4C        02DB 74  4C
02DD 22           02DD 22
02DE 74 FF        02DE 01
30 jmp sc2
02E0 22           02E0  00
nop
```

and run the program, the result is the same as above with a few changes as given below:

```
2000 xx    ; on reset
2000 x1    ;on pressing key 1 on key board
2000 xx    ;on relies key 1 on key board
 2000 11      ;on pressing key 1 on key board continuously
 2000 11      ; on pressing key 1 on key board continuously & enter key
2001 xx   ; addres incremented to 2001
 2001 22      ;on pressing key 2 on key board continuously & enter key
 2001 22      ;on pressing key 2 on key board continuously & enter key
2003 xx
```

where x may be any hex number from 1 through F.

Another problem is that the kit resets automatically during address increment, decrement and while entering number in my modified monitor. Kindly tell me where the fault lies?

Bipin J. Patel
Surat, Gujarat

The author K. Padmanabhana replies:

A1. You have checked only the connections to the LCD. Check the gating signal to pin 6 of the LCD. You should use only a Hitachi-make LCD.

A2. For this kit, you need to use Hitachi make HD44780U LCD. Its details are included in CD. The Lampex make LCD doesn't work on the kit.

A3. The cross-assembler X8051 and linker link151 are included in CD. The assembler ASM51 is also included in CD.

A4. Yes, the hex dump of the listing given can be programmed into the 89C51 IC for the unit to work.

A5. The keyboard problem seems to be due to some mistake in the monitor program programmed by you. Try changing the address 01 1E 01 31 to 01 32 jmp sc1.

The key FF hex, which is the code obtained for any non-used key or null key, is to be avoided by not taking any action in that case. In your case, FF is also shown on the display and that is why you get that xF after pressing any key and releasing it. This is not changed by your amendment of making FF to 01 in line 02DE. It just makes 1 appear after every key is pressed. Thus, the kit is prevented from working by the null key.

So you have to thoroughly check the codecheck routine to see if there is any mistake during your program burning process. Otherwise, there seems to be nothing wrong in your kit as far as the hardware is concerned. Try changing the keyboard for a chance, since some keyboards may send scan codes so fast that they don't match with our KBD routine.

PC BASED PROGRAMMER FOR THE AT89C51 MICROCONTROLLER

KULAJIT SARMA

The article describes the construction of a low-cost device programmer for the Atmel AT89C51 microcontroller. The programmer is controlled by PC through the parallel port. This programmer is intended for hobbyists as well as the professionals who want to start of with microcontrollers without investing much on universal programmers and development systems, Before starting with the programmer, we should know about the relevant features of the AT89C51 microcontroller itself.

AT89C51 Microcontroller

The 89C51 is the ideal device for learning microcontrollers. A lot of tutorial material and free tools like Assemblers (eg. Metalink's ASM51), source code library, 89Cmanual Developmnt board designs are available on the Internet. The AT89C51 is a CMOS 8-bit mcu with 4KB internal Flash EEPROM program memory with 1000 erase-write endurance. The main features of this device are

- Compatible with Intel MCS-51 (8051 family) products.
- 4KB internal Flash EEPROM program memory.
- Fully static operation from 0 to 24 MHz.
- 128 x 8-bit Internal RAM.
- Two 16-bit Timers/Counters
- 32 programmable I/O lines arranged as 4 Ports.
- Six interrupt sources.
- Programmable serial I/O channel.
- Low power Idle and Power-down modes

The Pin configuration of the 40 pin AT89C51 DIP device is shown in Fig. 1.

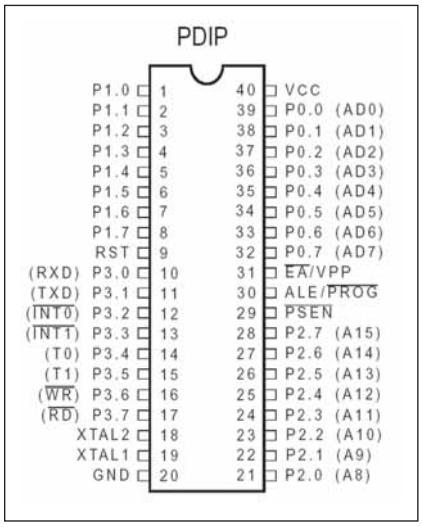


Fig. 1: Pin configuration of the 40-pin AT89C51 DIP device

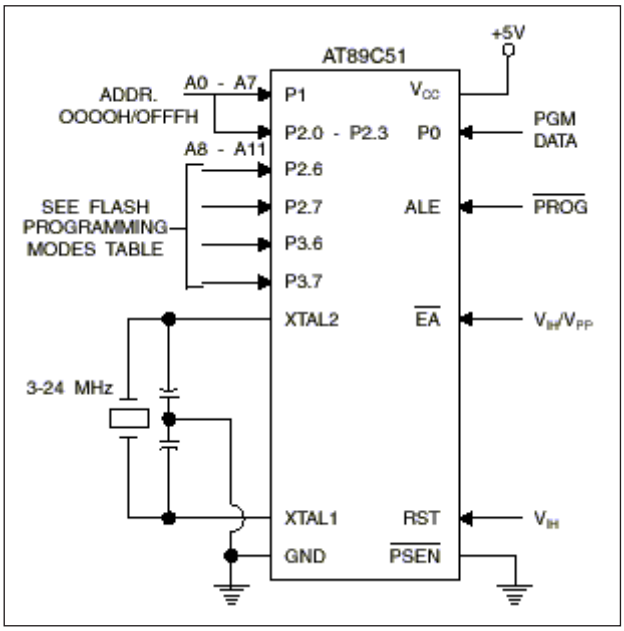


Fig. 2: Programming the Flash

Programming the Flash Memory

The AT89C51 code memory array is programmed byte-by-byte. Initially the Flash memory array comes in the erased state (that is, all bytes = FFH). However to reprogram any non-blank byte in the on-chip Flash memory, the entire memory must first be erased using the chip-erase mode.

Programming Algorithm: Before programming the AT89C51, the address, data and control signals should be set up according to the Flash programming mode shown in Table I, and Figs 2 and 3. To program the AT89C51, proceed as follows:

1. Place the desired memory location on the address lines.
2. Place the appropriate data byte on the data lines.
3. Activate the correct combination of control signals.
4. Raise $\overline{EA}$/Vpp to 12V to enable programming
5. Pulse ALE/$\overline{PROG}$ once to program a byte in the Flash memory. The byte-write cycle is self timed and maximum time required is 2 ms.
6. Repeat steps 1 through 5, changing the address and data for the entire array or until the end of the program file is reached

Ready/Busy Checking: The progress of byte programming can be monitored by the RDY/$\overline{BSY}$ output signal. During Programming P3.4 is pulled low after ALE goes high to indicate BUSY. When program-ming is done, P3.4 is pulled high again to indicate READY.

Program Verification: If lock bits have not been programmed, the programmed code can be read back for verification via the data lines by setting up the address lines and appropriate combination of control signals.

Erasing the chip: The entire Flash array is erased by using a proper combination of control signals and by holding ALE/$\overline{PROG}$ low for 10 ms. The chip-erase operation must be executed before the code memory can be reprogrammed.

(***Note.*** For more information on programming of AT89C51, go through Atmel's AT89C51 datasheet.)

Programmer Hardware

Fig. 4 shows the programmer circuit for AT89C51 microcontroller that uses low-cost CMOS logic devices.

Two 74HC393 dual 4-bit binary ripple counters (IC2 and IC3) are used to set up address lines for the 89C51. All the four, 4-bit counters are cascaded to form a 16-bit counter. The clock input to the first counter in the chain is used to serially input the address using a single line from the PC.

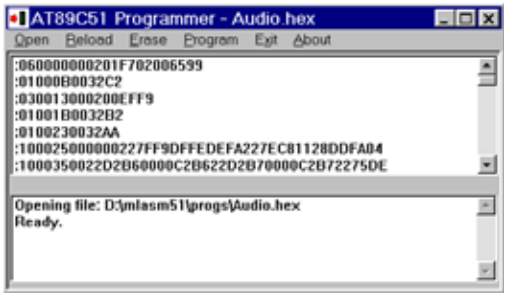
An 8-bit serial-in/ parallel-out shift register IC74HC595 (IC4) with 3-state output latch is used to set up the data lines of the 89C51. For IC4, separate positive edge triggered clocks (SRCK and RCK) are used for serially shifting and parallel latching of the data, respectively.

Quad 2:1 multiple IC 74HC157 (IC5) is used during program code verification to check the data byte through four input

TABLE I
Flash Programming Modes

Mode	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/Vpp	P2.6	P2.7	P3.6	P3.7
Write code data	H	L	⌇	H / 12V	L	H	H	H
Read code data	H	L	H	H / 12V	L	L	H	H
Write lock Bit-1	H	L	⌇	H /12V	H	H	H	H
Write lock Bit-2	H	L	⌇	H / 12V	H	H	L	L
Write lock Bit-3	H	L	⌇	H / 12V	H	L	H	L
Chip erase	H	L	⌇ (1)	H /12V	H	L	L	L
Read signature byte	H	L	H	H	L	L	L	L

Note. 1 Chip erase requires a 10ms PROG pulse.



Screenshot of the programm for Windows version

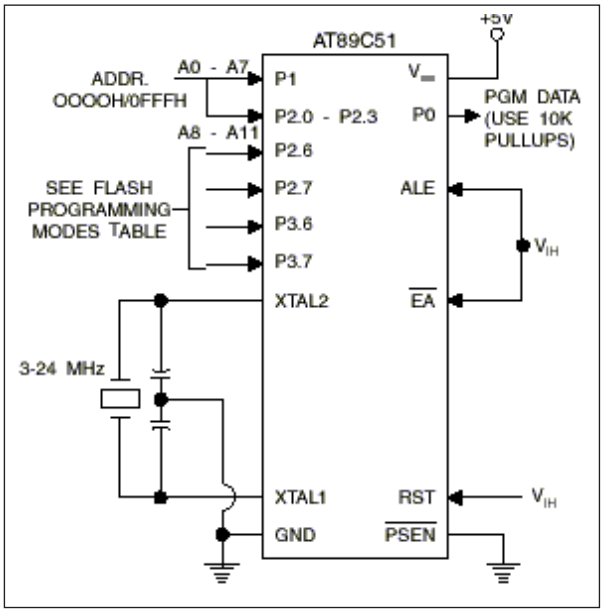


Fig. 3: Verifying the Flash

TABLE II
Details of Connections to the Parallel Port

Parallel Port Pin No.	Port Signal Name	Direction	HW Inverted	Register	Bit No.	Function
1	NStrobe	I/O	Yes	Control	C0	P2.6 Control
2	D0	Out		Data	D0	EA/Vpp Control
3	D1	Out		Data	D1	Address Counter reset (HC393)
4	D2	Out		Data	D2	ALE/PROG
5	D3	Out		Data	D3	Address Counter Serial Clock
6	D4	Out		Data	D4	Not Used
7	D5	Out		Data	D5	Shift Register clock in (HC595)
8	D6	Out		Data	D6	Storage Reg clock-in & $\overline{OE}$ of HC595
9	D7	Out		Data	D7	A/B of HC157 & Serial Data in of HC595
10	NACK	In		Status	S6	3Y (Nibble out of HC157)- D4/D5
11	Busy	In	Yes	Status	S7	2Y (Nibble out of HC157)- D2/D3
12	PaperOut	In		Status	S5	1Y (Nibble out of HC157)- D0/D1
13	Select	In		Status	S4	Ready/$\overline{Busy}$
14	Nauto-Lf	I/O	Yes	Control	C1	P2.7
15	Nerror	In		Status	S3	3Y (Nibble out of HC157)- D6/D7
16	Ninitialize	I/O		Control	C2	P3.6
17	Nselect	I/O	Yes	Control	C3	P3.7
18-25	Ground	Gnd				

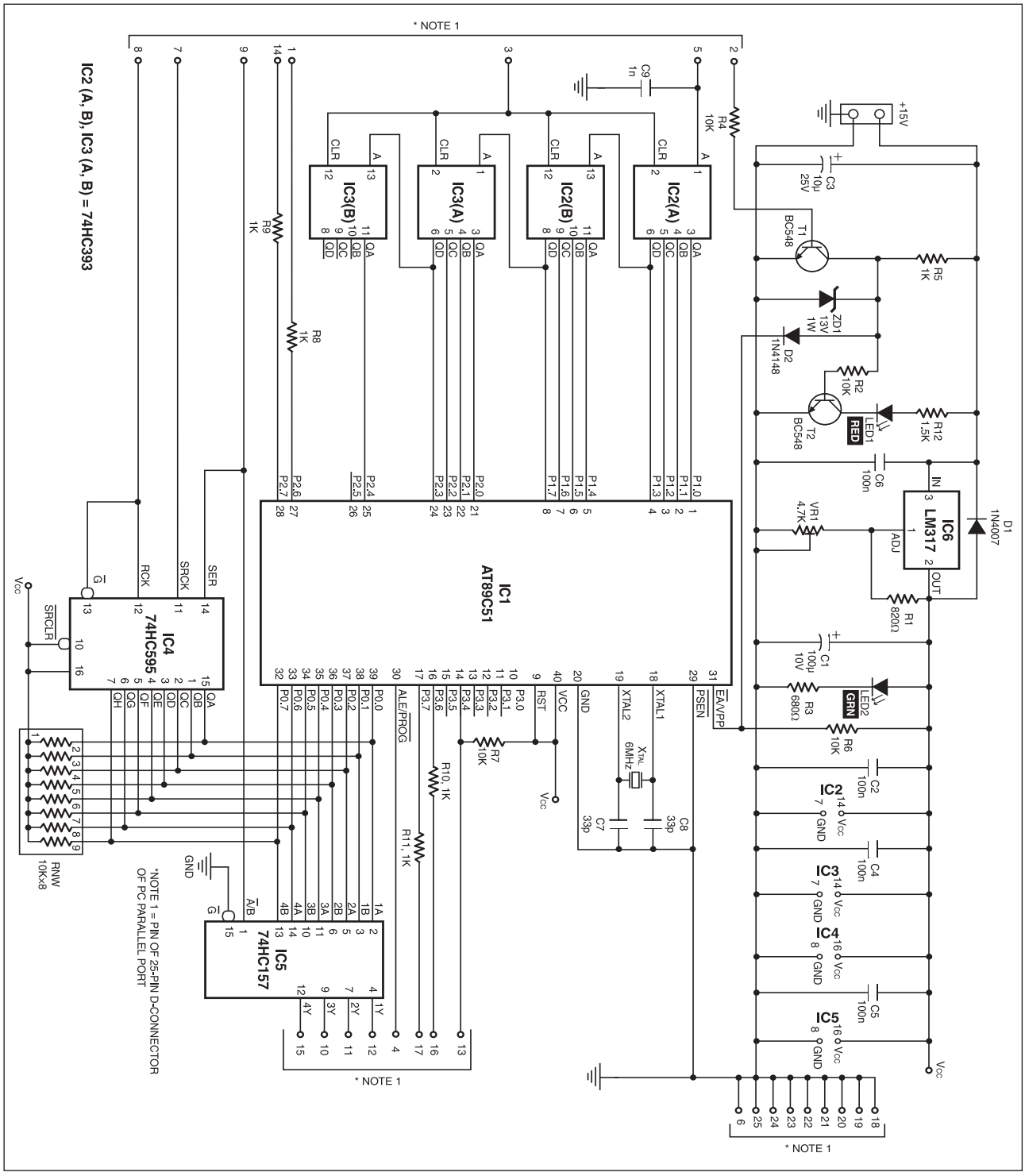


Fig. 4: Schematic diagram of the programmer for AT89C51 microcontroller

lines of the parallel port. The select line (pin 1) of all the four multiplexers inside the IC5 is connected to a single line from the PC. Thus, at any time, only four bits of the data byte can be read through the parallel port.

Transistor T1 is used to turn on/off the programming voltage (Vpp), which is derived from the power supply using a 13V Zener diode ZD1. The +5V supply is generated using a variable regulator LM317L. Adjust the output of the LM317 as close as possible to the high logic level of your PC's parallel port; preferably few millivolt higher than high logic level. This is imperative for proper working of the programmer. (**Caution**. Don't use a fixed voltage regulator like 7805 for this purpose.)

An actual-size, single-side PCB for the programmer is shown in Fig 5 and its component layouts in Fig 6.

The Parallel Port

All the functions of the programmer are controlled by a PC through its parallel port. The details of connections to the parallel port are given in the Table II.

A 40 pin ZIF socket is preferable for

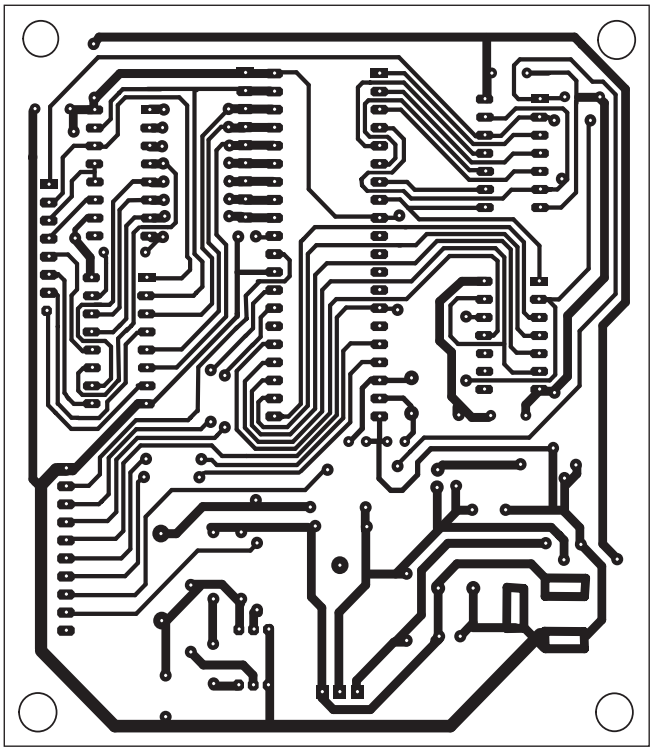


Fig. 5: Actual size, single-side PCB for the programmer

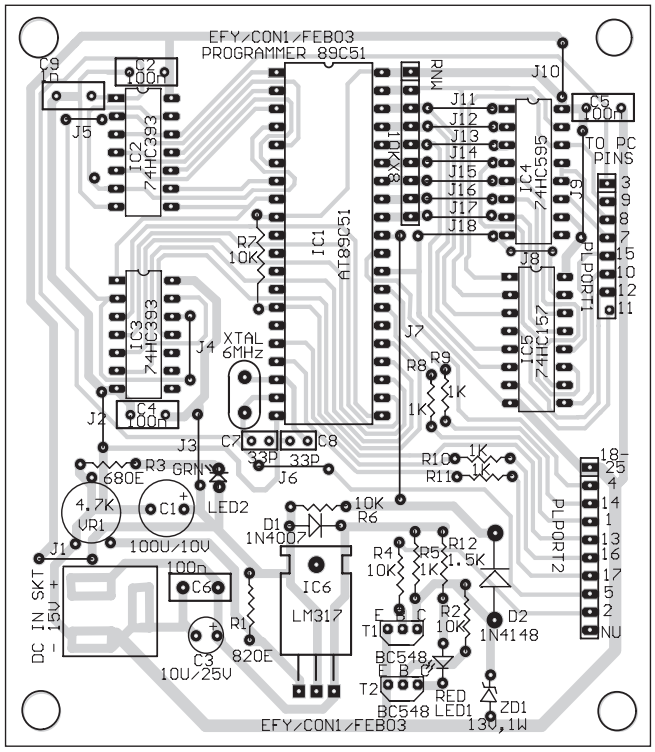


Fig. 6: Component layout for the PCB

the programming socket. The use of a 40-pin DIP IC socket may result in damage to the microcontroller unit while inserting and taking the chip out from the DIP programming socket. (***Caution.*** Don't remove the microcontroller from the socket until the programming is over).

PARTS LIST

Semiconductors:

IC1	-	AT89C51 in ZIF Socket
IC2,IC3	-	74HC393 dual 4-bit binary counter
IC4	-	74HC595 8-bit shift registers with output latches
IC5	-	74HC157 quad 2-line to 1-line data selectors
IC6	-	LM317L adjustable 3-pin regulator
T1,T2	-	BC548B npn transistor
LED1	-	RED LED
LED2	-	GREEN LED
D1	-	1N4007 rectifier diode
D2	-	1N4148 switching diode
ZD1	-	13V, 1W zener diode

Resistors (all ¼-watt ± 5% carbon, unless stated otherwise):

R1	-	820-ohm
R2,R4, R6,R7	-	10-kilo-ohm
R3	-	680-ohm
R5, R8-R11	-	1-kilo-ohm
R12	-	1.5-kilo-ohm
VR1	-	4.7-kilo-ohm preset
RNW	-	10Kilo-ohm X 8, resistor network (SIL9)

Capacitors:

C1	-	100uF,10V electrolytic
C2,C4-C6	-	100nF ceramic disk
C3	-	10uF,25V electrolytic
C7,C8	-	33pF
C9	-	1n F

Miscelaneous:

Xtal	-	6 MHz quartz crystal
		2-pin SIP Connector
		25-pin(F) 'D' connector

The Software

The full programming logic for the programmer is implemented in the software. Two versions of the controlling software are available: one is the DOS based command line version and the other is the Windows version. The DOS version is written in Turbo C++ and is useful for understanding the working of the programmer. The Windows version is written in VC++ 6.0 and is faster with some bugs fixed.

Here, we have given the source code of the DOS version. The source code has been extensively commented for easier understanding. At present, the software can decode only 8-bit Intel Hex files. Though this programmer is designed for the AT89C51, with minor modifications in the software it will be able to program the AT89C52 device also.

The Windows version of the software is more advanced and is easier to use. To access the parallet port, the program uses a dill file named 'inpout32.dll'. Place the dill file in the same directly as the executable file or in the windows\system directory.

EFY note. The Windows executable version, along with other relevant documents/program files, are included in the CD.

Programming the AT89C51

Follow the steps given below to program the AT89C51 microcontroller using the programmer:

- Connect the programmer to the parallel port.
- Swtich on the power to the programmer.
- For DOS version of the program enter the command line as 89C51PRG [path]<intel hex file name>. If everything is right, the programmer prompts you to insert the microcontroller into the socket. (For Windows version simply run the application and load the hex file via 'Open' menu, and then click on 'Program' and follow the on-screen instructions.)
- Turn off the power to the device. Insert the chip into the programming socket and turn the power on .
- Press any key. After the programming is complete, switch off the power to the device and take the chip out.

Caution. Don't insert the microcontroller into the programmer untill you are promoted by the program.

89C51PRG.H—HEADER FILE

```
/************************* --89C51prg.h-- ****************************/
/* Header file for the AT89C51 programmer.
                */
/* Author: Kulajit Sarma
                */
/*********************************************************************************/
# include <iostream.h>  /*for  */
# include <fstream.h>   /*for ifstream */
# include <conio.h>                     /*for inp() & outp() */
# include <dos.h>                       /*for delay() */
# include <string.h>                    /*for strlen */

typedef unsigned char   BYTE;
typedef unsigned int    WORD;

# define TRUE           1
# define FALSE          0
# define OK 1
# define ERROR          0

/* Signal Names */

# define    EA_Vpp                          1
# define    AddrReset               2
# define    ALE_PROG                3
# define    AddrSerialClock         4
# define    DataShiftClock                  5
# define    DataStoreClock                  6
# define    _OE                             7
# define    SerialData              8
# define    _AB                             9
# define    D0_D1                           10
# define    D2_D3                           11
# define    D4_D5                           12
# define    D6_D7                           13
# define    Ready_Busy                      14
# define    P2_6                            15
# define    P2_7                            16
# define    P3_6                            17
# define    P3_7                            18

/* Control signal states        P3.~7    P3.6    P2.~7    P2.~6    */
# define READ 0x07      /*    0        1       1        1        */
# define WRITE0x05      /*    0        1       0        1        */
# define ERASE 0x0A     /*    1        0       1        0        */
# define INVALID 0x03   /*    0        0       1        1        */

/* prints an error message in the format ERROR: <err>\n */
inline void ErrorPrint(char *err1)
{
        cout<<"ERROR: "<<err1<<endl;                                    }

inline void PrintDone(void) { cout<<"Done.\n"<<endl; }

inline void PrintWaitMsg(void)
{
cout<<"Please insert the microcontroller into the programming socket.\n";
cout<<"Press any key to continue...\n";
getch();
cout<<"Programming Flash. Please wait...\n"<<endl;              }
/*********************************************************************//*
Implements the logic for controlling the AT89C51 programmer             */
/* through the parallel port                                            */
/*********************************************************************************/
class ParallelPort
{
private:
        BYTE Data, Status, Control;     /* registers */
        WORD Base;                      /* port base address */
BYTE BitPos[8];                 /* bit positions */
public:
        ParallelPort()
        {       Base=0x378;             /* Default Port Address */
                ResetSignals();
        BitPos[0]=0x01;BitPos[1]=0x02;BitPos[2]=0x04;BitPos[3]=0x08;
        BitPos[4]=0x10;BitPos[5]=0x20;BitPos[6]=0x40;BitPos[7]=0x80;
}
        ~ParallelPort()         { ResetSignals(); }
        /* Set all signals to a value so that the MC could be inserted */
        /*          in the socket safely  */
        void ResetSignals ( void ) ;
        /* Set base address of parallel port */
        void SetPort(WORD Port) { Base=Port; }
        /* Set a signal bit of any register to 1 */
        void SetBit(BYTE Bit);
        /* Clear a signal bit of any register to 0 */
        void ClearBit(BYTE Bit);
        /* Read a bit from status register */
        BYTE ReadBit(BYTE Bit);
        /* Reset Address counters 74HC393 */
        void ResetAddressCounter(void);
        /* Set an address value in the address input lines of 89C51 */
        void SetAddress(WORD Address);
        /* set up control signals for Read, Write or Erase */
        void SetControls(BYTE Mode);
        /* set a data value in the data input lines P0 of 89C51 */
        void SetData(BYTE data);
        /* read a byte of data from 89C51 in read mode for program verification
*/
        BYTE ReadData(void);
};
/*********************************************************************************/
/* Implements the logic for decoding a 8-bit Intel hex file and the Flash */
/* programming algorithm for 89C51 */
/*********************************************************************************/
class Programmer
{
private:
        char *Buffer;           /* temporary buffer to read-in a file into memory */
        char Record[267];       /* temp buffer to hold a iHex file record */
        BYTE Program[4097];     /* stores the binary program decoded from a iHex
file*/
        int BufferPosition;     /* read pointer position n the buffer */
        ParallelPort Port;      /* parallel port object*/

        /* extracts the next record from Buffer into Record*/
        /* each Record is a NULL terminated string. strlen(Record)==0 if error*/
        void GetNextRecord(void);
        /* varifies checksum for a record. Returns 0 if error, 1 if correct*/
        int VarifyChecksum(void);
        /* convert two hex characters to one Byte*/
        BYTE HexToByte(char Hex1,char Hex2);
        /* convert four hex characters to a two-byte Word */
        WORD HexToWord(char Hex1, char Hex2, char Hex3, char Hex4);
        /* converts a hex character into a Decimal integer */
        int HexToDecimal(char Hex);
        /* reads the whole file into the in-memory Buffer. returns 0 on error */
        int ReadFile(char *File);
        /* decodes the iHex file in Buffer into binary format in Program       */
        /* returns 0 on error                                                  */
        int DecodeHexFile(void);
        /* detects presence of programmer by setting up a data value in        */
        /* the 74HC595 and then reading back through HC157. returns 0 on
        error. */
        int DetectHardware(void);
public:
        Programmer();
        /* Programs the iHex `File' into 89C51. Implements the Flash           */
        /* programming algorithm. Returns 0 on error.                          */
        int WriteProgram(char *File);
};
```

89C51PRG.CPP—SOURCE FILE

```
/************************* --89C51prg.h-- ****************************/
/* Source file for the AT89C51 programmer.                          *//* Author:
Kulajit Sarma                                                       */
/*********************************************************************************/

# include "89C51PRG.H"

void ParallelPort::ResetSignals(void)
{
        Data=0x45;              // 01000101
        Control=INVALID;        // 00001001
        outp(Base,Data);
        outp(Base+2,Control);
}

void ParallelPort::SetBit(BYTE Bit)
{
        /* Base register offset */
        WORD Offset=0;

        switch (Bit)
        {
        case EA_Vpp             :
                Data |= BitPos[0]; break;
        case AddrReset:
                Data |= BitPos[1]; break;
```

```
		case ALE_PROG:
			Data |= BitPos[2]; break;
		case AddrSerialClock:
			Data |= BitPos[3];	break;
		case DataShiftClock:
			Data |= BitPos[5]; break;
		case DataStoreClock:
		case _OE:
			Data |= BitPos[6]; break;
		case SerialData:
		case _AB:
			Data |= BitPos[7]; break;
		case P2_6:
			Control |= BitPos[0]; Offset=2; break;
		case P2_7:
			Control |= BitPos[1]; Offset=2; break;
		case P3_6:
			Control |= BitPos[2]; Offset=2; break;
		case P3_7:
			Control |= BitPos[3]; Offset=2;
		}
		if(Offset==0)
			outp(Base, Data);
		else
			outp(Base+Offset, Control);

} /* end SetBit() */


void ParallelPort::ClearBit(BYTE Bit)
{
		/* Base register offset */
		WORD Offset=1;

		switch (Bit)
		{
		case EA_Vpp		:
			Data &= ~BitPos[0]; Offset=0; break;
		case AddrReset:
			Data &= ~BitPos[1]; Offset=0; break;
		case ALE_PROG:
			Data &= ~BitPos[2]; Offset=0; break;
		case AddrSerialClock:
			Data &= ~BitPos[3]; Offset=0; break;
		case DataShiftClock:
			Data &= ~BitPos[5]; Offset=0; break;
		case DataStoreClock:
		case _OE:
			Data &= ~BitPos[6]; Offset=0; break;
		case SerialData:
		case _AB:
			Data &= ~BitPos[7]; Offset=0; break;
		case P2_6:
			Control &= ~BitPos[0]; Offset=2; break;
		case P2_7:
			Control &= ~BitPos[1]; Offset=2; break;
		case P3_6:
			Control &= ~BitPos[2]; Offset=2; break;
		case P3_7:
			Control &= ~BitPos[3]; Offset=2;
		}

		if(Offset==0)
			outp(Base, Data);
		else if(Offset==2)
			outp(Base+Offset, Control);

} /* end ClearBit() */


BYTE ParallelPort::ReadBit(BYTE Bit)
{

		Status = (BYTE) inp(Base+1);

		switch (Bit)
		{
		case D0_D1:
			return (Status & BitPos[5]);
		case D2_D3:
			return (Status & BitPos[7]);
		case D4_D5:
			return (Status & BitPos[6]);
		case D6_D7:
			return (Status & BitPos[3]);
		case Ready_Busy:
			return (Status & BitPos[4]);
		}
		return 2;
} /* end ReadBit() */
```

```
void ParallelPort::ResetAddressCounter(void)
{
		ClearBit(AddrReset);
		SetBit(AddrReset);
		ClearBit(AddrReset);
}

void ParallelPort::SetAddress(WORD Address)
{
		WORD i;
		ClearBit(AddrSerialClock);
		ResetAddressCounter();

		for(i=0;i<Address;i++)
		{
			delay(1);
			SetBit(AddrSerialClock);
			delay(1);
			ClearBit(AddrSerialClock);
		}
}


void ParallelPort::SetControls(BYTE Mode)
{

if(Mode==READ||Mode==WRITE||Mode==ERASE||Mode==INVALID)
		{
			Control=Mode;
			outp(Base+2,Control);
		}
}


/* Data is changed only in WRITE mode so that ~OE of 74HC595		*/
/* is activated only when P0 is in input mode				*/
void ParallelPort::SetData(BYTE data)
{
		int i;

		if(Control==WRITE)
		{
			for(i=7; i>=0; i--)	/*Shift-in data 8 times*/
			{
				if(data & BitPos[i])
/* bit=1 */
					SetBit(SerialData);
				else
		/* bit=0 */
					ClearBit(SerialData);

				ClearBit(DataShiftClock);
				SetBit(DataShiftClock);		/* Shift-
in data */
			}
			ClearBit(DataShiftClock);
			ClearBit(DataStoreClock); /*Assuming DataStoreClock in
H state initially*/
			SetBit(DataStoreClock); /* Load output register of HC595
*/
		}
}


BYTE ParallelPort::ReadData(void)
{
		BYTE data=0;

		ClearBit(_AB);				/*Select D0 D2 D4
D6*/
		data=(ReadBit(D0_D1))? data | BitPos[0]: data & ~BitPos[0];
		data=(ReadBit(D2_D3))? data & ~BitPos[2]: data | BitPos[2];/		/
Inverted signal
		data=(ReadBit(D4_D5))? data | BitPos[4]: data & ~BitPos[4];
		data=(ReadBit(D6_D7))? data | BitPos[6]: data & ~BitPos[6];
		SetBit(_AB);				/*Select D1 D3 D5
D7*/
		data=(ReadBit(D0_D1))? data | BitPos[1]: data & ~BitPos[1];
		data=(ReadBit(D2_D3))? data & ~BitPos[3]: data | BitPos[3];/		/
Inverted signal
		data=(ReadBit(D4_D5))? data | BitPos[5]: data & ~BitPos[5];
		data=(ReadBit(D6_D7))? data | BitPos[7]: data & ~BitPos[7];

		ClearBit(_AB);
		return data;
}

/*********************************************************************/

Programmer::Programmer()
{
		Buffer=NULL; BufferPosition=0; Program[4096]='\0';
```

```
        for(int i=0; i<4096;i++) Program[i]=0xFF;
}


int Programmer::WriteProgram(char *File)
{
        WORD Address;


        if(!DetectHardware()){ErrorPrint("Can not detect Programmer");return
ERROR;}
        if(!ReadFile(File)) { ErrorPrint("Can not read file"); return ERROR; }
        if(!DecodeHexFile()) { ErrorPrint("Can not decode Hex file"); return
ERROR; }

        PrintWaitMsg();

        /* Erase Chip Flash Memory */
        {
                Port.SetBit(ALE_PROG);
                Port.SetControls(ERASE);
                Port.ClearBit(EA_Vpp);
                delay(1);
                Port.ClearBit(ALE_PROG);
                delay(12);              /*Hold ALE_PROG down at least
10ms */
                Port.SetBit(ALE_PROG);
                delay(1);
                Port.SetBit(EA_Vpp);
        }

        /* Start Write & Varify cycle */
        Port.SetBit(ALE_PROG);                                  /       *
ensure ALE_PROG is high */
        for(Address=0;Address<4096;Address++)
        {
                if(Program[Address]==0xFF) continue;
                Port.SetControls(WRITE);        /* Write mode */
                Port.SetAddress(Address);       /* Set address */
                Port.SetData(Program[Address]);     /* set data */
                Port.ClearBit(_OE);                             /       *
Enable output of HC595 */
                Port.ClearBit(EA_Vpp);          /*              Apply
Programming voltage */
                delay(1);
                Port.ClearBit(ALE_PROG);    /* Pulse ALE_PROG */
                delay(2);
                Port.SetBit(ALE_PROG);
                delay(1);
                while(Port.ReadBit(Ready_Busy)==0) {;}          /*  wait
till Busy is low */
                Port.SetBit(_OE);                               /       *
Disable output of HC595 */
                Port.SetBit(EA_Vpp);                            /       *
Remove programming voltage */
                delay(1);
                Port.SetControls(READ);
                delay(1);
                if(Program[Address]==Port.ReadData())
                        continue;
                else
                {
                        Port.SetControls(INVALID);
                        ErrorPrint("Can not varify data");
                        return ERROR;
                }
        }

        Port.SetControls(INVALID);
        PrintDone();
        return OK;
}


int Programmer::DetectHardware(void)
{
        BYTE data=0xFA;
        Port.SetControls(WRITE);
        Port.SetData(0x55);
        Port.ClearBit(_OE);
        data=Port.ReadData();
        Port.SetBit(_OE);
        Port.SetControls(INVALID);
        return (data==0x55)? OK:ERROR;
}


int Programmer::ReadFile(char *Filename)
{
        ifstream File;
        int i=0,c;
        long FileLength;

        File.open(Filename,ios::in);
        if(File.fail()) { ErrorPrint("Can not open file"); return ERROR; }

        File.seekg(0,ios::end);
        FileLength=File.tellg();
        Buffer=new char[FileLength+1];
        if(!Buffer) { ErrorPrint("Not enough memory"); return ERROR; }

        File.seekg(0,ios::beg);
        while((c=File.get())!=EOF)
                Buffer[i++]=c;
        Buffer[i]='\0';
        return OK;
}


int Programmer::DecodeHexFile(void)
{
        BYTE RecLen, RecType,Error=FALSE;
        WORD Address;
        int i,j;

        while(!Error)
        {
                GetNextRecord();
                if              (Record[0]=='\0'){ErrorPrint("Corrupt
record");Error=TRUE; continue;}
                if      (!VarifyChecksum()){ErrorPrint("Checksum
error");Error=TRUE; continue;}
                RecType=HexToByte(Record[7],Record[8]);
                if(RecType==0x01) break;                        /* End of
File */
                if(RecType==0x00)
                {
                        RecLen=HexToByte(Record[1],Record[2]);

Address=HexToWord(Record[3],Record[4],Record[5],Record[6]);
                        for(i=1,j=9;i<=RecLen;i++,Address++,j+=2)
                                if(Address<4096)

Program[Address]=HexToByte(Record[j],Record[j+1]);
                                Else
                                {
        /*Max program size 4K */
                                        Error=TRUE;
ErrorPrint("Address exceeds 4K");
                                        break;
                                }
                }
                else                                    /* Ignore other record
types */
                        {   Error=TRUE;   ErrorPrint("Extended
address"); }
        }
        delete Buffer;
        return (Error)?ERROR:OK;
}


int Programmer::VarifyChecksum(void)
{
        BYTE Checksum, DataByte;
        WORD Sum=0;
        int i,len;

        len=strlen(Record);
        Checksum=HexToByte(Record[len-2],Record[len-1]);

        for(i=1; i<len-2; i+=2)     /* sum all data bytes except : & Checksum */
                {
                DataByte=HexToByte(Record[i],Record[i+1]);
                Sum += DataByte;
                }

        if((0x100-(BYTE)Sum)==Checksum) /*2's complement of Sum ==
Checksum */
                return OK;
        else
                return ERROR;
}


void Programmer::GetNextRecord(void)
{
        int i,Pos;

        for(Pos=BufferPosition,i=0; Buffer[Pos]!='\n' && Buffer[Pos]!=NULL;
Pos++,i++)
                Record[i]=Buffer[Pos];

        BufferPosition=Pos+1;
```

```
        if(Record[0]!=':')
                Record[0]='\0';
        Else
                Record[i]='\0';
}


BYTE Programmer::HexToByte(char Hex1,char Hex2)
{
        BYTE Byte;

        Byte=HexToDecimal(Hex1);
        Byte <<=4;
        Byte |=HexToDecimal(Hex2);
        return Byte;
}

WORD Programmer::HexToWord(char Hex1, char Hex2, char Hex3, char Hex4)
{
        WORD Word, temp;
        Word=HexToDecimal(Hex1); Word <<=12;
        temp=HexToDecimal(Hex2); temp<<=8; Word |=temp;
        temp=HexToDecimal(Hex3); temp<<=4; Word |=temp;
        Word |=HexToDecimal(Hex4);
        return Word;
}

int Programmer::HexToDecimal(char Hex)
{
        int Decimal=256;

        if(Hex>='0' && Hex<='9') Decimal=Hex-'0';
        else if(Hex>='A' && Hex<='F') Decimal=Hex-55;
        else if(Hex>='a' && Hex<='f') Decimal=Hex-87;
        return Decimal;
}

/*********************************************************************/


int main(int argc,char *argv[])
{
        cout<<"\nAT89C51 Programmer V1.0 : Kulajit Aug-2002\n\n";
        if(argc<2)
        {
                cout<<"Insufficient arguments... \nUSAGE: 89C51PRG
<hex file name>\n";
                return 1;
        }
        if(argc>2)
        {
                cout<<"Too many arguments... \nUSAGE: 89C51PRG <hex
file name>\n";
                return 1;
        }
        Programmer Pg;
        if(!Pg.WriteProgram(argv[1])) return 1;          /* error */
        else return 0;                                  }
```

Readers' comments:

Q. I've assembled the construction project 'PC-based Programmer for AT89C51 Microcontroller, but it is not working. I am getting the error message "Error: Can Not Verify Data" on the screen. I loaded the DOS program (89c51prg.exe) file from CD in my Intel 486/33MHz system and observed that program LED1 is not glowing.

Also I am not getting any Vpp voltage (12V) pulse at pin 31 of the MCU (IC1) while running the program. Could you please check the program?

Sravan Kumar
Saidabad, Hyderabad

The author Kulajit Sarma replies:

It appears that your Vpp circuit is not working properly. The software is able to detect the programmer but the MCU is not getting programmed.

Check the connections between pin 2 of LPT1 and the base of transistor T1. Also check the connections between R4, T1, R5, ZD1, D2, R2, T2, R12, and R6 and the components. If everything is fine, remove the MCU and disconnect it from the parallel port.

On connecting pin 2 of the parallel port connector to GND, you should get a voltage of 11.5V<Vpp<12.5V at pin 31 of the MCU socket. If pin 2 is connected to 5V, you should get 5V at pin 31. During program execution, you should get at least a brief 12V pulse at pin 31, which can be checked by an oscilloscope. All the programs and the circuit were completely tested prior to publication in EFY. If possible, use the Windows version of the executable file as a few bugs were fixed in that version.

DTMF REMOTE CONTROL SYSTEM

REJO G. PAREKKATTIL

Remote control through the telephone line is an interesting proposition. Although this concept is not new, and control circuits based on the same were developed almost ten years back, it however became more popular with the introduction of dual-tone multi-frequency (DTMF) mode of dialing. Single-chip DTMF encoders/decoders are available today, which make the designing of such systems easy and reliable.

The DTMF remote control system described here has the following main features:

- It allows remote control of up to 12 electrical/electronic appliances through the telephone lines. The control application may vary from simple on/off operation to complex operations.
- To perform any operation through the telephone line, the user only needs to dial the required telephone number (to which the master unit is connected in parallel using a DPDT switch) in Pulse/DTMF mode and then the required device/appliance number in DTMF mode. The system automatically detects the ringing current from the exchange with the help of a ring detector and goes to off-hook state to receive the control signals (in DTMF signals).
- After the device number is dialed, the system generates a short-duration tone, which is sent back to the controlling end so that the user can know the resulting status of the controlled device.
- The system goes to off-hook state exactly after 1.5 minutes automatically. So the controlling time is limited to 1.5 minutes.
- A compact radio remote control allows the user to perform the control function within the station (in a range of about 30 metres) without the use of any telephone line. However, the use of telephone line is preferred for controlling function. The system can be disconnected from the telephone line (with the help of a simple switch) so that only the remote unit can be used.
- The system is very economical and doesn't require complex devices such as microprocessors and other programmable devices. It is best suited to home and factory applications. With a little modification, the system can be used as an automatic telephone answering unit.

Note. It is understood that the Department of Telecommunication (DoT) allows use of such remote control systems on the telephone line.

Overview

Fig. 1 shows the block diagram of the complete DTMF remote control system, which can be divided into two main sections, namely, a master unit and a remote unit. The remote unit is used to control the master unit. It comprises a DTMF encoder with a keypad and an FM transmitter to transmit the DTMF-modulated

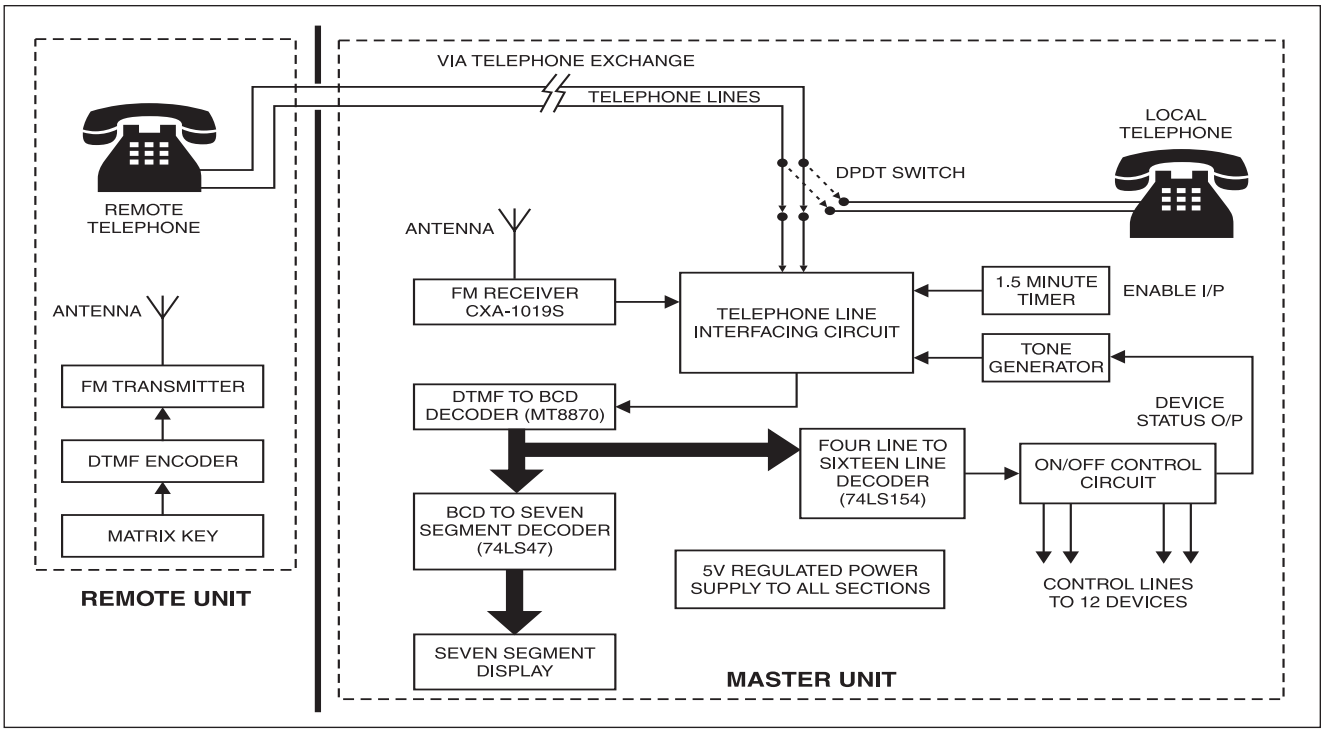


Fig. 1: Block diagram of the DTMF remote control system

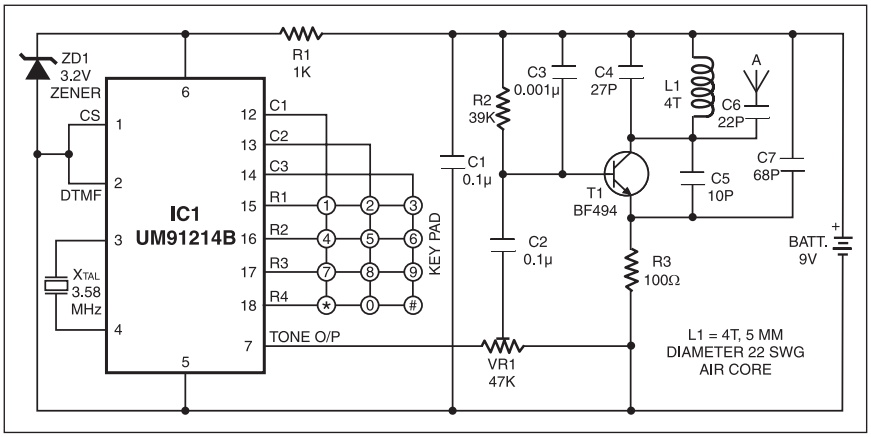


Fig. 2: Circuit diagram of remote control unit

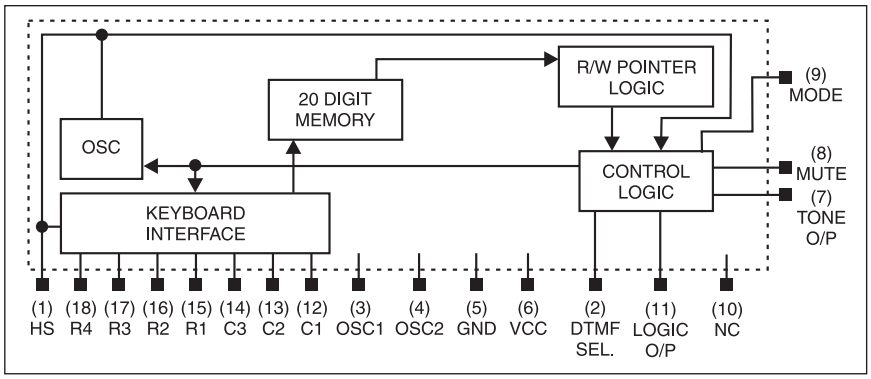


Fig. 3: Internal block diagram of IC UM91214B

FM signal corresponding to the pressed key. When remote control is done via the landlines, the telephone instru-ment itself subs-titutes for the remote control unit.

The master unit can work in conjunction with either the signal from a remote FM transmitter or a remote telephone. When FM method of transmission is used, the FM receiver comes into play to detect the DTMF tones, which are passed to DTMF-to-BCD decoder circuit via N/C (normally closed) contacts of a relay (forming part of the telephone-line interfacing circuit).

When a telephone acts as the remote control unit, the telephone-line interfacing circuit comes into operation as soon as a ring is detected. It consists of a ring detector that detects the ring from the exchange and triggers a timing circuit. The output of the timer is given to a relay driver circuit in order to simulate off-hook condition. The timer circuit maintains the telephone line in the off-hook state for 1.5 minutes on detecting a ring from the exchange and connects the telephone line to the DTMF decoding section through energised contacts of the relay.

The DTMF decoder uses IC MT8870, which forms a vital part of the circuit. It converts the dual tones to corresponding binary outputs. The 4-bit binary output of the DTMF decoder is decoded by a BCD-to-7-segment decoder that drives a 7-segment LED. A 4-to-16 line decoder (74LS154) is used to convert the 4-bit binary into 16 individual lines.

The output of the 4-to-16 line decoder is applied to the appliance on/off control circuit that consists of AND gates and D flip-flops. The output of the on/off control circuit is used to control the required devices with the help of relays.

This circuit also provides a device status output that is used to enable a tone generator. The short-duration tone thus generated is transmitted through the telephone line by the line-interfacing circuit to inform the user about the resulting status of the controlled device/appliance.

Remote unit circuit

Fig. 2 shows the circuit diagram of the remote control unit. As already mentioned, its main parts are a DTMF dialer IC UM91214B (IC1) and an FM transmitter. For any depressed key, the corresponding DTMF tone output is available at pin 7 of IC1. This tone is given as input signal to the FM transmitter wired around a high-frequency BF494 silicon transistor (T1), where it is frequency modulated by the input DTMF tones. Thus the FM transmitter transmits the carrier (around 100 MHz), frequency modulated by the DTMF dialer IC output tones.

A 9V battery is used for the remote control unit. However, the DTMF dialer IC requires only 3V for its operation, which is derived with the help of a zener diode voltage regulator.

The DTMF encoder IC UM91214B is commonly used as a dialer IC in telephones. Its function is to generate the DTMF tones corresponding to the depressed key. The internal block diagram of UM91214B is shown in Fig. 3.

For its time base the UM91214B requires a quartz crystal of 3.58 MHz, which is connected between pins 3 and 4 of the IC to form part of an internal oscillator. The oscillator output is converted into appropriate DTMF signals through frequency division and mixing by the control logic.

The keyboard interfacing

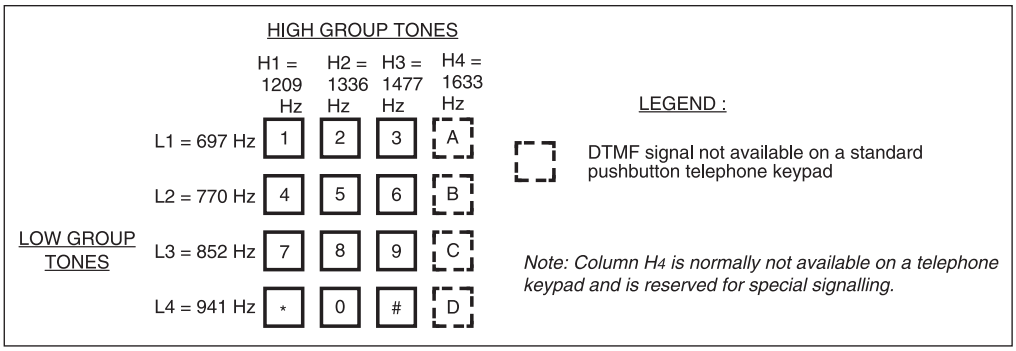


Fig. 4: Tones associated with keys on telephone DTMF keypad matrix

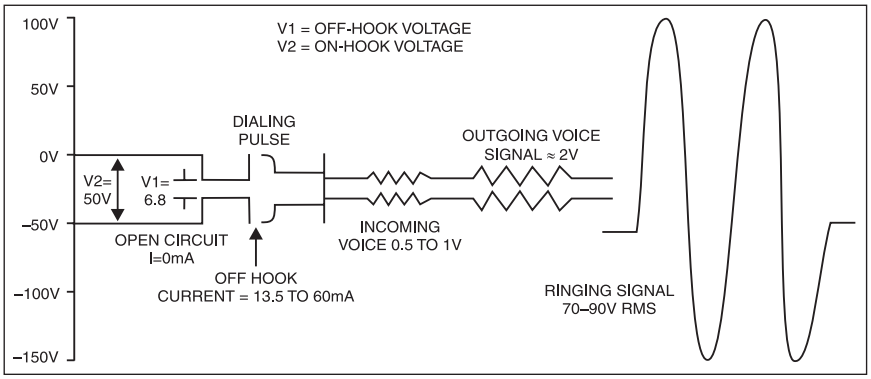


Fig. 5: Different telephone line conditions

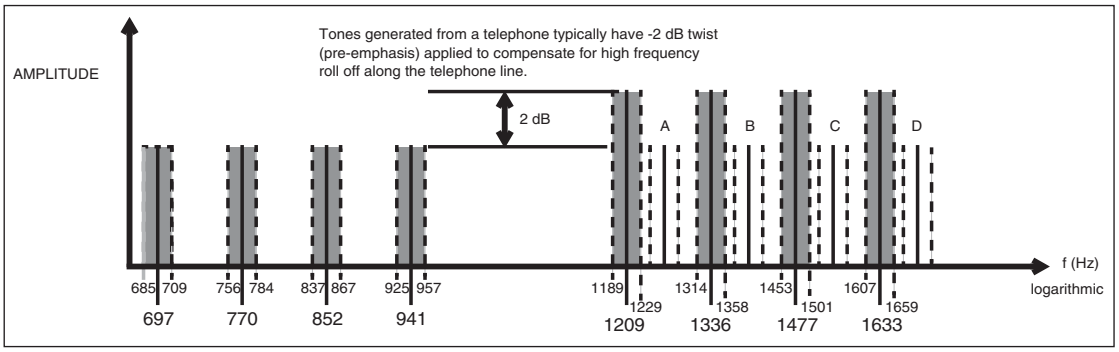


Fig. 6: Standard DTMF frequency spectrum ± (1.5%+2 Hz). Second harmonics of the low group (possibly created due to a non-linear channel) fall within the passband of the high group (indicated by A, B, C, D). This is a potential source of interference.

section interfaces the matrix type keyboard with the control logic. Pins 15 through 18 are row pins and pins 12 through 14 are column pins. Up to 12 switches are possible with this key array. They represent digits 1 through 9, 0, and symbols * and # (used for special functions). To find out the dual tones associated with each digit, refer to Fig. 4. You can easily read the low and high group tones associated with each key. The fourth column corresponding to 1633Hz frequency is not applicable to IC UM91214B.

IC UM91214B also incorporates a 20-digit dialed number memory. This feature of the IC is not used in the present remote control system. The memory unit and read/write pointer logic is controlled by the control logic. The DTMF tones are obtained from pin 7 of the IC. The IC also has some control

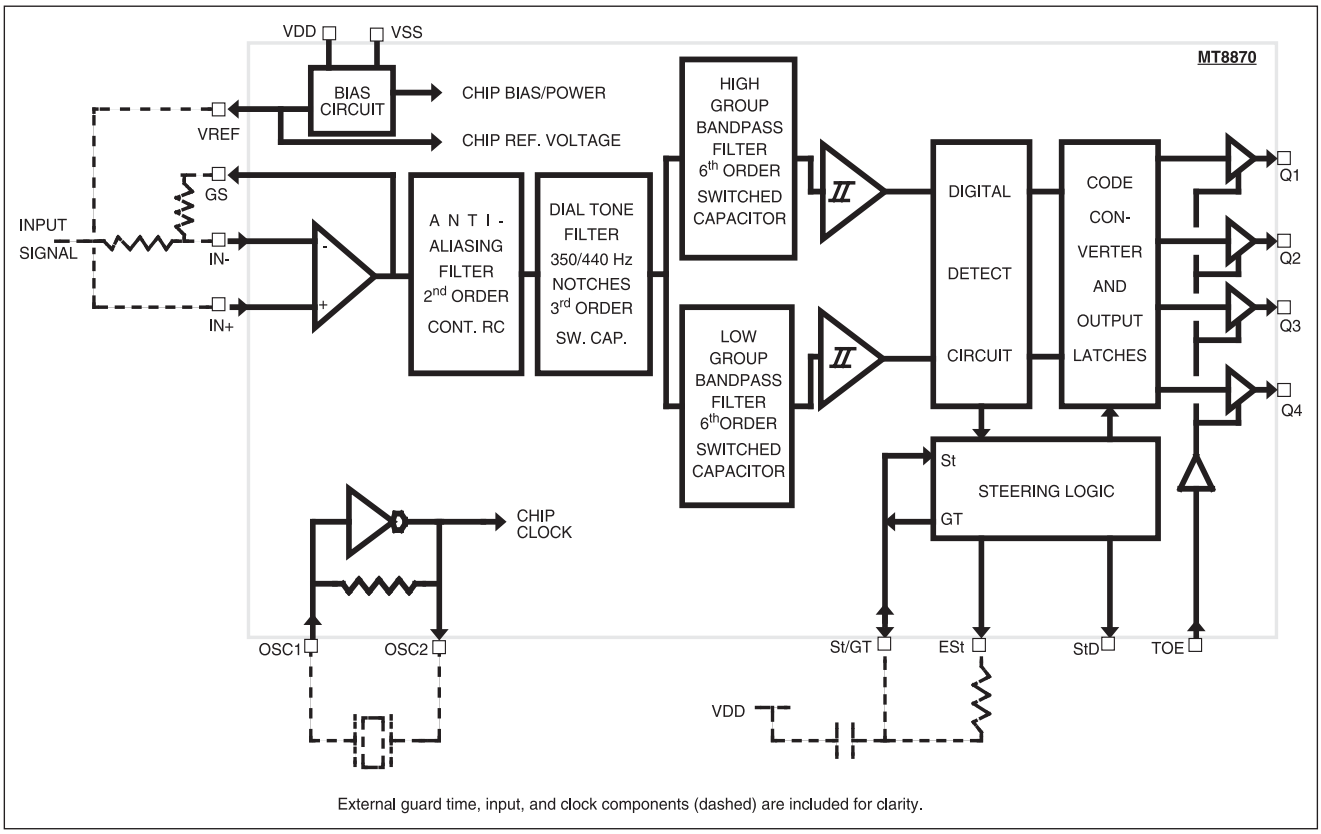


Fig. 7: Functional block diagram of IC MT8870

PARTS LIST

Semiconductors:

IC1	- MT8870 DTMF decoder
IC2	- 74LS47 BCD-to-7-segment decoder/driver
IC3	- 74LS154 4-to-16 line decoder/ demultiplexer
IC4	- PC817 optocoupler
IC5, IC6	- NE555 timer
IC7, IC8	- 74LS08 quad 2-input AND gate
IC9, IC10	- 74LS74 dual-'D' flip-flop
IC11	- 7805 5V regulator
T1, T3-T8	- BC548 npn transistor
T2, T9	- CL100 npn transistor
D1-D9	- 1N4001 rectifier diode
D10-D13	- 1N4007 rectifier diode

Resistors (all ¼-watt, ±5% carbon, unless stated otherwise):

R1	- 12-kilo-ohm
R2	- 220-kilo-ohm
R3	- 220-ohm 0.5W
R4	- 82-kilo-ohm
R5, R10, R12, R37	- 1-kilo-ohm
R6	- 5.6-kilo-ohm
R7, R14, R15, R25-R28	- 1.5-kilo-ohm
R8, R13	- 100-kilo-ohm
R9	- 330-kilo-ohm
R11	- 1.2-kilo-ohm
R16, R17	- 390-kilo-ohm
R18-R24	- 560-ohm
R29-R32	- 27-kilo-ohm
R33-R36	- 33-kilo-ohm

Capacitors:

C1, C3	- 0.22μF polyester
C2, C6	- 1μF, 25V electrolytic
C4	- 100μF, 25V electrolytic
C5, C9	- 0.01μF ceramic disk
C7	- 10μF, 25V electrolytic
C8	- 220μF, 25V electrolytic
C10	- 0.1 μF ceramic disk
C11-C14	- 2.2μF, 25V electrolytic
C15	- 1000μF, 35V electrolytic

Miscellaneous:

DIS1	- LT542 common-anode 7-segment display
	- FM receiver plate based on Sony CXA1019S
RL1	- 12V, 285-ohm 2C/O (OEN make, series 58 type 2C)
RL2	- 12V, 285-ohm 1C/O (OEN make, series 58 type 1C)
X1	- 230V AC primary to 0-12V, 1A secondary transformer
Xtal	- 3.58MHz crystal

Remote FM transmitter

IC1	- UM91214B telephone dialer
T1	- BF494 npn RF transistor
ZD1	- Zener 3.2V, 0.5W
R1	- 1-kilo-ohm
R2	- 39-kilo-ohm
R3	- 100-ohm
VR1	- 47-kilo-ohm preset
C1, C2	- 0.1μF ceramic disk
C3	- 0.001μF ceramic disk
C4	- 27pF ceramic disk
C5	- 10pF ceramic disk
C6	- 22pF ceramic disk
C7	- 68pF ceramic disk
L1	- 4T, 22SWG on 5mm air core
Xtal	- 3.58MHz crystal
	- 9V battery

inputs that are not used in its present application.

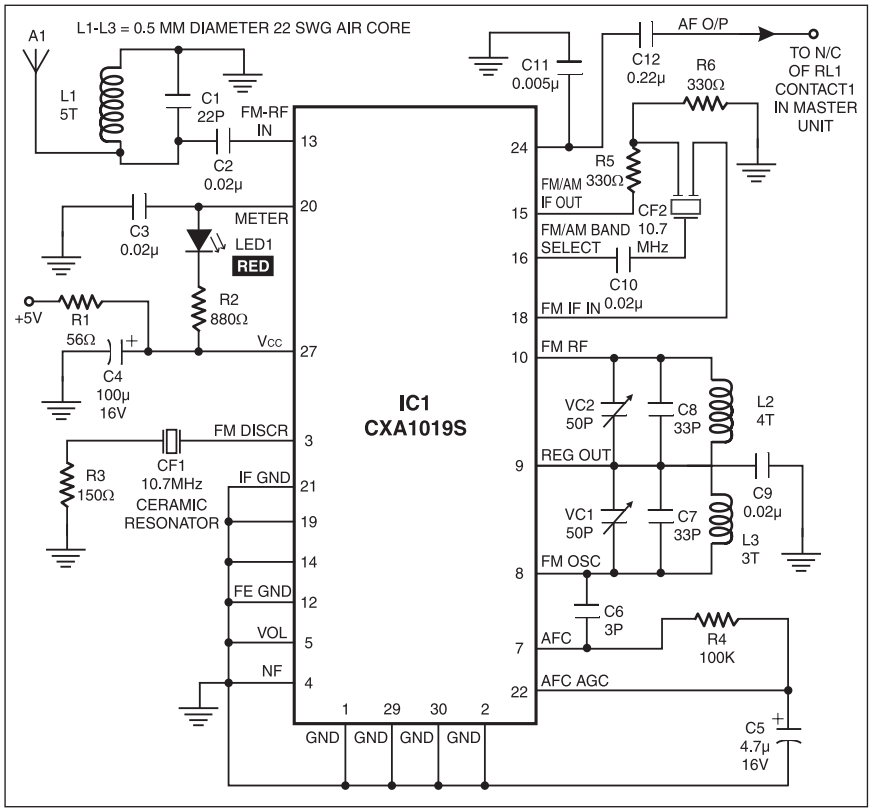


Fig. 8: Circuit diagram of FM receiver using CXA1019S

Master unit

Before describing the integrated working of the master unit, it will be appropriate to gain some useful knowledge about the telephone line, DTMF dialing, decoder MT8870, and the FM receiver using Sony CXA1019S and other sections.

Basic telephone line. A telephone line basically carries voice and various signaling information between the subscriber telephone instrument and the exchange. Suitable protection circuitry on both ends of the line protects the exchange equipment and the telephone instrument against damage from lightening, high-voltage transients, and polarity reversal. Signaling information is required to inform the subscriber and exchange about on-hook, off-hook, busy/not busy, and out-of-order conditions of the telephone/line. The ringing signal from the exchange to the subscriber is of 70-90V RMS, 20-25Hz.

The outgoing signaling refers to signals reaching the exchange from the subscriber's telephone, indicating on-hook, off-hook, hang-up, dialing, etc. Outgoing signals can be of two types: line signaling and register signaling. Line signaling encompasses on-hook, off-hook, and hang-up state signals, while register signaling refers to dialing, wherein digits of the destination or the called party are passed on to the exchange for establishing a connection.

When the telephone is in on-hook condition, the cradle switch is in open condition. There is no flow of current in the telephone circuit. When the telephone handset is lifted off the cradle, the cradle switch closes to form a closed-loop circuit with the exchange battery and the telephone circuit. This circuit is also referred to as the local loop circuit. Exchange battery voltage is typically 48 volts. The loop current is used by the exchange to establish off-/on-hook status of the telephone. (***EFY***. If the loop current is 13.5 mA to 60 mA the exchange detects it as off-hook condition, and if the loop current is less than 7.5 mA the exchange interprets it as on-hook condition.)

Different line conditions are depicted in Fig. 5. The open-circuit line voltage is about 50V DC. Incoming voice voltage to the telephone instrument varies from 0.5V to 1V and the maximum outgoing voice voltage is about 2V RMS. The ringing signal is 70-90V RMS at 20-25 Hz. In pulse dialing telephones, register signaling is known as DC loop signaling. In this case, the dialed number is conveyed to the exchange by 'make' and 'break' of the loop circuit.

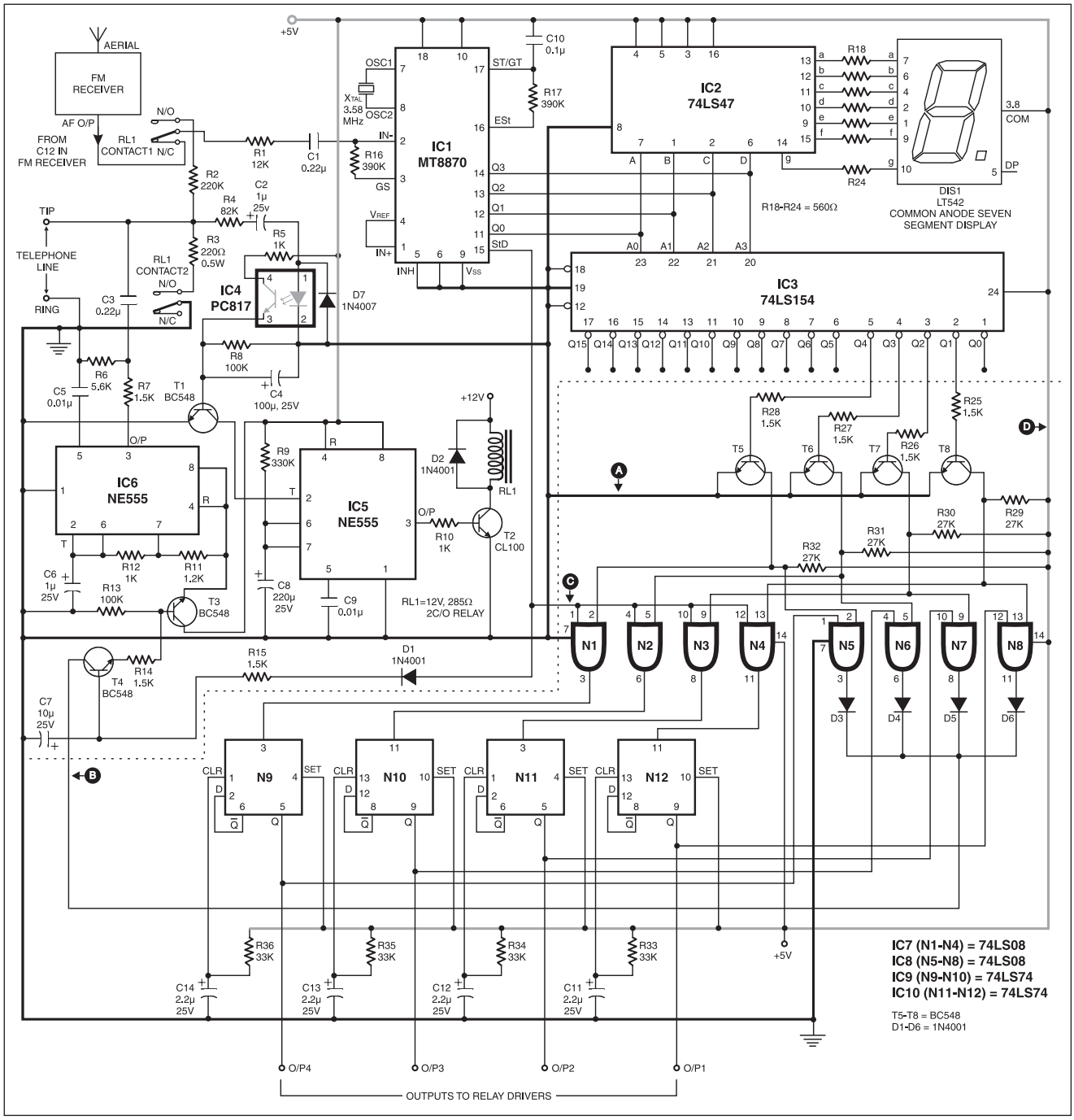


Fig. 9: Circuit diagram of master unit

DTMF signaling. AC register signaling is used in DTMF telephones. Here, tones, rather than make/break pulses, are used for dialing. Each dialed digit is uniquely represented by a pair of sinewave tones. These tones (one from low group for row and another from high group for column) are sent to the exchange when a digit is dialed by pushing the key. These tones lie within the speech band of 300 to 3400Hz, and are chosen so as to minimise the possibility of any valid frequency pair existing in the normal speech simultaneously. Actually, this minimisation is made possible by forming pairs with one tone from the higher group and the other from the lower group of frequencies. The DTMF spectrum is shown in Fig. 6.

A valid DTMF signal is the sum of two tones, one from a lower group (697-941 Hz) and the other from a higher group (1209-1663 Hz). Each group contains four individual tones. The DTMF dialing scheme is shown in Fig. 4. This scheme allows 16 unique combinations. Ten of these codes represent digits 1 through 9 and 0. The remaining six digits are reserved for special-purpose dialing.

Tones in DTMF dialing are so chosen that none of the tones is harmonic of any other tone. Therefore there is no chance of distortion caused by harmonics. Each tone is sent as long as the key remains pressed.

The DTMF coding scheme ensures

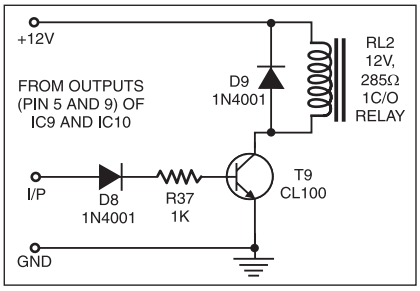


Fig. 10: Relay driver circuit

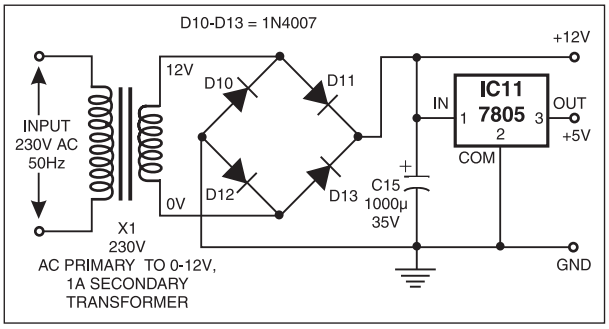


Fig. 11: Power supply circuit

that each signal contains only one component from each of the high and low groups. This significantly simplifies decoding because the composite DTMF signal may be separated with band-pass filters into single frequency components, each of which may be handled individually. As a result, the DTMF coding scheme is a flexible signaling scheme with high reliability, hence motivating innovative and competitive decoder design.

Inside MT8870. The MT8870 is a single-chip DTMF receiver incorporating switched capacitor filter technology and an advanced digital counting/averaging algorithm for period measurement. The functional block diagram of Fig. 7 depicts the internal working of this device.

The DTMF signal is first buffered by an input op-amp that allows adjustment of gain and choice of input configuration. The input stage is followed by a low-pass RC active filter, which performs anti-aliasing function. Dial tone at 350 and 440 Hz is then rejected by a third-order switched capacitor notch filter. The signal is still in its composite form and is split into its individual components by two 6th- order switched capacitor band-pass filters. Each component is smoothed by an output filter and squared by a hard limiting comparator. The two resulting rectangular waveforms are then applied to a digital circuit, where a counting algorithm measures and averages their periods. An accurate reference clock is derived from an inexpensive external 3.58MHz crystal.

The time required to detect a valid tone pair, t_{DP}, is a function of decode algorithm, tone frequency, and the previous state of the decode logic. ESt (early steering output) indicates that two tones of valid frequency have been detected and initiates an RC timing circuit. If both tones are present for a minimum guard time, determined by an external RC network, the DTMF signal is decoded and the resulting data is latched on the output register. The delayed steering output (StD) is raised to indicate that new data is available. The output corresponding to each key pressed is shown in the truth table.

MT8870 Output Truth Table

F_{LOW}	F_{HIGH}	KEY	TOE	Q4	Q3	Q2	Q1
697	1209	1	1	0	0	0	1
697	1336	2	1	0	0	1	0
697	1477	3	1	0	0	1	1
770	1209	4	1	0	1	0	0
770	1336	5	1	0	1	0	1
770	1477	6	1	0	1	1	0
852	1209	7	1	0	1	1	1
852	1336	8	1	1	0	0	0
852	1477	9	1	1	0	0	1
941	1209	0	1	1	0	1	0
941	1336	*	1	1	0	1	1
941	1477	#	1	1	1	0	0
697	1633	A	1	1	1	0	1
770	1633	B	1	1	1	1	0
852	1633	C	1	1	1	1	1
941	1633	D	1	0	0	0	0
—	ANY		0	Z	Z	Z	Z

FM receiver. The circuit diagram of the FM receiver using CXA1019S is shown in Fig. 8. In the circuit, L1 and C1 form the tank circuit for producing oscillations for the RF stage. L3, C7, and variable capacitor (VC1) form the tank circuit for the local oscillator. A 10.7MHz ceramic filter (CF2) is used to separate the intermediate frequency of about 200 kHz bandwidth. The audio output is available from pin 24 of the IC through coupling capacitor C12. LED1 is used for fine-tuning indication. A +5V is applied to pin 27 of the IC through current-limiting resistor R1.

Integrated working of master unit

The circuit diagram of the master unit is shown in Fig. 9, with FM receiver, relay driver, and power supply portions shown in Figs 8, 10, and 11, respectively.

The master unit receives DTMF signal transmitted by the remote unit as well as from the telephone line, decoding it into binary form, displaying the received number on a 7-segment display and performing the on/off control of the connected devices according to the received signal.

The FM receiver built around the popular Sony chip CXA1019S works with a +5V power supply, which is derived from a +5V voltage regulator (IC 7805). The AF output of the FM receiver goes to the normally-closed contact of relay RL1. The pole of the relay is connected to the IN terminal of the DTMF decoder (MT8870). Hence in normal condition (when RL1 is not energised), the signals available for the decoder are only from the FM receiver.

Telephone line interface circuit. When a ringing voltage from the exchange comes via the telephone lines, capacitor C2 bypasses the AC ringing current so that the LED in the optocoupler glows, turning on the internal transistor of the optocoupler. (Diode D7 in anti-parallel to internal LED of the optocoupler provides conductive path during negative half cycles of the ringing current.) As a result, transistor T1 gets forward biased and it conducts, pulling its collector towards ground. This, in turn, triggers the monostable multivibrator wired around IC5 (NE555). Once triggered, the output of the monostable multivibrator at pin 3 goes high (for about 1.5 minutes), turning on relay driver transistor T2 and thereby energising relay RL1.

When the relay is turned on, the ring ceases and the DC voltage across the telephone lines is reduced to around 10V due to introduction of voltage-dropping resistor R3 across the telephone lines via contact 2. Then the audio signals on the telephone lines are extended to the decoder through resistor R2 and N/O contacts of pole 1 of relay RL1. The circuit returns to its normal condition when the output of

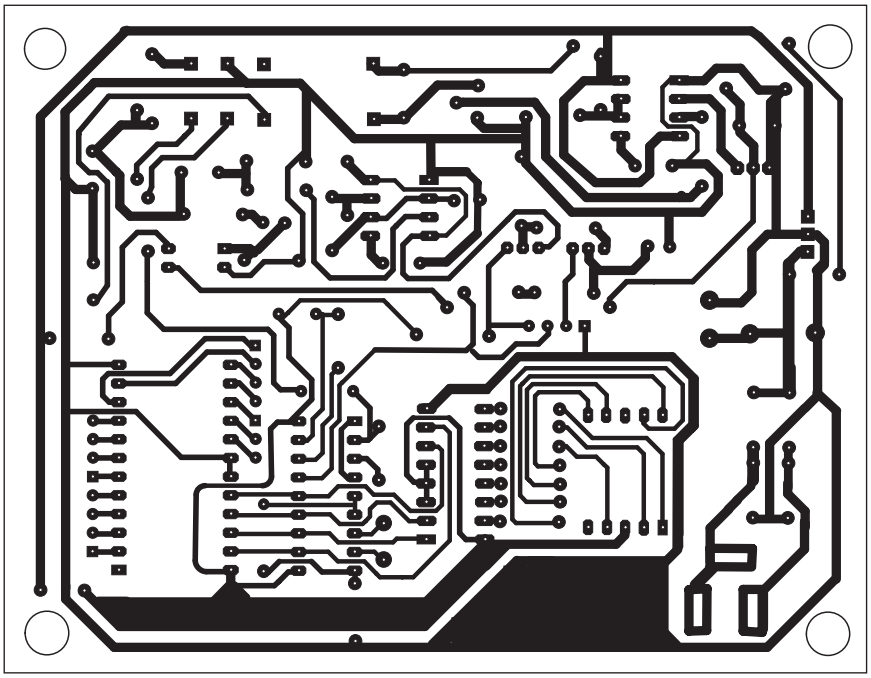


Fig. 12: Actual-size, single-side PCB layout for the master unit

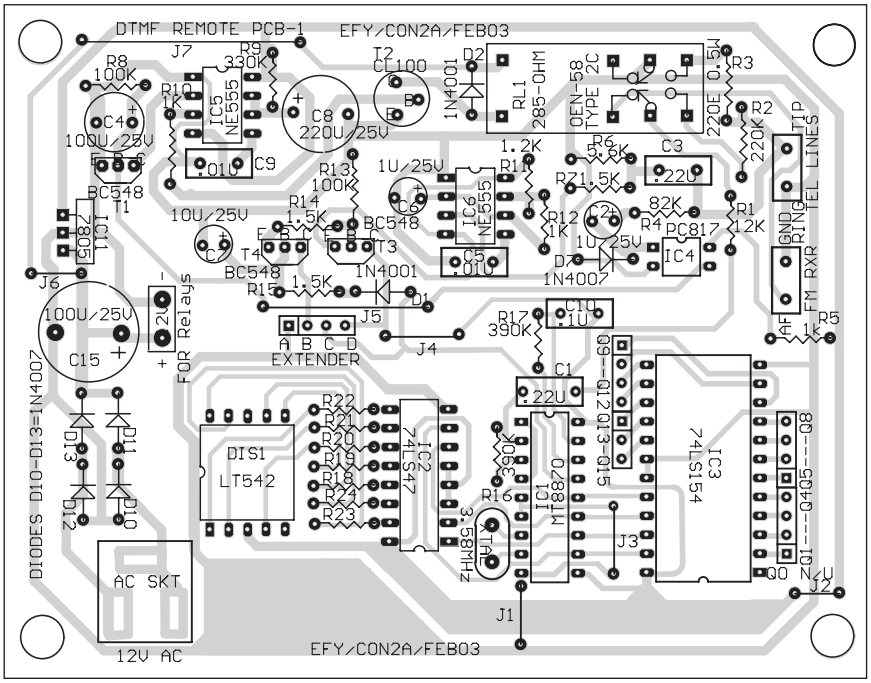


Fig. 13: Component layout for the PCB

the monostable multivibrator goes low after a time period determined by the external R9-C8 combination.

Tone decoder. After receiving a valid DTMF tone, the DTMF decoder (MT8870) places the corresponding binary number on its output terminals and the delayed steering output (StD) goes high to show that the new data is available. The duration of the delayed steering output is equal to the duration of the received DTMF signal. The working of the DTMF receiver has already been explained.

Displaying the dialed digit. The binary outputs of the decoder are connected to 7-segment display decoder/driver 74LS47 (IC2). The 7-segment decoder/driver decodes the binary output of the DTMF decoder to drive a 7-segment LED. This display indicates the dialed number.

Appliance on/off control circuit. The on/off control circuit is used to provide toggle outputs that may be used to control relays, whose contacts may be used to switch on/off the connected appliances. This circuit includes a 4-to-16-line decoder, D flip-flops wired as toggle flip-flops, and eight AND gates. It also senses the condition of a selected toggle output and enables a tone generator circuit (wired with 555 timer IC) accordingly. The output of the tone generator is transmitted via the telephone line to inform the user about the resulting status of the controlled device.

The 4-bit binary output of the DTMF decoder (MT8870) is applied to the 4-to-16 line decoder IC 74LS154. The active-low outputs of this decoder are converted into active-high states by inverter circuits wired around transistors T5 through T8.

The output of the 4-to-16 line decoder goes to two sets of AND gates. AND gates N1 through N4 are used for generating a clock pulse for D flip-flops. The other input for AND gates N1 through N4 is the StD output of the DTMF decoder IC. AND gates N5 through N8 are used for deriving a device status output. The output of an AND gate (out of N5 through N8) is high if the Q output of the corresponding D flip-flop is high. Hence the outputs of these AND gates reflect the output conditions of the corresponding flip-flops and can be used as the device status output.

The status outputs of all flip-flops are OR-ed together and the resultant output is used as the device status output for the selected device. This output is applied to a time delay circuit consisting of a single transistor stage, whose output is used to enable a tone generator circuit.

D flip-flops in the circuit are wired as toggle flip-flops. Power-on-reset is provided to all D flip-flops with the help of RC network comprising resistors R33 through R36 and capacitors C11 through C14. Their Q outputs are returned to D inputs. Thus for each positive-going trigger pulse applied at the clock input, the Q output toggles. The Q output of each flip-flop is connected to a relay driver circuit that drives a relay.

Although we have 12 useful outputs (from 4-to-16 line decoder IC3) corresponding to 12 buttons of the keypad, here only four outputs corresponding to digits 1 through 4 of the keypad have been wired. If required, the other eight outputs can also be wired in an identical manner to control 12 relays. The device status out-

puts of all the circuits are OR-ed together by diodes D3 through D6 to obtain a single status output for the selected device.

Tone delay circuit. In the DTMF remote control system, there is a facility for the user to know the resulting status of the controlled device/appliance through the telephone line. This is achieved by sending an audio tone after a particular device is switched on and sending no tone after a particular device is switched off.

The frequency of the tone sent through the telephone line is about 650 Hz, which is well suited to transmission over the telephone line. But still there is a chance for these tones to get mixed with the DTMF control tones, resulting in unexpected results. To avoid this problem, the tone is sent only for a short duration determined by a tone control circuit.

The tone control circuit is a simple timer built using a pair of transistors and a few passive components. It is implemented between the device status output of the appliance-on/off control circuit and the tone generator circuit. The circuit actually works as a short-duration power supply switch to the tone generator circuit (built around IC6).

The status output from cathode junction of diodes D3 through D6 is extended to the collector of transistor T4. Now, when the StD output goes high (on receipt of a valid DTMF digit), capacitor C7 charges through R15. When the voltage across it exceeds 0.7V, the transistor pair of T3 and T4 is switched on, providing +5V power supply to tone generator IC6 (NE555). When the StD output goes low, C7 starts discharging. When the voltage across it falls below 0.7V, T4 is turned off and the power supply to tone generator IC6 is cut off.

The output of the astable multivibrator (about 650 Hz) is reduced in voltage by the potential divider network comprising resistors R6 and R7 and injected into the telephone line through capacitor C3. Thus a short-duration tone is transmitted over the telephone line for each turning on of a device. If the device status output is not high, the transistors don't turn on and therefore the tone generator doesn't produce any tone.

The relay driver circuit. As mentioned earlier, the outputs of D flip-flops are connected to the relay driver circuit (Fig. 10). The relay driver circuit consists of a medium-power transistor wired in the switching mode. When an input voltage of sufficient magnitude is applied to the base of the transistor, the transistor goes into saturation, turning on the relay connected at its collector terminal. A diode is used as a free wheeling element to prevent the induced voltage in the relay coil from damaging the transistor when relay driver transistor is cut-off.

Power supply. The power supply circuit (Fig. 11) consists of a bridge rectifier with shunt capacitance filter. A 5V regulated source is used for the entire circuit, as all ICs belong to TTL family. Three-terminal voltage regulator IC 7805 is used to provide 5V supply. To improve the current-handling capacity and to prevent the thermal runaway, these ICs are provided with heat-sinks for sufficient cooling. Due to the limitations in the current capacity of L7805 series, separate ICs are used for the master unit, on/off control unit, and FM receiver.

Assembly and testing

The master unit (except the appliance on/off control circuit), along with power supply circuit, may be assembled on a single PCB. An actual-size, single-side PCB for the same is shown in Fig. 12 and its component layout in Fig. 13. The appliance on/off control circuit (below dotted line in Fig. 9) can be assembled on a separate PCB for operation of required number of devices. Connectors have been provided on the PCB for extending circled points A through D (Fig. 9) and outputs of IC3 (74LS154) for wiring up the control circuit for as many appliances as desired by the reader. Since requirement for control of number of appliances may vary from one user to the other, hence no PCB for the same is included here.

The remote transmitter circuit is fairly simple and the same may be assembled on a general-purpose PCB, while the readily-available FM plate using Sony CXA1019S may be procured from the local market and integrated with the main PCB for master unit.

To test the remote control, tune an FM radio to its frequency and check whether the correct tones are heard when the corresponding keys are depressed. Then tune the trimmers on the FM receiver (in the master unit) for maximum brightness of the tuning indicator LED. (***Note.*** For this adjustment, the remote transmitter unit must be kept on nearby with any one key depressed.) Then check whether the seven segments indicate the dialed number.

If everything goes right, connect the master unit to the telephone line. Then trigger IC5 by applying a small pulse to its pin 2. (You can do this by just touching pin 2 with a screwdriver momentarily.) The relay should get activated. Now check whether the 7-segment display correctly the number dialed on a telephone (in tone mode) connected in parallel. Also check whether the output of the appliance on/off control circuit toggles with each successive dialing of the same number. If everything is alright, the device is ready for use. Connect the relay drivers to the outputs and determine what appliances are to be controlled by your DTMF remote control system.

Conclusion. Remote control is necessary in many situations. Using the DTMF remote control system, you can turn on or off the water pump, washing machine, garden light, or any other electrical appliance in your house with just a telephone call. The system is particularly suitable for use at homes. It is simple to operate and is user-friendly. The FM remote control unit provides an added advantage since it is omnidirectional unlike IR remote controls.

Further enhancements. The present DTMF remote control system is designed to switch on/off four devices. However, it can be modified to control up to 12 devices by adding some more components.

The circuit can be developed to perform time-varying operations with use of an additional control circuit. Thus the circuit may be used to increase/decrease the volume of a record player or to rotate a stepper motor with suitable control circuits.

An automatic telephone-answering unit can be formed in association with the circuit by adding suitable voice-record/playback chips.

The prominent drawback of the circuit is that it has no security to prevent the operation of the circuit by strangers over the telephone line. A password-lock circuit can be added to the circuit for ensuring security. ❑

Readers' comments:

Q. We are doing 'DTMF Remote Control System' project. Please confirm whether we can use IC CM8870P1-0218-B in lieu of MT8870. Further, is there any problem in the circuit as the relay changes its position as soon as the supply is switched on?

K.D. Dheesh
Through e-mail

EFY: The CM8870 with the stated suffix is equivalent to MT8870. The suffixes normally relate to the type of package such as plastic, ceramic, surface mount or PLCC type etc. As regards energisation of relay due to switching-on noise, please note that this IC has a very good sensitivity starting from –29dBm (equivalent to 27.5mV RMS) and hence the switching noise must be kept as small as possible by proper decoupling and use of well regulated supply, else the noise may penetrate the signal input pins via common power supply impedance. Problem was not observed in the author's prototype extensively tested at EFY Lab.

PROGRAMMABLE LOGIC CONTROLLER

SANTHOSH JAYARAJAN

Here's a freely programmable system to control outputs based on the status of the inputs. In an industrial setting such a controller is called programmable logic controller (PLC). The accompanying software program has been written and compiled using Borland C++ version 5.2. Worked out examples have been included to demonstrate the practical use of the programmable logic.

The computer printer port, commonly called LPT1, offers possibilities for reading inputs and sending outputs. It provides five inputs, which can be read, and eight outputs, which can be used to control various appliances.

Hardware interface

Pin diagram of the printer's parallel port is shown in Fig. 1. Prefix letters D, S, and C denote data, status, and control register bits, respectively. Bits of the three registers indicated by empty boxes are not available externally. Binary weight, bit position, and the corresponding pin numbers of Fig. 1 are summarised in Table I.

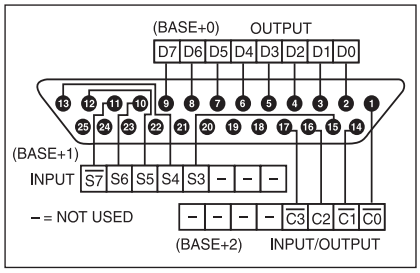


Fig. 1: Parallel port D connector pin details

As shown in the table, D0 through D7 are the outputs, which are extended via pins 2 through 9 of the 25-pin D connector. These pins output approx. 5V (logic 1) signal when energised by the outp command in C++. If the outp command sends a value of 1, the D0 output bit goes high (5V). Similarly, for a value of 128, D7 bit goes high. All other output bits/pins, individually or in combination, can be made high by outputting a decimal number equivalent to sum of the binary weight of the individual data bits; for example, to turn on D4 and D2 bits, output a value of 4 + 16 = 20 (decimal).

Bits S3, S4, S5, S6, and $\overline{S7}$ (inverted internally) serve as inputs. The inputs, except S7, are normally at logic 1. Thus when shorted to pin 25 (ground), these will change state. The program senses an input when it is shorted with pin 25 (Gnd). Normally, the input devices in systems using PLCs are pushbutton switches, proximity switches, limit switches, etc.

Any input to the PC is used in the user program to start a sequence of operations or control one/more outputs in a way programmed by the user. The inputs are all potential-free, meaning that these are contacts of pushbutton or proximity switches/relay contacts having no voltage. Only the input S7 is inverted and adequate code has been programmed in the software to take care of this. The five available inputs can be used altogether or as required. Each input carries with it a binary weight of 8 to 128, as shown in Table I.

The eight outputs are connected via optocouplers to the output relays, whose contacts can be used to control motors, solenoids, lamps, or other output devices. The current drawn from the outputs is limited by the current capacity of the output relay contacts. It is typically 5 amp.

Screenshot of opening menu

The complete schematic diagram of the interface circuit is shown in Fig. 2. The inputs are taken to the respective pins on the D connector directly, while the outputs from D connector are isolated from the load using hex buffers CD4050B (IC1 and IC2) and optocouplers (IC3 through IC10). The output from each of the optocoupler is used to activate a relay driver circuit comprising SL100 medium-power transistor, which, in turn, energises a 12V relay, the contacts of which can be used to turn on the output device such as a motor or solenoid. Although all the eight outputs and all the five inputs can be used, we've used here only four of the five possible inputs.

A single-side, actual-size PCB for the circuit, including the power supply, is shown in Fig. 3 with its component layout in Fig. 4. The step-down power supply transformer and relays have to be installed externally.

TABLE I: Summary of PC Parallel Port Pins

Binary weight:	128	64	32	16	8	4	2	1
Output bit (pin):	D7 (9)	D6 (8)	D5 (7)	D4 (6)	D3 (5)	D2 (4)	D1 (3)	D0 (2)
Input bit (pin):	$\overline{S7}$ (11)	S6 (10)	S5 (12)	S4 (13)	S3 (15)	—	—	—
Control bit (pin):	—	—	—	—	$\overline{C3}$ (17)	C2 (16)	$\overline{C1}$ (15)	$\overline{C0}$ (14)

Notes. *1. A bar over specific bits indicates that these bits are inverted (inside the PC) before being read by the PC.*
2. The control port, which is an input/output port, has not been used in the present application.
3. Pins 18 through 25 are shorted to ground in the PC.

Software

The software is the heart of the system. It is designed to:

- Access a line editor where the user program can be entered or edited.

- Provide a display of the status of inputs and outputs.
- Support programmable timers, counters, output latches, and real-time clock functions.

The opening menu of the program is shown in screenshot above. It provides the following options:

1. F1 to run a logic program. Enter the file name (containing the PLC program) and press F1 again to start running the program or press ESC to terminate and exit the program.

2. F2 to view and edit a program. Enter a file name and edit or write a fresh program.

3. F3 to view input /output status. Check the inputs once they are connected.

4. F4 to force outputs. Turn on any particular output to check if the output devices are working. A 500ms delay is required before turning off a device.

5. F5 for online help on all options available.

6. F6 to exit the program.

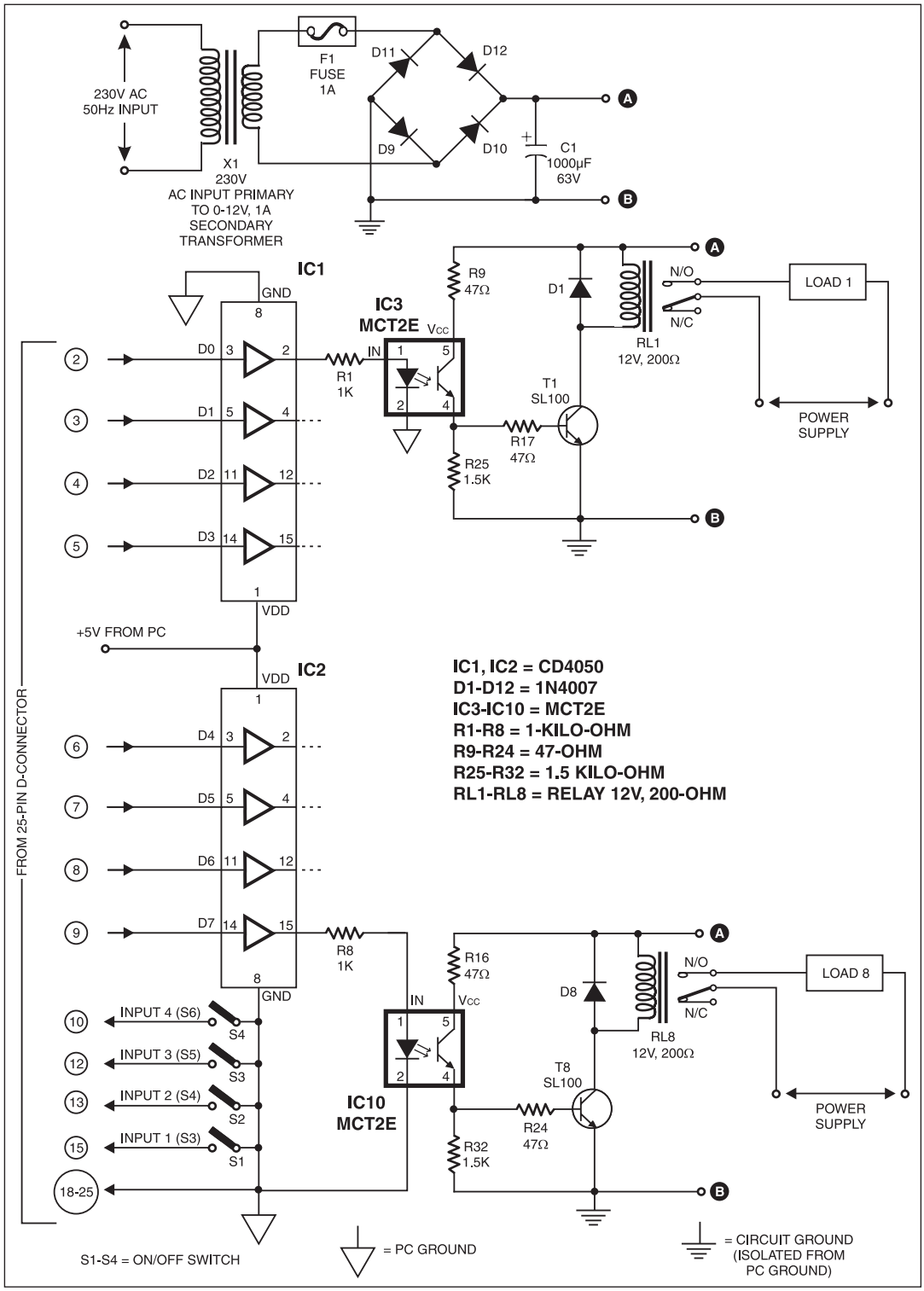


Fig. 2: Complete circuit of the PC interface

How to write and run a logic program

Before writing a program, you need to learn the following fundamentals:

- Usually, the inputs are of normally open (N/O) or normally closed (N/C) types. A N/O contact is electrically open until actuated, while a N/C contact will be electrically closed until actuated.
- Any input or output can have a complimentry contact that can be used in the program. For example, input I1 will have a complimentry contact NI1, which will be closed when I1 is open, and vice versa. Use of prefix N defines a complimentry input or output contact.

The various program elements like inputs, outputs, timers, counters, and real-time outputs are explained below. Each input, output, or program element has an associated program code number (refer Table II), which will be used in the program.

Inputs. The system can accept five inputs (I1, I2, I3, I4, and I5) from the field. These can be pushbuttons, limit switches, proximity switches, etc. The compliments of these inputs are termed

TABLE II: Program Elements With Codes

Input/output timer, counter	Programming code	Remarks
I1	1	Input number 1
I2	2	Input number 2
I3	3	Input number 3
I4	4	Input number 4
I5	5	Input number 5
NI1	6	Inverse of input number 1
NI2	7	Inverse of input number 2
NI3	8	Inverse of input number 3
NI4	9	Inverse of input number 4
NI5	10	Inverse of input number 5
O1	11	Output number 1
O2	12	Output number 2
O3	13	Output number 3
O4	14	Output number 4
O5	15	Output number 5
O6	16	Output number 6
O7	17	Output number 7
O8	18	Output number 8
NO1	19	Inverse of output number 1
NO2	20	Inverse of output number 2
NO3	21	Inverse of output number 3
NO4	22	Inverse of output number 4
NO5	23	Inverse of output number 5
NO6	24	Inverse of output number 6
NO7	25	Inverse of output number 7
NO8	26	Inverse of output number 8
CDN1	27	Counter number (up) 1 done
CDN2	28	Counter number (up) 2 done
CDN3	29	Counter number (up) 3 done
CDN4	30	Counter number (up) 4 done
CDN5	31	Counter number (up) 5 done
NCDN1	32	Inverse of counter (up) number 1 done
NCDN2	33	Inverse of counter (up) number 2 done
NCDN3	34	Inverse of counter (up) number 3 done
NCDN4	35	Inverse of counter (up) number 4 done
NCDN5	36	Inverse of counter (up) number 5 done
CODN1	37	Counter (down) number 1 done
CODN2	38	Counter (down) number 2 done
CODN3	39	Counter (down) number 3 done
CODN4	40	Counter (down) number 4 done
CODN5	41	Counter (down) number 5 done
NCODN1	42	Inverse of counter (down) number 1 done
NCODN2	43	Inverse of counter (down) number 2 done
NCODN3	44	Inverse of counter (down) number 3 done
NCODN4	45	Inverse of counter (down) number 4 done
NCODN5	46	Inverse of counter (down) number 5 done
TEN1	47	Timer 1 enable
TEN2	48	Timer 2 enable
TEN3	49	Timer 3 enable
TEN4	50	Timer 4 enable
TEN5	51	Timer 5 enable
NTEN1	52	Inverse of timer 1 enable
NTEN2	53	Inverse of timer 2 enable
NTEN3	54	Inverse of timer 3 enable
NTEN4	55	Inverse of timer 4 enable
NTEN5	56	Inverse of timer 5 enable
TDN1	57	Timer 1 done
TDN2	58	Timer 2 done
TDN3	59	Timer 3 done
TDN4	60	Timer 4 done
TDN5	61	Timer 5 done
NTDN1	62	Inverse of timer 1 done
NTDN2	63	Inverse of timer 2 done
NTDN3	64	Inverse of timer 3 done
NTDN4	65	Inverse of timer 4 done
NTDN5	66	Inverse of timer 5 done
RTDN	67	Real-time output done
NRTDN	68	Inverse of real-time output done

PARTS LIST

Semiconductors:

IC1, IC2 - CD4050 hex buffer
IC3-IC10 - MCT2E optocoupler
T1-T8 - SL100 npn transistor
D1-D12 - 1N4007 rectifier diode

Resistors (all ¼-watt, ±5% carbon, unless stated otherwise):

R1-R8 - 1-kilo-ohm
R9-R24 - 47-ohm
R25-R32 - 1.5-kilo-ohm

Capacitors:

C1 - 1000μF, 63V electolytic

Miscellaneous:

X1 - 230V AC primary to 0-12V AC, 1A secondary transformer
F1 - 1A fuse
RL1-RL8 - 12V, 200-ohm relay
- 25-pin D M/F connector

NI1, NI2, NI3, NI4, and NI5, respectively.

Outputs. The system can turn on eight outputs (O1, O2, O3, O4, O5, O6, O7, and O8). These can be solenoids, motors, indicators, lamps, etc. The compliments of these outputs are NO1, NO2, NO3, NO4, NO5, NO6, NO7, and NO8, respectively.

Up-counters. The system has five up-counters termed as C1, C2, C3, C4, and C5. These can be set for a fixed terminal count, and when the terminal count is reached, an output can be turned on. The counters have two other associated variables called CACCx and CDNx, where x represents the counter number. CACCx holds the actual count at any given time and it advances each time the condition changes from true to false. CDNx is a variable that becomes 1 when the set count is reached.

Down-counters. The system has five down-counters termed as CO1, CO2, CO3, CO4, and CO5. These can be set for a fixed initial count, and when the zero is reached, an output can be turned on. The counters have two other variables called COACCx and CODNx, where x represents the counter number. COACCx holds the actual count at any given time and it decrements each time the condition changes from true to false. CODNx is a variable that becomes 1 when the zero count is reached.

Timers. The system has five timers, which can be used to time any action or event. When the interlocks to start the timer have been closed, a variable called TENx (timer enable) becomes 1 and the timer starts to time, updating the accumulator value denoted by variable TACCx (timer accumulator). When the set

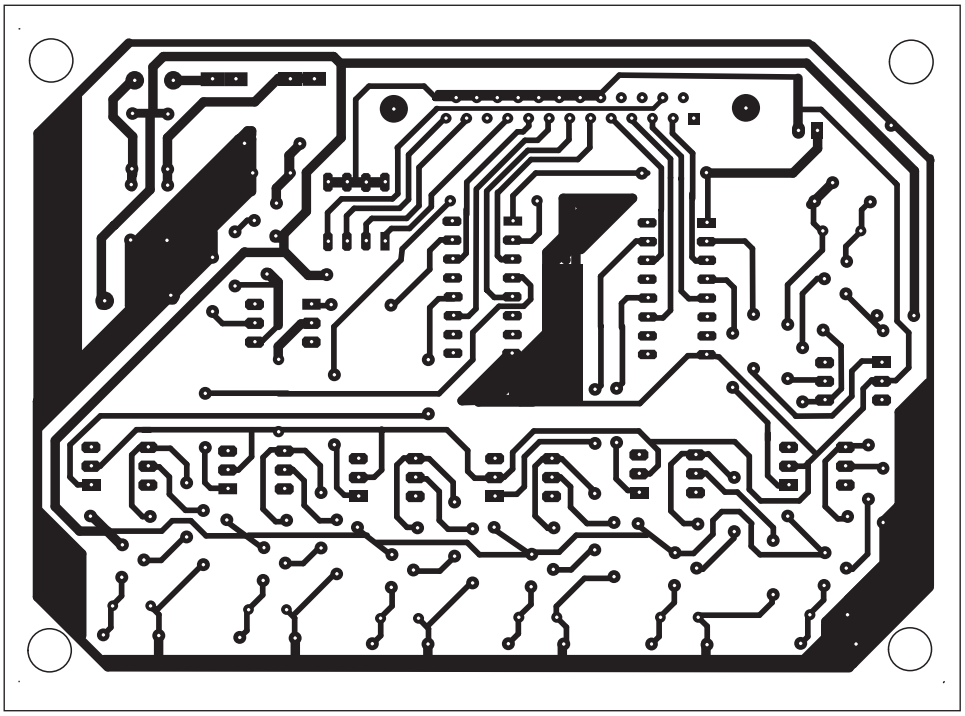


Fig. 3: Actual-size single-side PCB for the PLC interface circuit

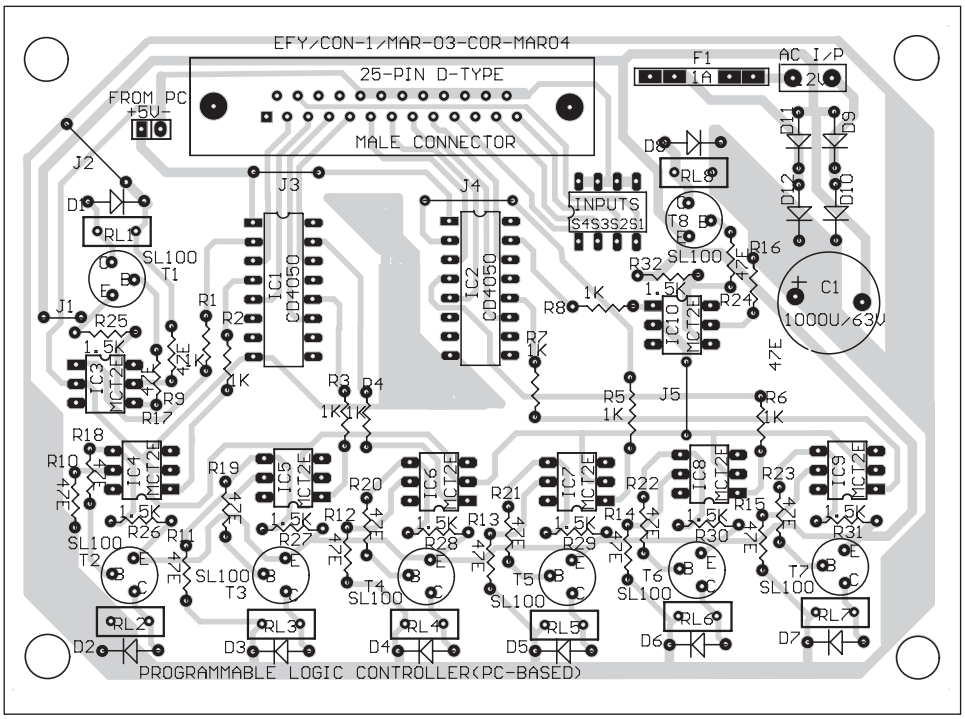


Fig. 4: Component layout for the PCB

time (set in variable Tx) is reached, a variable called TDNx is set to 1, which can be used to turn on any output.

Real-time timer. The real-time timer can be set to turn on any output at any preset time. This function has a variable called RTDN (real-time timer done) that becomes 1 when the set time is reached.

Program commands

The commands for writing the program, along with their functions, given in Table III.

Each of the above commands needs to be followed by associated parameters (generally termed as command line parameters), which the program appropriately interprets. The use of command-specific parameters, along with their syntax, is explained below:

ser (series contacts). This command accepts a single parameter, namely, the address number (refer Table II) of the interlock being used. If you require a number of series contacts, you can use corresponding number of ser commmands. For example, to use a series contact of input I1, use the command as:

```
ser
1
```

par (parrellel contacts). This command accepts a single parameter, namely, the address number of the interlock being used. If you require a number of parrallel contacts, you can use the corresponding number of par commmands. For example, to use a parrallel contact of input I1, use the command as:

```
par
1
```

ope (output energise). This command accepts a single parameter, namely, the address number of the output to energise. For example, to energise output O1, use the command as:

```
ope
1
```

where 1 is the address of output O1.

opl (output latch). This command accepts a single parameter, namely, the address number of the output to be latched. A latched output will remain energised until it is unlatched by the opu command. For example, to latch output O1, use the command as:

```
opl
1
```

where 1 is the address of output O1.

opu (output unlatch). This command accepts a single parameter, namely, the address number of the output to unlatch. For example, to unlatch output O1, use the command as:

```
opu
1
```

where 1 is the address of output.

tmr (timer). This command accepts

TABLE III: Commands and Their functions

S.No.	Command	Function
1.	ser	Series contacts
2.	par	Parrallel contacts
3.	ope	Energise the specified output
4.	opl	Latch the specified output
5.	opu	Unlatch the specified output
6.	tmr	Start the specified timer
7.	ctu	Start the specified up-counter
8.	ctd	Start the specified down-counter
9.	rto	Real-time output

two parameters, namely, the timer number and the set time in seconds. For example, to start timer T2 with a set time of 5 seconds, use the command as:

tmr
2
5

ctu (up-counter). This command accepts two parameters, namely, the down-counter number and the set count. For example, to start counter C2 with a set count of 5 counts, use the command as:

ctu
2
5

ctd (down-counter). This command accepts two parameters, namely, the down-counter number and the set count.

For example, to start counter C3 with a set count of 15 counts, use the command as:

ctd
3
15

rto (real-time output). This command is used to turn on any output at any fixed time (real time). It accepts two parameters, namely, hour and minute. For example, to enable the real-time command at 4 hours 15 mins, use the command as:

rto
4
15

Example

In the example shown in Fig. 5, a pump is used to pump water from a sump to an overhead tank. The sump has a low-level switch to ensure that the pump doesn't pump when there is no water.

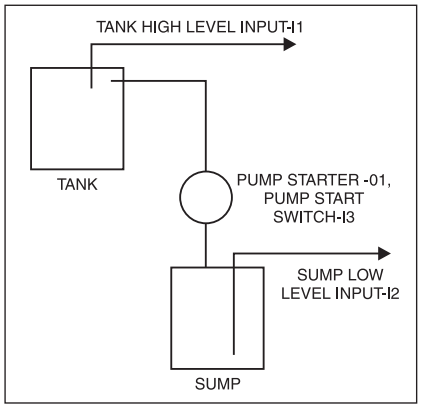


Fig. 5: The example

Similarly, the overhead tank has a high-level switch to ensure that the pump doesn't pump when the tank is full. To achieve the intended result, the logic should be designed such that:

1. The pump starts (output O1 should be available) only when the sump is not empty (not I2, i.e. NI2) AND the overhead tank is not full (not I1, i.e.NI1) AND the start switch (I3) is on.

2. The pump is off when the sump is empty or the overhead tank is full.

The program logic for the above is as follows:

I3 AND (NOT I2) AND (NOT I1)= O1

and its program is:

ser
3
ser
7
ser
6
ope
1

where ser is the command for ANDing (in series) contacts, while 3, 7, and 6 are the programming codes for I3, NI2 (Not I2), and NI1 (Not I1), respectively (refer Table II).

As is seen in the example, the logic for this control can be programmed into the software directly. The inputs and outputs are the only wired components, while the logic to be followed is in the software program.

Conclusion

The PLC system described here uses four inputs (out of five possible inputs) and eight outputs, which can be programmed in any way to control any equipment. The example discussed is a very small program and the system is capable of handling very complex programs (to be written by the user).

Sample program by EFY

For the benefit of readers, we've included here an additional sample program that has been written by us with guidance from the author. This program comprises three stages for turning on output 1, output 4, and output 8 on successful completion of operation sequences of stages 1, 2, and 3, respectively.

In the first part, the output 1 is to be turned on after a delay of 5 seconds after switch S1 (input 1) is closed. Here timer 1 is used for delay. In the second part, the output 4 is turned on only when output 1 is off. In the third part, the output 8 is turned on only after the up-counter 1 reaches terminal count of 3 and input 1 (switch S1) is off.

```
ser     ; Start of Stage 1
1       ; Scan input1 (switch S1) and it
          is ON
tmr
1       ; then set the timer 1 to count 5
          (sec.)
5
Ser
57      ; If terminal count for timer1
          reached (address for TDN1=57).
ope
1       ; Then turn 'ON' output 1
Ser     ; Start of Stage 2
19      ; Check for output1 to be off
          (NO1 address code=19). If yes,
ope
4       ; then turn 'ON' output 4
Ser     ; Start of Stage 3
2       ; Read input switch S2 to
          determine if it is ON. If yes
Ctu
1
3       ; then set the Up-counter 1 to
          count of 3.
Ser
27      ; If terminal count reached
          (address code for CDN2=27)
Ser
6       ; then check for input1 off
          (address code for NI1=6). If yes,
ope
8       ; then Turn 'ON' output 8
```

EFY note. All relevent files, including the source code and the executable version of the program, included in the CD. ❑

AUTOMATED CAR PARKING SYSTEM

ATUL APTE,
SHEETAL KUMAR AJMERA,
ROHAN D'SA, RAHUL GODBOLE

With the growing number of vehicles and the consequent shortage of parking space, there is haphazard and totally unregulated parking of vehicles all over. The situation calls out for an automated parking system that not only regulates parking in a given area but also keeps the manual control to a bare minimum.

To cater to the need, here we present a miniature model of an automated car parking system that regulates the number of cars that can be parked in an area at any given time based on the parking space availability. The entry and exit of vehicles are facilitated using a totally automated gate. Status signals indicate whether space is currently available in the parking lot, and whether a car is currently in the process of entering or leaving the parking space.

After the initial installation, the system requires no manual control. Everything, right from maintaining the

PARTS LIST

Semiconductors:

IC1	- 74LS155 dual 2:4 decoder
IC2	- 7404 hex inverter
IC3	- 7400 NAND gate
IC4	- 7432 OR gate
IC5-IC8	- 74LS74 dual 'D' flip-flop
IC9	- 4511 7- segment driver
IC10	- 74193 4-bit up-/down-counter
IC11	- L293D push-pull four-channel driver with diode
IC12-IC13	- NE555 timer
D1-D2	- 1N4148 diode
LED1	- 5mm yellow LED
LED2	- 5mm red LED
LED3	- 5mm green LED
IR1-IR2	- Infrared transmitter LED
IR3-IR4	- Infrared receiver module (TSOP 1738)
DIS1	- LTS-543 common-cathode 7-segment display

Resistors (all ¼-watt, ±5% carbon, unless stated otherwise):

R1-R2	- 3.3-kilo-ohm
R3-R4	- 1.8-kilo-ohm
R5, R6, R8	- 100-ohm
R7, R9	- 1-mega-ohm
R10-R19	- 330-ohm

Capacitors:

C1	- 0.22μF ceramic disk
C2-C4	- 0.01μF ceramic disk
C5, C7	- 4.7μF, 16V electrolytic
C6, C8	- 22μF, 16V electrolytic

Miscellaneous:

S1-S3	- Push-to-on tactile switch
	- Two 8-pin bases
	- Seven 14-pin bases
	- Four 16-pin bases
	- 5V, 1A regulated power supply
	- Flexible wire
	- Motor up to 600mA output current capability

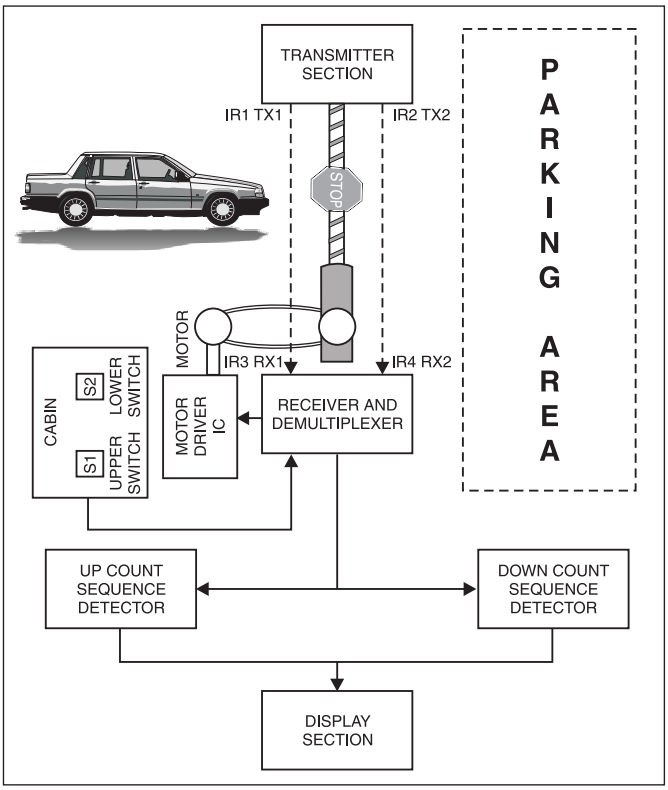


Fig. 1: Block diagram of automated car parking system

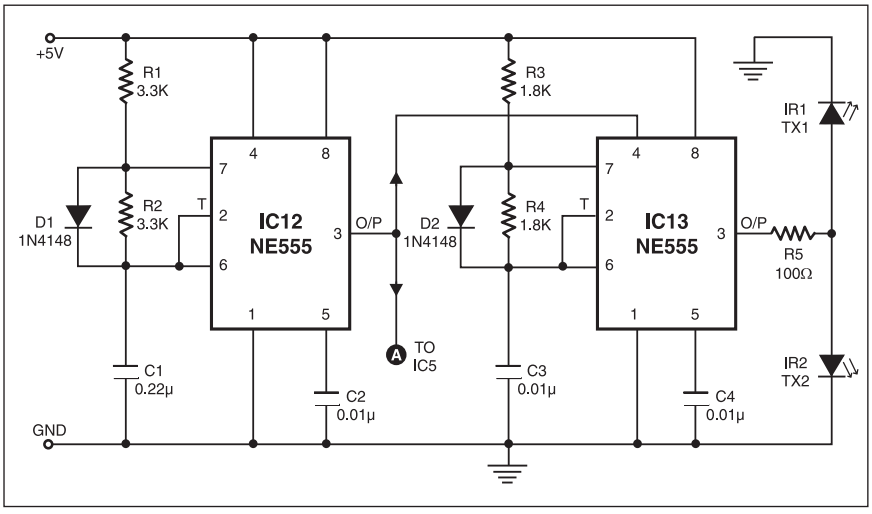


Fig. 2: Circuit diagram of IR transmitter part

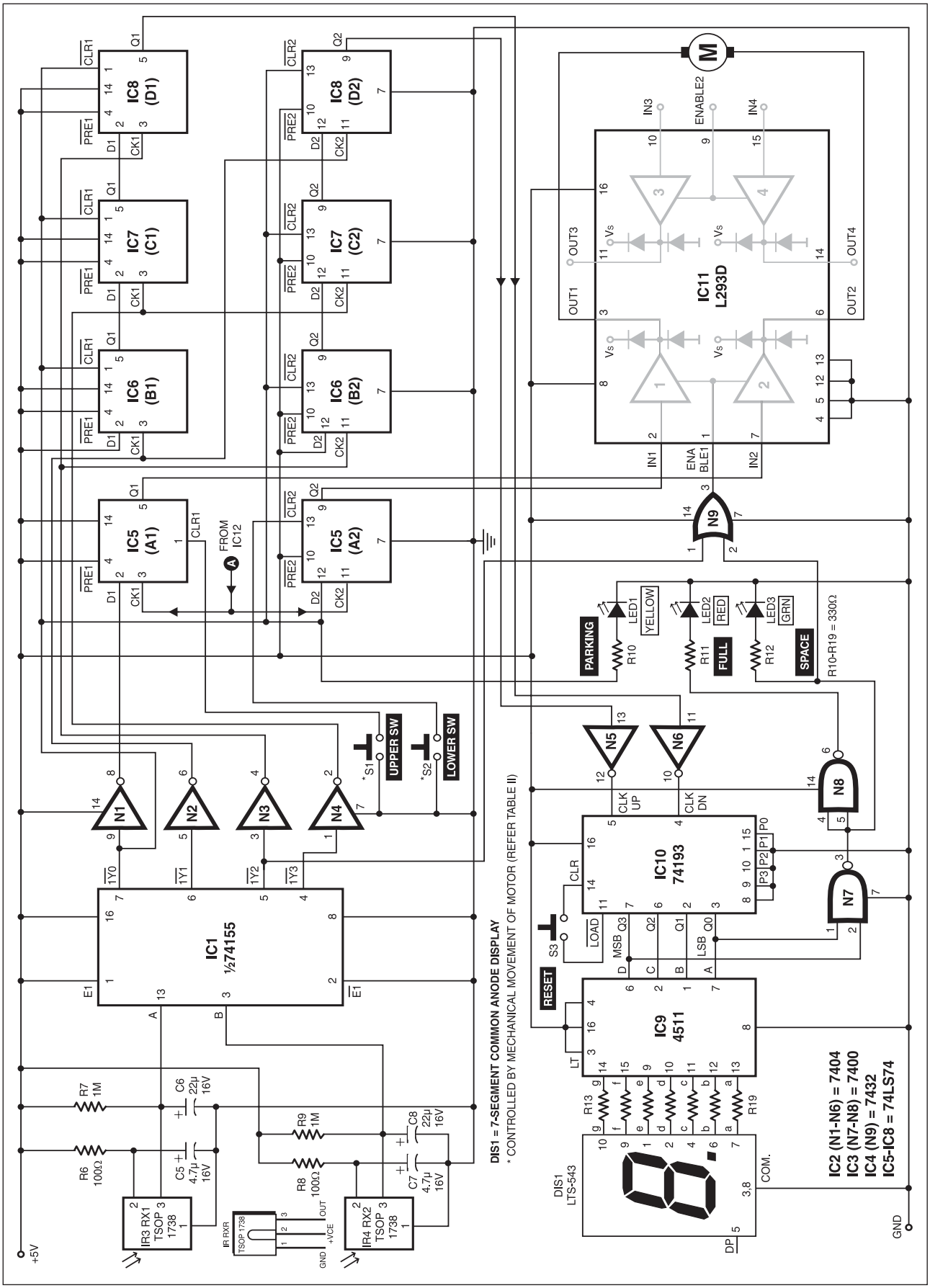


Fig. 3: Circuit diagram of automated car parking system (main)

count of vehicles to opening and closing of the gate, is automatically controlled. As the circuit uses low-cost and easily available discrete ICs, it is cost-effective.

System overview

A gate has been provided at the entry of the parking space, which opens on the arrival or departure of a car.

A display section has been provided, which consists of status signals and a display showing the number of cars present in the parking space at any point of time.

After the maximum number of cars have entered the parking space, the gate is automatically disabled (closed) for vehicles seeking entry into the parking lot.

A logic circuit distinguishes between the cars and persons/two-wheelers, so that persons and two-wheelers aren't not included in the count for cars.

Block diagram

Fig. 1 shows the block diagram of the automated car park system. The system consists of transmitter, receiver and demultiplexer, up-counter, down-counter, and display sections.

The transmitter section comprises two infrared transmitters (IR1 TX1 and IR2 TX2), which transmit infrared beams as shown in Fig. 2. These light beams are incident on the corresponding infrared receiver modules (IR3 RX1 and IR4 RX2), which produce an output of 0V if the beam is received uninterrupted and +5V if the beam is interrupted by a car.

Whenever a car enters the parking area, it interrupts the infrared beams in a definite sequence. This sequence is given to the up-count sequence detector, which generates a high output only if the correct sequence has been detected. Similarly, when the car leaves the parking area, it generates a fixed sequence, which is given to the down-count sequence detector. The down-count sequence detector generates a high output only if the correct sequence is produced by the exiting car.

The outputs of the up-count and down-count blocks are given to the display section. The display section has a counter and a 7-segment display along with its driver IC to display the count. Depending on the sequence detector that generates an actuating signal, the count is either incremented or decremented. The display section also consists of status signals, which include:

1. A yellow signal to indicate that a car is currently in the process of entering or leaving the parking space.
2. A green signal to indicate that the parking lot has not reached its maximum capacity, and that space is available for the parking of a car in the parking area.
3. A red signal to indicate that the parking space is full. The activation of this signal coincides with the disabling of the green signal, and is accompanied by the disabling (closing) of the gate for vehicles trying to enter the parking lot.

TABLE I
Truth Table of 74155 (IC1)

Address/Inputs		Enable		Outputs			
Pin 13 (A)	Pin 3 (B)	Pin 1 E1	Pin 2 E1	Pin 7 1Y0	Pin 6 1Y1	Pin 5 1Y2	Pin 4 1Y3
0	0	H	L	L	H	H	H
1	0	H	L	H	L	H	H
1	1	H	L	H	H	H	L
0	1	H	L	H	H	L	H

TABLE II
Truth Table of 7474 (IC5)

PIN 2 (D1)	PIN 13 (D2)	PIN 5 Q1	PIN 9 Q2	State
0	1	0	0	Default. Lower switch S2 closed.
1	0	1	0	First sensor cut. The gate starts opening and lower switch S2 is released.
1	0	1	0	The gate keeps opening.
1	0	0	0	Upper switch S1 closed. The gate stops opening.
0	1	0	1	Car completely enters the parking area. The gate starts closing and upper switch S1 is released.
0	1	0	1	The gate continues to close.
0	1	0	0	The gate pushes lower switch S2 and stops moving. Back to default states.

Note. *Q1 of IC5 is connected to pin 2 of IC11 and Q2 of IC5 is connected to pin 7 of IC11.*

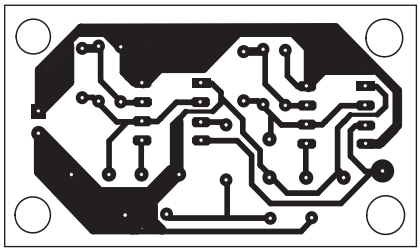


Fig. 4: Actual size, single-side PCB for transmitter

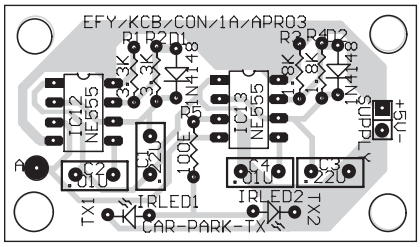


Fig. 5: Component layout for PCB in Fig. 4

The circuit

The automated car parking circuit (shown in Figs 2 and 3) primarily uses two NE555 timer ICs, four 74LS74 D flip-flops, 74155 2:4 decoder, up/down binary counter 74193, 7-segment display driver CD4511, miniature motor driver L293D, NAND gate IC 7400, and NOT gate IC 7404. In addition, the circuit uses two TSOP 1738 infrared receiver modules, two infrared transmitting LEDs, 7-segment display, and green, red and yellow LEDs, along with three push-to-on switches.

For easy understanding of the circuit, let's divide the circuit into the following four basic sections:

1. Sensor
2. Sequence detector
3. Counter and display
4. Gate control

The sensor section. This section senses the movement of objects and transfers that information to IC1 in the main circuit. The sensor section can be further divided into the transmitter section and the receiver section. The promi-

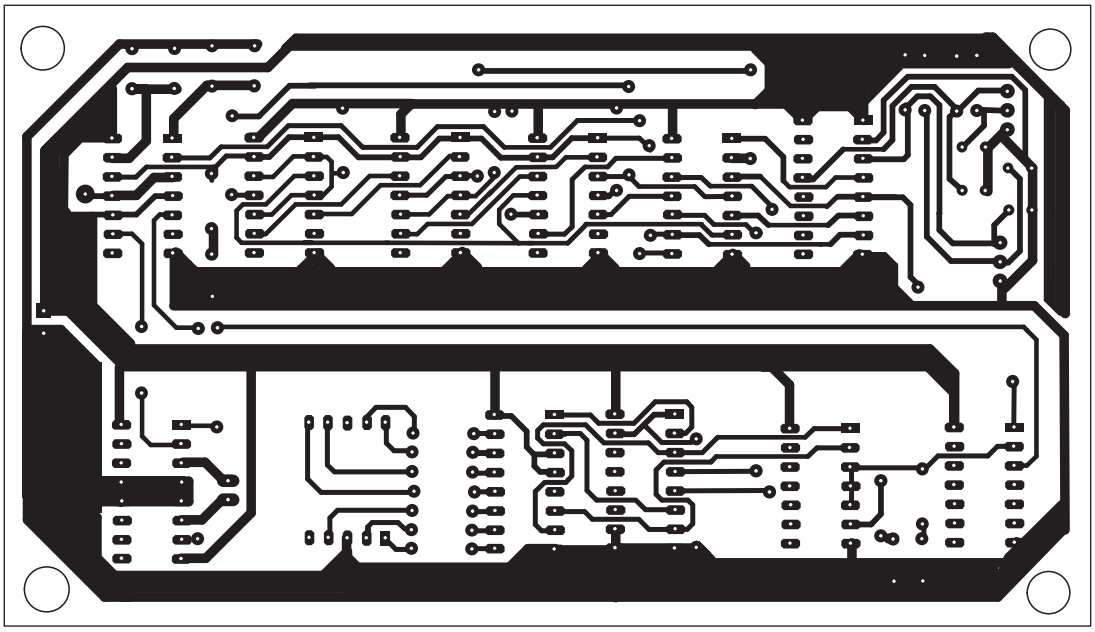


Fig. 6: Actual-size, single-side PCB for circuit in Fig. 3

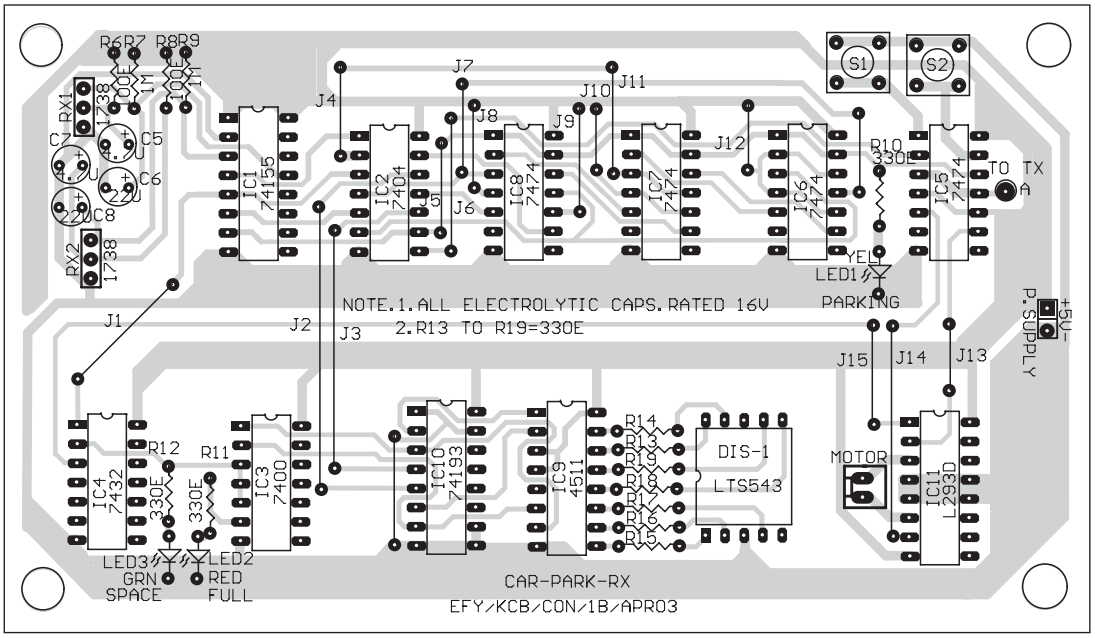


Fig. 7: Component layout for PCB in Fig. 6

nent component used in the design of the transmitter and receiver sections is the IR receiver module TSOP 1738. This is a highly selective receiver, which comprises a photodetector and a preamplifier with IR filter in a single package to provide demodulated output. It works efficiently with 1kHz modulation of 38kHz bursts. This feature of the receiver determines the composition of the transmitted signal.

For generating approximately 38kHz frequency carrier signal modulated by a 1kHz square wave, we use two NE555 timer ICs in astable mode in the transmitter section. One NE555 timer (IC12) is designed to produce a square wave of 1kHz with 50% duty cycle, while the second timer (IC13) is designed to produce a square wave of 38 kHz with 50% duty cycle. In order to modulate the 38kHz wave, output pin 3 of the first NE555 (IC12) is connected to reset pin 4 of the second NE555 (IC13). The final output of this cascaded arrangement is given to a pair of IR LEDs through current-limiting resistor R5, which prevents the IR LED from getting heated and thus damaged.

The receiver section consists of two identical receiver circuits, using one infrared receiver TSOP 1738 each. The output of this receiver is open-collector type, and hence requires a pull-up resistor, whose value must be much greater than 10k. A 4.7μF electrolytic capacitor must be connected between the supply and ground for this receiver to minimise the interference of spurious signals in the operation of the receiver.

When the signal is received correctly, the original 1kHz squarewave signal is obtained at the output of the receiver. In the absence of the signal, however, a +5V DC level is obtained. Since the ICs in the following blocks are of TTL family, the receiver output must be TTL compatible.

The +5V DC level occasionally drops to 0V, even when the signal strength is quite low, due to the very high sensitivity of the receiver. This may lead to false triggering of the circuit, which must be eliminated. For this, a 22μF electrolytic capacitor is connected between the output of the receiver and ground. This capacitor bypasses the square wave to ground and holds the DC value of the signal (which is 0V) in the normal state and +5V when the signal is blocked. In place of this capacitor, you may also use any capacitor of comparable value.

The output of the sensor section goes to the sequence detection section.

The sequence detection section. This section is the heart of the entire system. It consists of a 2:4 decoder and flip-flops, which are used for the sequence detection. The 74155 dual 2:4 decoder IC1 receives

its select signals at pins 13 (A) and 3 (B) (for one of the decoders) from receivers RX1 and RX2, respectively. The other decoder is not used. The output lines of the enabled decoder are active-low.

For convenience, the receiver before the entrance to the gate is connected to pin 13 of IC1. In default state, each receiver is active and inputs 0 to the decoder, making the Y0 output line low.

When the first sensor is blocked, the Y1 line goes low. The low-going Y2 line indicates that only the second sensor is blocked. A low Y3 line indicates that both signals have been blocked. Refer truth table of IC1 74155 given in Table I. The four output lines act as decoding and control signals for the remaining circuit.

The sequence detection logic circuit consists of three flip-flops for detecting incoming as well as outgoing vehicles. The Y0 line is connected to the clear pins of all the flip-flops, which gives 0 at their respective outputs. A vehicle entering the parking area must interrupt the first sensor (before entrance), then both sensors, and finally just the second sensor (after entrance). Thus it must generate states 1 0, 1 1, and 0 1, necessarily in that sequence.

For identifying the states and the order in which they occur, we give the Y2 , Y3, and Y1 lines after logical inversion to the clock inputs of three successive flip-flops, respectively. A Vcc signal is input to the first flip-flop, while each subsequent input is the output of the previous flip-flop. The logic states of the three decoded output lines are inverted because these are active low, while the 74LS74 D flip-flops are triggered by a rising edge of the clock signal.

Only the proper sequence of logic states will cause a high logic at the output of the third flip-flop. Any other sequence will not allow the transfer of the high signal through the series of flip-flops. The output of the third flip-flop is given to the counter-and-display section, which increments the count. Thus when a vehicle enters the parking area, the Y0 signal clears all the flip-flops, and at this very instant, the count is incremented.

An identical circuit is used for detecting a vehicle leaving the parking area. In this case, however, the states generated by the vehicle are 0 1, 1 1, and 1 0, necessarily in that order. Hence the clock signals for the three successive flip-flops are derived from Y1, Y3, and Y2 lines, respectively.

The working of this circuit is identical to the one for detecting a vehicle entering the parking area. In this case, the final D flip-flop output is given to the counter-and-display section for decrementing the count. This occurs at the instant when the outputs of the flip-flops are cleared by the low-going Y0 signal (explained in the counter-and-display section).

The counter-and-display section. This section consists of up-/down-counter IC 74193, BCD-to-7-segment decoder, display driver IC 4511 (to drive a common-cathode 7-segment display), and three LEDs (red, yellow, and green).

The counter IC 74193 is capable of handling up as well as down counts, if configured for the same. The count is incremented by one when a rising edge is encountered on the up pin (pin 5) and decremented by one when a rising edge is encountered on the down pin (pin 4). In our circuit, the former occurs when the vehicle has entered the parking area and line Y0 clears the output of the final flip-flop, causing a transition from the high to low logic state, which, when passed through an inverter, provides a rising edge. The count decrements in the same fashion when the flip-flops in question are those used for detecting the vehicle leaving the parking area.

The preset data pins of the counter IC are connected to Vcc, while the load data pin is connected to one end of a push-to-on switch whose other pin is grounded. Such an arrangement can be used to reset the counter, and consequently all the drivers and display unit in the circuit. The four BCD output lines of up-/down-counter (74193) are fed to the corresponding pins in the decoder/driver 4511 (IC9). The active-high outputs of the decoder are connected to their corresponding pins in the 7-segment common-cathode display.

The MSB and LSB lines of the outputs of counter IC10 are ANDed using gates N7 and N8. The output from gate N8 is fed to the anode of the red LED, which indicates that nine vehicles are present in the parking area, and there is no further space. This happens because the output of binary 9 on the lines makes the extreme lines high, which gives a high at the otherwise-low anode of the red LED, thus turning it on.

The same signal after inversion is given to the anode of the green LED, which indicates the availability of space for at least one vehicle in the parking area.

TABLE III

LED	Indication
Yellow	Car is in the process of parking
Red	No vacancy
Green	Parking space available

TABLE IV

Input	Enable*	Output
H	H	H
L	H	L
H	L	Z
L	L	Z

Note. *1. Z is high-impedance output*
*2. *For channel under consideration*

The yellow LED indicates that a vehicle is either entering or leaving the parking area. Hence, this LED must be on when at least one of the sensors is being cut. For this reason, the Y0 line of the decoder is given at the anode of the LED. When no signal is being cut, the Y0 line is low, keeping the LED off. But as soon as any of the signals is cut, the Y0 line goes high, turning the yellow LED on. The LED indication for various situations is depicted in Table III.

The gate control section. The gate control section consists of IC5, IC4, and IC11, which provide the appropriate logic used for controlling operation of the gate/barrier.

Assume that the lower position of the barrier is the default position. Now whenever the input to motor driver IC11 is 1 0, it causes the motor to rotate, thereby causing the barrier to move such that it opens the entrance.

Similarly, when the input to motor driver is 0 1, the motor rotates in the opposite direction to lower the barrier, thereby closing the gate. When the input to the motor driver is 0 0, the motor does not rotate.

When the car has entered the parking area completely, the input to the L293D (IC11) is 0 1, causing the motor to rotate such that the gate begins to close till it pushes the lower switch, at which point it stops moving.

Thus, the movement of the gate is automatically controlled on the arrival or departure of a car. Table II gives a clear picture of the working of the gate control section.

In order to disable the gate from opening for a vehicle entering the parking area after the count of 9, we use a simple combinational logic circuit consisting of NAND

and OR gates, whose output is given to enable pin 1 of the L293D motor driver (IC11). In normal condition, the output of this logic circuit is high, enabling IC11. When the maximum count of 9 is reached, the output of the logic circuit becomes low, thereby disabling the motor, and keeping the gate closed for all vehicles seeking entry to the parking area.

However, when a vehicle wishes to leave the area, IC11 gets enabled, thus opening the gate. The output current capability per channel of L293D is approximately 600 mA. The truth table of L293D is given in Table IV.

An actual-size, single-side PCB for the transmitter circuit (Fig. 2) is shown in Fig. 4 with its component layout in Fig. 5. The actual-size, single-side PCB for the main circuit (Fig. 3) is shown in Fig. 6 with its component layout in Fig. 7.

This circuit is useful for underground parking, company parking, etc. It can even be modified to work on pay-and-park scheme. With a few adjustments, the number of cars that the parking space can accommodate can be altered.

EFY note. 1. There should be a battery back-up for knowing exact number of cars parked.

2. This project caters for 9 cars only. It applies for cars only (not for cycles/scooters).

3. Proper orientation of receiver and transmitter is very important.

4. The distance between the two transmitted beams should be less than the length of the longest car to be parked.

❑

Readers' comments:

Q1. I have the following queries regarding its circuit:

1. What type of motor has been used in the project?
2. What type of gate we should use for a miniature model?
3. How can I demonstrate the working of the circuit in my final-year project?

Renu Chand
Through e-mail

Q2. I am not getting the output. The display shows only '0' and does not change on giving the right sequence. I want to present this circuit as my mini project for my engineering course. Could you please send me a list of suggestions in order to overcome these problems?

S. Shakeel
Through e-mail

The authors Atul Apte, Sheetal Kr Ajamera, Rohan D'sa, and Rahul Godbole reply:

A1. 1. The motor used is a 5V DC motor.

2. We had used a rack and pinion kind of arrangement for the gate. A small grooved bar moves in the horizontal direction when the motor shaft rotates.
3. We had constructed a small-scale model of the car parking system, and had demonstrated the actuation of the gate, along with the change in the display, on the entry or exit of a car.

A2. Since the count is remaining at 0, it means that both the receivers are receiving signals continuously. That's why even though the signals might be getting cut in the correct order, the count is not incrementing. This could be due to a couple of reasons:

1. The strength of the transmitted signal is too large. Reduce the signal strength by increasing the transmitter resistance values.
2. The signal from one transmitter may be getting picked up by the other receiver. Since the TSOP1738 can receive signals at an angle of +45° to –45° of its axis, i.e. a total receiving angle of 90°, it can easily pick up signals from the other transmitter. To prevent this, enclose both transmitters in long black tubes, which could be easily made from black chart paper. This will ensure that the signal sent from one transmitter travels in a straight line and doesn't diverge. So the other receiver will not pick up the signal.

Observe the block diagram of the automated car parking system in article's Fig. 1. It shows both the trans-mitters on one side and both the receivers on the other side. This might increase the probability of cross-pick up, i.e. TX1 signal may incorrectly get picked up by RX2. To avoid this, place TX1 and RX2 on one side, and TX2 and RX1 on the other.

This, along with the use of long black tubes, will most likely solve your problem. Also, ensure that the transmitters and corresponding receivers are perfectly lined up in front of each other.

SIMPLE 32-BIT RELAY CARD FOR PC'S PARALLEL PORT

VIJAYA KUMAR P.

The PC's parallel port is a very versatile port for developing PC-based input and output applications. Controlling relays to switch on/off the gadgets is one of these applications, where the computer decides the sequence of switching. But the parallel port has only 12 outputs including 8 data lines and 4 control lines. The card described here can be used to expand the number of data output lines using octal D flip-flops to overcome this limitation. This card provides up to a maximum of four TTL-compatible latched output ports having eight output lines each, i.e. the card can provide up to 32 latched output lines. Each latched bit can be used to control a switching device such as relay.

Fig. 1 shows the block diagram of the relay card system. It comprises two sections, namely, the computer system and the external interface circuit. The computer system consists of input (keyboard), computer's internal circuitry for processing the input data, and the computer data output port. The programmed output data is available at the 25-pin parallel port, called printer port. The external interface circuit consists of data buffer circuits, latches, relay drivers, and relays for switching on/off the appliances. Before going into the actual hardware, let us have a brief description about the parallel port.

The parallel port

The parallel port or line printer terminal (LPT) port is a 25-pin 'D' type female connector available at the back of your PC. A basic IBM PC usually comes with one or two LPT ports. The original parallel port, called standard parallel port (SPP), is a bundle of three ports (or registers), namely, data port, status port, and control port. Pins numbered 2 through 9 form the 8-bit data port. This port is a purely write-only port. This means it can be used only to output some data through it. Pins 1, 14, 16, and 17 form the control port. The control port has both read and write capabilities. Pins 15, 13, 12, 10, and 11 together form the status port. The status port is a read-only port.

The base address of the first parallel port (LPT1) is 0378 in hexadecimal (hex) notation (or 888 in decimal notation). Hexadecimal addressing is most often used. The data port of the first parallel port can be accessed by its base address, i.e. 0378 in hex (or 888 in decimal). The status port

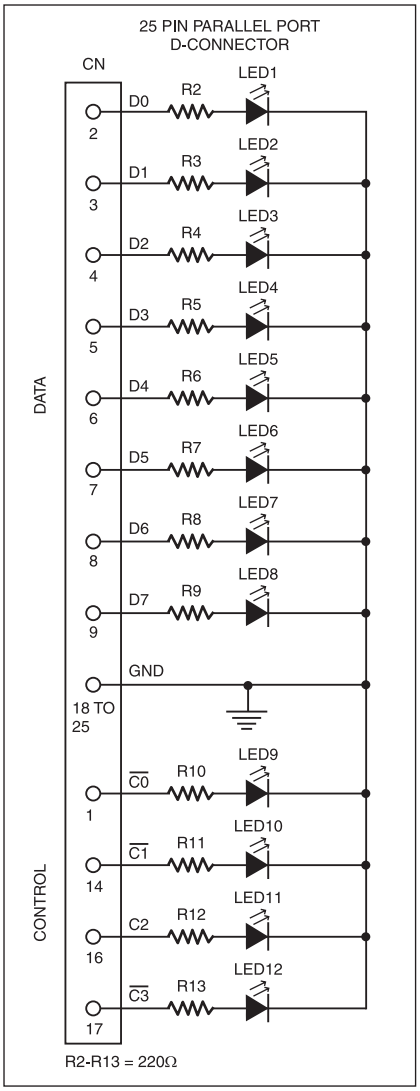


Fig. 2: Arrangement for testing the status of data and control bits

PARTS LIST

Semiconductors:

IC1	- 74LS244 octal tristate buffer
IC2-IC5	- 74LS273 octal D flip-flop
IC6	- ULN2803 octal Darlington array driver

Resistors (all ¼-watt, ±5% carbon, unless stated otherwise):

R1	- 2.2k, 1/4W, ±5% tolerance carbon resistor
R2-R9	- 220-ohm, 1/4W, ±5% tolerance carbon resistor

Capacitors:

C1-C5	- 100nF ceramic disk capacitor
C6	- 10μF/16V tantalum capacitor

Miscellaneous:

S	- Push-to-on switch (PCB mountable)
K1-K4	- 8-way female connector
K5	- 8-way male connector
CN	- 25-pin D-type male connector
RL0-RL7	- 12V, 500-ohm reed relay

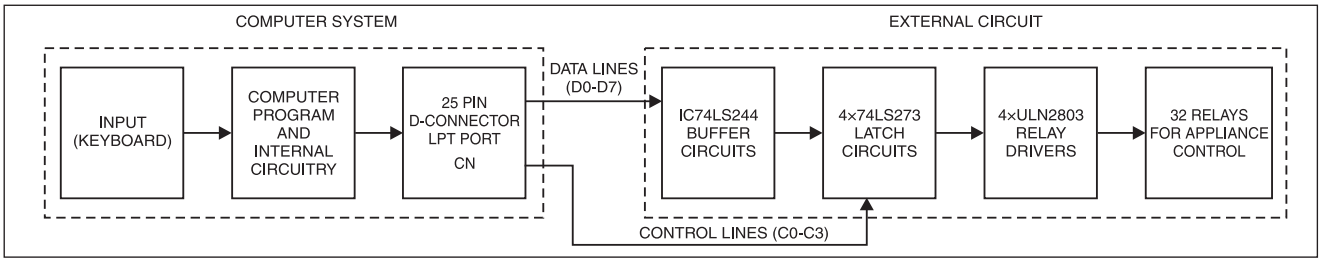


Fig. 1: Block diagram of the relay card system

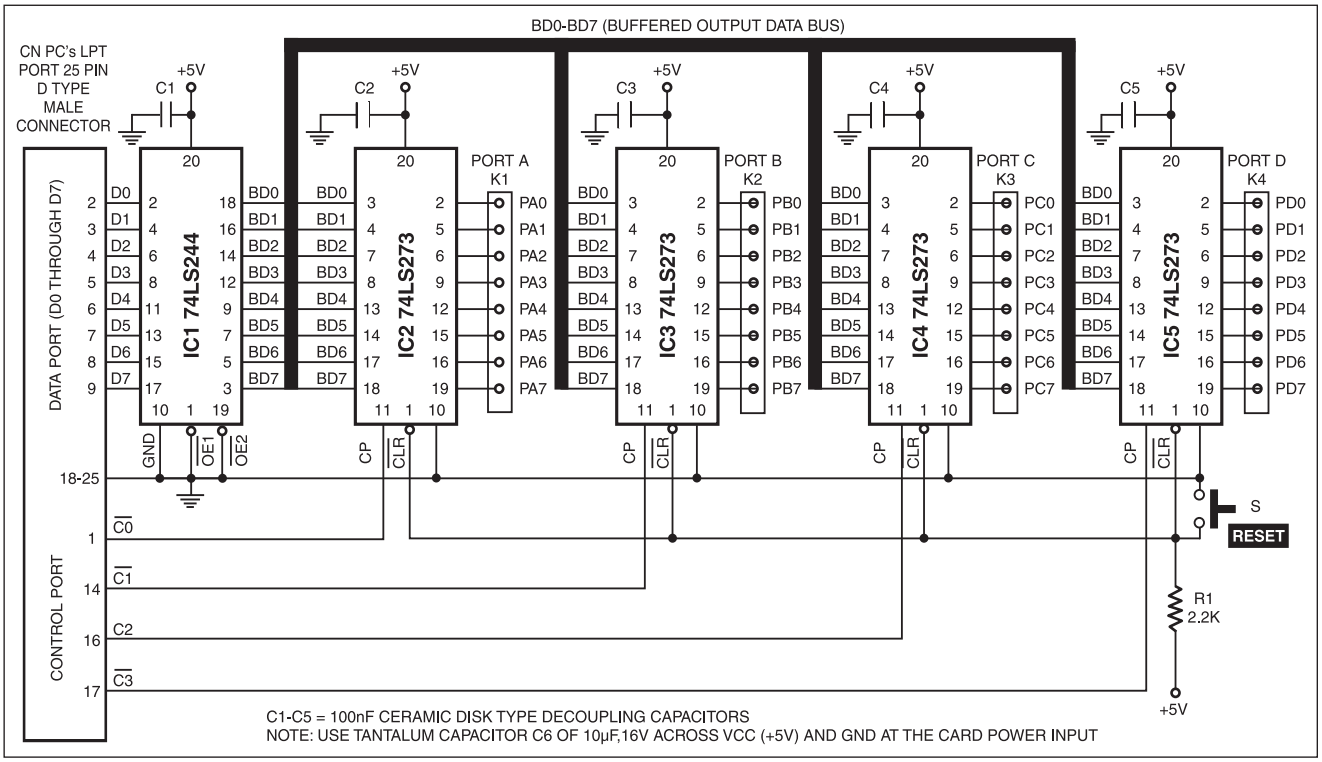
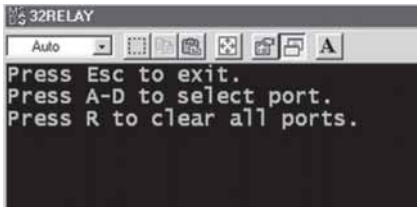


Fig. 3: Circuit diagram of 32-bit relay card for parallel port



Menu-like screen which appears on running the program

can be accessed using base address +1 = 0379 in hex (or 889 in decimal). The control port can be accessed using base address + 2 = 037A in hex (or 890 in decimal). A similar procedure can be followed for LPT2 whose base address is 0278 in hex.

Since this card requires only output lines and status port is a read-only port, the status port finds no use in the present application. Table I gives pin details of the standard parallel port for traditional use and for use with relay card.

Programming the card

The card can be used for any output application by suitably programming it. In the software program explained below, C0, C1, and C3 bits of the control port are inverted inside the PC (see Table I). So in order to make these pins high, one has to write 0's to the pins while programming. To help you understand the functioning of the card, a simple program in 'C' is given at the end of this article.

The program can be written using any text editor (notepad, wordpad, etc). It is compiled using Turbo C compiler. On running the program, a menu-like screen appears (refer the screenshot), which prompts you to press the name of the port. Suppose you press A, then the message "PORTA is selected" is displayed. The screen now prompts you to enter any decimal number from 0 to 255. If you enter 255, data outputs D0 through D7 (pin 2 through pin 9) will go high, i.e. the data output status will become 11111111. Reset R is used to

TABLE I
Parallel Port Pin Details

Pin number	Traditional use	Port name	Read/Write	Port Address	Port Bit	Hardware inverted?	Relay card use
2-9	Data Out	Data Port	W	Base	D0-D7	No	Data bits
1	Strobe	Control Port	R/W	Base+2	C0	Yes	Clock input for IC2
14	Auto Feed		R/W	Base+2	C1	Yes	Clock input for IC3
16	Initialise		R/W	Base+2	C2	No	Clock input for IC4
17	Select Input		R/W	Base+2	C3	Yes	Clock input for IC5
15	Error	Status Port	R	Base+1	S3	No	Not used
13	Select		R	Base+1	S4	No	Not used
12	Paper End		R	Base+1	S5	No	Not used
10	ACK		R	Base+1	S6	No	Not used
11	Busy		R	Base+1	S7	Yes	Not used
18-25	Ground	—	—	—	—	—	Ground

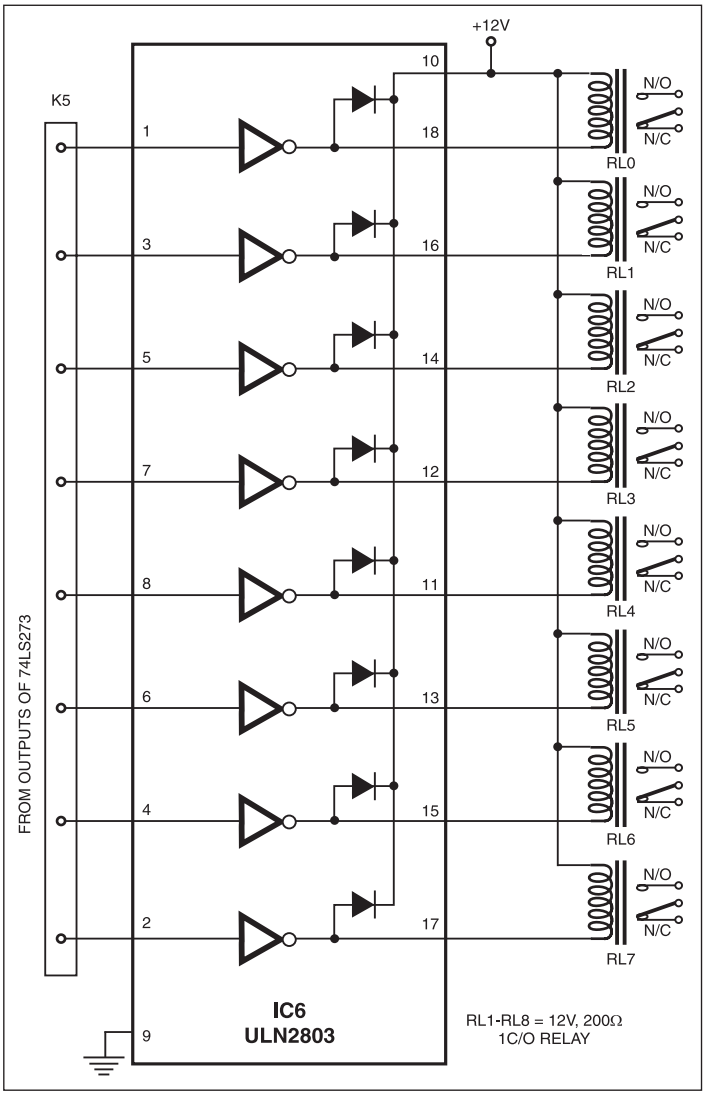


Fig. 4: Relay driver circuit

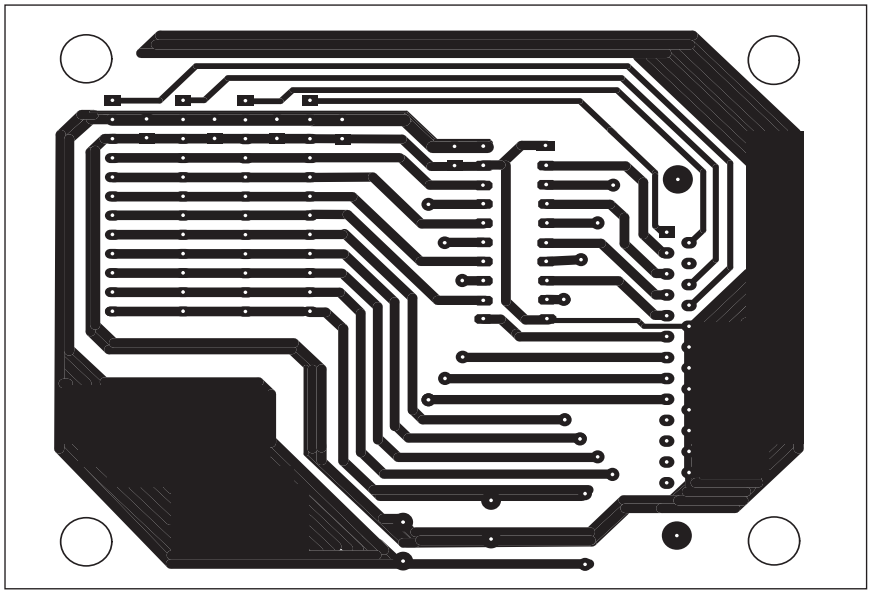


Fig. 5: Actual-size, single-side PCB-1 layout for relay card circuit

reset all the data outputs to 0.

To write data to any other port, follow the same procedure by choosing the port name from the output of the program. Pressing R will clear all the data bits of the data port and make all the control lines to go low simultaneously. This will clear all the data at the output ports. Press Esc to exit the program.

The program first prompts you to enter the port name (A to D) and then the data (decimal number 0 through 255). This data is converted into eight bits by the computer system and becomes available at the output port when Enter key is pressed. With the circuit given here, there are four 8-bit output ports, i.e. a total of 32 output lines are available.

The status of the data at the output port and the control lines can be tested using LEDs (LED1-LED8 and LED9-LED12, respectively) as shown in Fig. 2. When port A is selected, only LED9 corresponding to the control line C0 must go low and high again. Since this action is very fast and cannot be sensed by our eyes, you can put a delay between the statements used to make control line low and high. For example , if you are testing control line $\overline{C0}$ corresponding to portA you can insert the delay as given below:

```
    if(ch=='A')
 {
 input();
 outportb(PORT+2,0x05);
 delay(100);   outportb(PORT+2,0x04);
 }
```

Thus, if A followed by 5 is entered from the menu, the corresponding LED1, LED3, and LED9 will go low and high once. Similarly, if B for port B is selected, only LED10 (corresponding to $\overline{C1}$) will go low and high once.

Having verified all the data ports and the control lines, remove all the LEDs and resistors. Connect the control lines $\overline{C0}$, $\overline{C1}$, C2, and $\overline{C3}$ to pin 11 of each of the four ICs (IC2 through IC5) as shown in Fig. 3. Connect the computer data lines (D0 through D7) to the inputs of IC1 (74LS244). The data outputs (BD0 through BD7) of IC1 are connected to the corresponding input pins of the four 74LS273 ICs (IC2 through IC5). The outputs of 74LS273 ICs are TTL-compatible and can be used to interface with any digital output application. The eight outputs of each 74LS273 IC form one latched data output port. The four latched data output ports are named as port A, port B, port C, and port D.

The status of data outputs of IC2 can also be observed using LEDs. Connect eight resistors of 220 ohms to output pins of IC2 in series with eight LEDs (as shown in Fig. 2). To observe all the four output ports of the ICs, a total of 32 resistors and 32 LEDs will have to be used. Note that in this 32-bit card, it is also possible to change the data bits of two or more ports simultaneously by taking the corresponding control bits from low to high simultaneously.

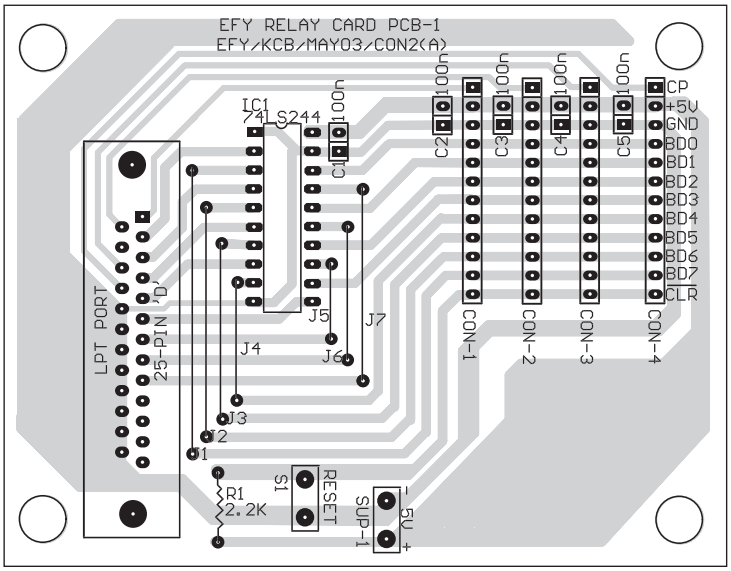


Fig. 6: Component layout for the PCB in Fig. 5

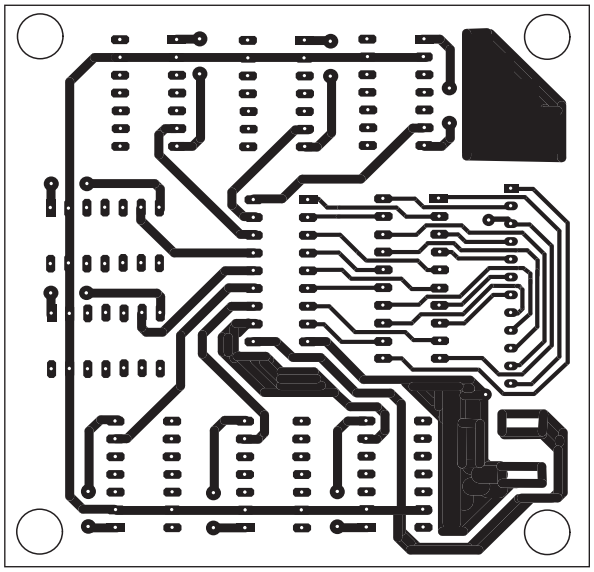


Fig. 7: Actual-size, single-side PCB-2 layout for relay driver circuit

Description

The hardware interface is shown in Fig. 3. All the ICs used in the circuit belong to standard TTL family. IC1 (74LS244) is an octal tristate buffer, which is used to buffer data output lines D0 through D7 of the data port. The buffered data lines BD0 through BD7 are shown as a single bus in the circuit diagram for simplicity, however it has eight distinct data lines. IC2 through IC5 (74LS273) are octal D flip-flops with active-low clear input. The data inputs of each IC 74LS273 get the data through the common buffered data bus (BD0-BD7).

A rising clock edge applied at the clock input pin 11 (CP) of IC 74LS273 causes the IC 74LS273 to latch the data available on the buffered data bus and the data will be available at its output pins. Clock inputs (CP) of the four 74LS273 ICs are connected to the control port's bits C0, C1, C2, and C3, respectively, as shown in the circuit diagram. And hence making a control bit low and then high will act as a rising-edge clock pulse to the corresponding octal D flip-flop 74LS273 to latch the data available on the data bus (BD0-BD7). The data once latched will not alter unless one writes new data into the data bus (BD0-BD7) and then changes clock input (CP) of the corresponding 74LS273 from low to high again.

All the active-low clear inputs (pin 1) of 74LS273 ICs are tied together and connected to a pull-up resistor (R1) and a push-to-on switch S to ground. Pressing switch S pulls the active-low clear inputs of all 74LS273 ICs to ground and clears all the data at ports A through D. All the ports can also be cleared by writing 0's to the data port in the software program and then making all the control bits to go low and then high.

Procedure to latch the new data

Write a program to accept input data from the keyboard. Output this data to data port using 'outportb' instruction. Make the control bit, say, C0 of IC2 (74LS273), low by writing a 4-bit binary number (nibble) to the control lines such that all remaining control bits are at high state. Now make all the control bits high by writing another 4-bit binary number (nibble) to the control lines.

Compile and run the program. Select port name, say, A, and enter any decimal number between 0 and 255. This data is available on the buffered data bus (BD0-BD7) of IC1. The data corresponding to IC2 gets latched, i.e. the data at the output of IC2 will not change even when new data is written to the output data port. As IC 74LS273 is a positive-edge triggered octal D flip-flop, keeping its clock input CP high after latching has no effect and will not alter its data content. Only rising edge of the clock pulse can make the 74LS273 to latch the data.

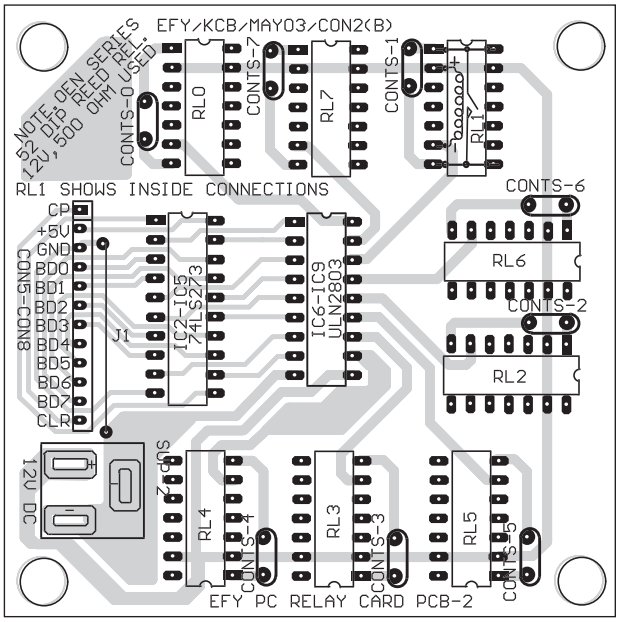


Fig. 8: Component layout for PCB in Fig. 7

Driving the relays

Each bit of the latched output ports can be used to control a separate relay. Since the digital outputs of the 74LS273 cannot sink much current, they are not capable of driving relays directly. Eight relays can be driven using octal darlington array driver ULN2803 (IC6) as shown in Fig. 4. So a total of 32 relays can be driven by connecting four such arrangements to four different ports of the 32-bit latched output card.

PROGRAM IN C

```
/*****************************************/
/* Test Program for 32 bit Relay card for LPT port
*/
/* Language: C                           */
/* File name: 32RELAY.C                  */
/* By Vijaya Kumar.P                     */
/*****************************************/
  #include <stdio.h>
  #include <dos.h>
  #define PORT 0x0378  /*Base address of LPT1*/
                       /*Use 0x0278 for LPT2
*/
  void input(void);
  void reset(void);

  void main()
   {
     char ch;
     reset();
     clrscr();
     do
     {
            printf("Press Esc to exit.\n");
            printf("Press A-D to select port.\n");
            printf("Press R to clear all ports.\n");
            ch=getch();
            ch= toupper(ch);
            if(ch>=65&&ch<=68)
             {
               printf("PORT %c is selected.\n",ch);
             }
            if(ch=='A')
                {
                  input();
                   outportb(PORT+2,0x05);/*making C3
C2 C1 C0 = 1110*/
                   outportb(PORT+2,0x04);/*making C3
C2 C1 C0 = 1111*/
                 }
                if(ch=='B')
                 {
                   input();
                   outportb(PORT+2,0x06);/*making C3
C2 C1 C0 = 1101*/
                   outportb(PORT+2,0x04);/*making C3
C2 C1 C0 = 1111*/
                 }
                if(ch=='C')
                 {
                   input();
                   outportb(PORT+2,0x00);/*making C3
C2 C1 C0 = 1011*/
                   outportb(PORT+2,0x04);/*making C3
C2 C1 C0 = 1111*/
                 }
                if(ch=='D')
                 {
                   input();
                   outportb(PORT+2,0x0C);/*making C3
C2 C1 C0 = 0111*/
                   outportb(PORT+2,0x04);/*making C3
C2 C1 C0 = 1111*/
                 }
                if(ch=='R')
                  {
                    reset();
                    printf("All ports are cleared.\n\n");
                  }
          }while(ch!=27);/* exit the program if esc is
pressed */
                clrscr();
    }

      /* function to take decimal number from the key-
board */
      void input(void)
       { int num;
                  printf("Enter a decimal number
between 0 to 255.\n");
                  scanf("%d" ,&num);
                  outportb(PORT,num);/* writes binary
equivalents of decimal

 number entered into the data pins*/
       }

       /* function to clear all the ports */
       void  reset(void)
       {
                  outportb(PORT,0x00);   /* making all
data bits 0s */
                  outportb(PORT+2,0x0B);/* making con-
trol bits C3 C2 C1 C0 = 0000 */
                  outportb(PORT+2,0x04);/* making con-
trol bits C3 C2 C1 C0 = 1111 */
       }
```

Note that Ground reference is same for both Figs 3 and 4.

The input of IC ULN2803 is TTL-compatible. ULN2803 has open-collector outputs. As each of these outputs can sink a maximum collector current of 500 mA, miniature PCB relays can be easily driven using ULN2803. No additional free-wheeling clamp diode is required to be connected across the relay since each of the outputs of IC ULN2803 has inbuilt free-wheeling diodes.

Construction and installation

The card can be constructed by mounting components on a suitably designed PCB. It can be used externally by connecting the card to the PC's parallel port using a normal printer cable having 25-pin D-type male connector. Each ULN2803 relay driver can be mounted on a separate PCB along with eight PCB-mountable relays. The relay driving circuit can be connected to one of the ports (port A through port B) using connectors. To drive 32 relays, connect four such relay driver cards (each comprising 8 relay drivers/relays) to four different latched output ports. Note that the relay drivers require a separate 12V supply in order to drive the relays but the ground reference is same as that of the PC.

In order to simplify the PCB design, two single-side PCBs have been designed. The first PCB (shown in Fig. 5 with its component layout in Fig. 6) accommodates a 25-pin connector (for LPT port), IC1 (74LS244), and four 12-pin SIP connectors to carry buffered data (8), clock control line (1), ground (1), 5 volt (1), and common reset (1). The second PCB (shown in Fig. 7 with component layout in Fig. 8) accommodates eight OEN relays (series 52-12V, 500-ohm), one 74LS273, one ULN2803, and 12V DC I/P connector. The 12-pin connector on this card goes to any of the four connectors of the first PCB. Four such cards need to be used for 32 relays.

Note. The source code, executable file and all the relevant files are included in CD. ❑

Readers' comments:

Q1. I have the following doubts:

1. Do the signals sent by the other software for printing affect the relay card output? How can I achieve the transparency for the printer signals?
2. What is the maximum output current provided by relay output contacts?

Also, suggest me a suitable bridge circuit for the circuit's power supply.

Kamlesh Kulkarni
Dombivli

Q2. If any port, say, port A, is selected, which relays energise when we press 1, 2, 3, 4,...., 255 in that sequence against the computer program instruction "Press decimal number 0-255"?

Deepak Kumar
Through e-mail

The author, Vijay Kumar P. replies:

A1. 1. Yes, the output status of the 32-bit relay card gets affected by the printing software. If your PC has more than one printer port, you can use the second/third printer port for the 32-bit relay card. While using second or third LPT port (LPT2/LPT3), you need to change the base address of the port to 278H (for LPT2) or 3BCH (for LPT3) in the source program. You cannot use the printer and the 32-bit relay card simultaneously with a single LPT port. But you can design a circuit to switch between the relay card and the printer manually using tristate buffers.

2. The maximum output current which can be drawn by the load through the relay depends upon the current rating of the relay used. You can use relays of a higher current rating depending upon your requirement.

The power supply circuit for the 32-bit relay card is shown in Figure below. A transformer of 2A current rating is chosen taking into account the total current requirement of 32 relays of 200 ohms each.

A2. Thank you for showing interest in my project. The program actually writes

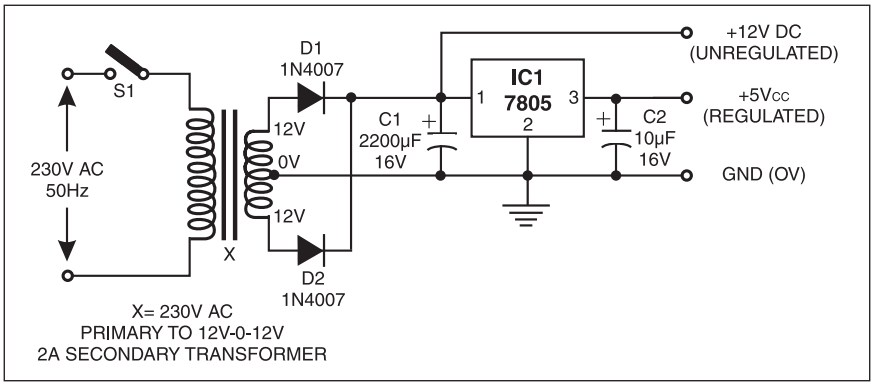


Fig.: Power supply circuit

the binary equivalent of the decimal number entered into the specified port. For example, if you select port A and enter decimal '1,' its binary equivalent '00000001' will be written to port A, so the relay connected to PA0 pin of port A energises while all other relays remain off.

Similarly, for decimal '2,' its binary equivalent '00000010' is outputted to port A and the relay connected to PA1 pin energises while all other relays remain off. For decimal '3,' the binary equivalent is '00000011,' so both the relays connected to PA0 and PA1 pins energise. For decimal '4,' the binary equivalent is '00000100,' so the relay connected to PA2 pin energises. For decimal '5,' binary equivalent is '00000101,' so relays connected to PA0 and PA2 pins energise, and so on.

Note that the program given in the article is a test program to test and understand the working of the 32-bit relay card. You may write your own program depending upon your application by understanding the working of the test program.

TEMPERATURE MEASUREMENT USING TRANSISTOR AS SENSOR

ARVIND S.

Many circuits have been published in EFY to measure temperature using commercially available sensors like AD590 and LM337. Here's a simple temperature sensor built around a transistor (used as a diode) and a single-chip analogue-to-digital converter (ADC) cum 3½-digit LED driver ICL7107. Apart from displaying the temperature with a resolution of 0.1°C, this circuit provides for temperature-based relay activation for controlling heaters, coolers, etc. The trip-point temperature, set for activating the relay, can also be displayed with simple flick of a switch. This circuit can be calibrated to have an accuracy of ±2°C, with an operating range of –25°C to 125°C.

Circuit operation

Fig. 1 shows the circuit diagram of the temperature measurement system using transistor T2 as a sensor. The base and collector of T2 have been shorted so that it is reduced to a diode. The forward voltage drop of the diode changes with temperature. The rate of change in voltage drop is around –2 mV/°C. Thus, the forward voltage of the diode drops with increase in temperature. This principle is used in this circuit to measure temperature with reasonable accuracy.

A small-signal transistor provides the best characteristics suitable for use as a temperature sensor. As a matter of fact, any small-signal transistor can be used in place of the BC547 in the circuit with similar results. The base-emitter diode is used as the sensing element in this circuit. This diode has a nominal forward voltage drop of around 0.6V at room temperature. The diode is forward biased with a current of 100 μA, derived from the internal reference of ADC ICL7107. The voltage drops by around 2 mV for every °C rise in temperature.

This voltage is connected to the IN LO input of the ICL7107CPL. (ICL7107CPL is an LSI ADC chip with necessary circuitry built into it, so a simple 3½-digit ADC can be implemented with just a handful of components. ICL710 7CPL is the LED-driver version of the popular ICL7106, which is commonly used in low-cost LCD digital multimeters.) The IN HI signal is connected to a trimpot, which is set such that at 0°C the potential difference between IN HI and IN LO is 0V.

With increase in temperature, the voltage drop in the base-emitter diode of transistor reduces and hence the potential difference between IN HI and IN LO increases. This potential difference is converted into a digital value by the ADC, as per the following relationship:

$$\frac{(\text{IN HI}-\text{IN LO})}{(\text{REF HI}-\text{REF LO})} \times 1000 \text{ counts}$$

where REF HI and REF LO are the voltage levels at pins 36 and 35 of IC1, respectively.

A reference voltage is critical in all analogue-to-digital conversions. The ICL7107CPL has an internal band gap reference that provides a stable output of around 2.8V, as measured from the positive voltage rail (+5V). The analogue common (AC) is used for all the analogue circuitry. Thus, at any point of time, the potential between the AC and the +5V rail is maintained at a constant 2.8V by the internal reference. Trimpot VR1 connected between the AC and the +5V sets the reference voltage for the analogue-to-digital conversion (REF HI–REF LO).

ADC parameters

The ADC requires a fairly stable clock, which is generated by using a simple R-C combination. With the components shown across pins 39 and 38 of IC1 in the circuit, a clock frequency of about 48 kHz is obtained. For all ranges

PARTS LIST

Semiconductors:

IC1	- ICL7107CPL 3½-digit LED single-chip ADC
IC2	- LM311 voltage comparator
IC3	- 7805 +5V regulator
T1	- SK100 pnp transistor
T2	- BC547 npn transistor
ZD	- 5.1V zener diode
D1-D7	- 1N4007 rectifier diode
DIS1-DIS3	- LTS542 common-anode display
DIS4	- 1/2 LTS542 common-anode display

Resistors (all ¼-watt, ±5% carbon, unless stated otherwise):

R1, R2, R12	- 100-kilo-ohm
R3	- 22-kilo-ohm
R4	- 47-kilo-ohm
R5, R6	- 3.3-kilo-ohm
R7-R9	- 3.9-kilo-ohm
R10, R14	- 220-ohm
R11	- 1.2-kilo-ohm
R13	- 28-kilo-ohm
VR1-VR2	- 10-kilo-ohm trimpot (low-drift type)
VR3	- 10-kilo-ohm potmeter

Capacitors:

C1	- 0.1μF ceramic disk
C2	- 100pF ceramic disk
C3	- 0.47μF polyester
C4	- 0.22μF ceramic disk
C5-C6	- 220μF, 25V electrolytic

Miscellaneous:

X	- 230V AC primary to 12V-0- 12V, 300mA secondary transformer
RL	- 12V, 200-ohm 1c/o relay
S	- Push-to-changeover switch
	- Shielded cable
	- Flexible cable
	- Heater or any other load

of the frequency, a 100k resistor is recommended and the capacitor value is calculated from the equation f = 0.45/RC. This gives a conversion rate of 3 readings per second.

Since this is an integrating type ADC, an integrating capacitor (C4, 0.22 μF) is used here. As the accuracy of the ADC greatly depends on the type of the capacitor, the capacitor must be carefully selected. Use a capacitor with low dielectric absorption, such as polypro-pelene, polystyrene, or polyester capacitor. Any other type of capacitor will provide erroneous results. The integrating current is set by the 47k resistor (R4).

The auto-zero capacitor has some influence on the noise of the system. For 200mV full scale, where low noise is very important, a 0.47μF polyester capacitor (C3) is recommended. A 0.1μF polyester capacitor (C1) is used for the internal reference circuitry. Since the ADC output section switches large currents, a 0.1μF ceramic disk decoupling capacitor is connected close to IC1.

The ADC has 23 outputs that directly interface with LTS542 common-anode 7-segment displays (DIS1 to DIS3) and 1/2-digit common-anode display (DIS4). You can use LTS542 by using its segments b, c, and g in place of 1/2-digit display with a, b, and POL (MINUS) segments, respectively. The result of the conversion, along with polarity, is displayed on these four 7-segment displays.

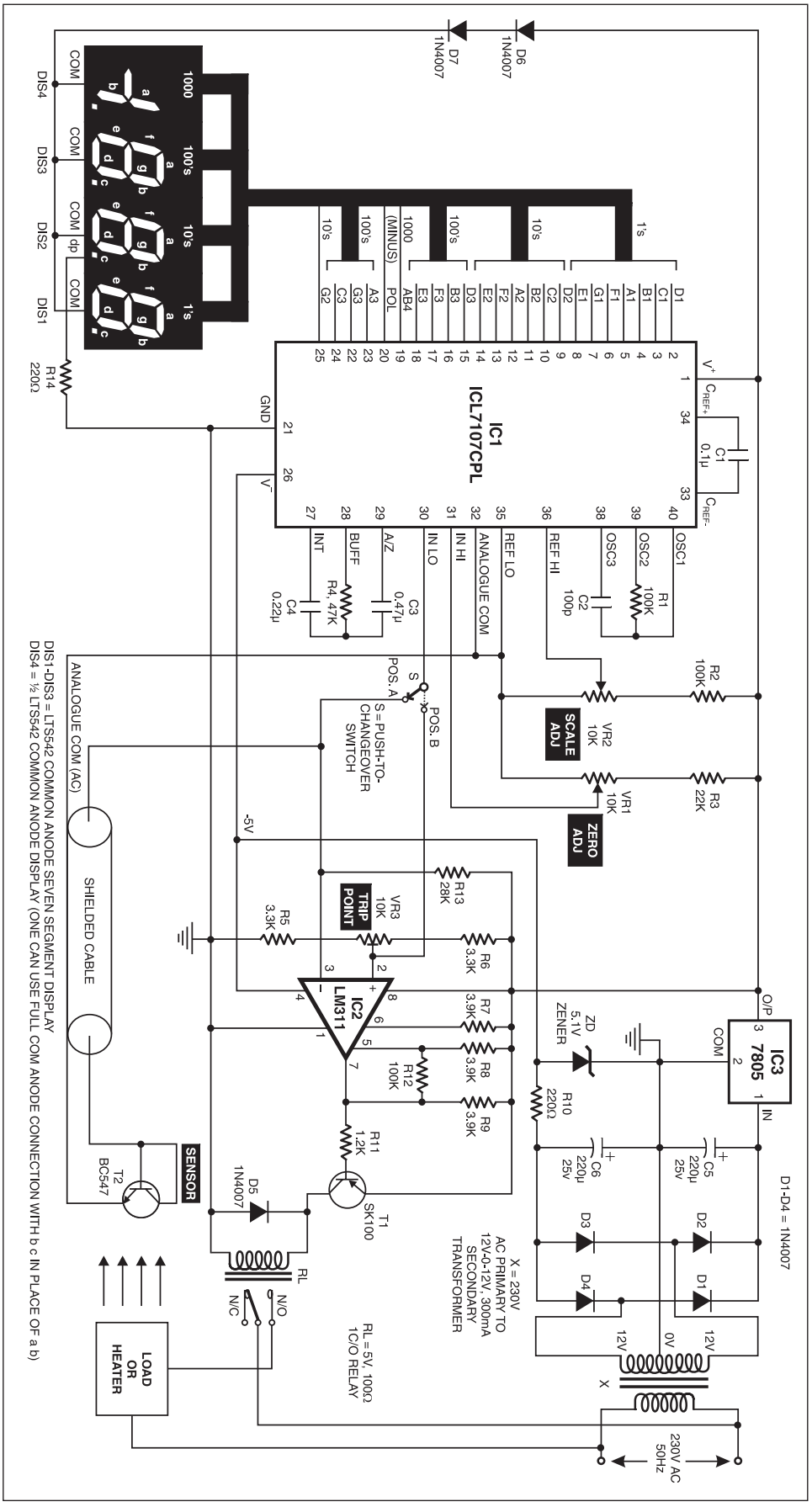


Fig. 1: Circuit diagram of the temperature measurement system using transistor as a sensor

Construction

An actual-size, single-side PCB for the temperature measurement system in Fig. 1 is shown in Fig. 2 with its component layout in Fig. 3.

The circuit operates off a simple power supply using an IC 7805 regulator. A ± 12V rectified and filtered voltage is created by using a bridge recti-

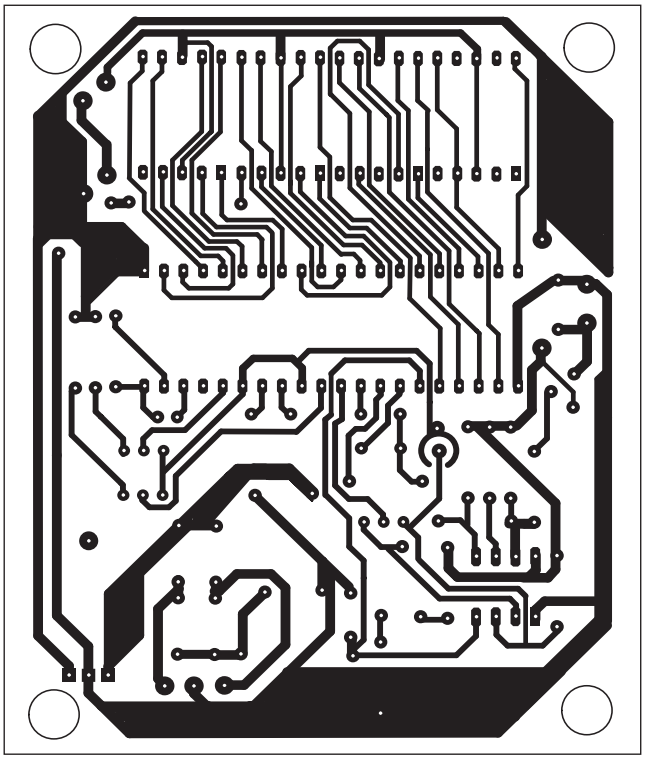


Fig. 2: Actual-size, single-side PCB for temperature measurement system using transistor as sensor

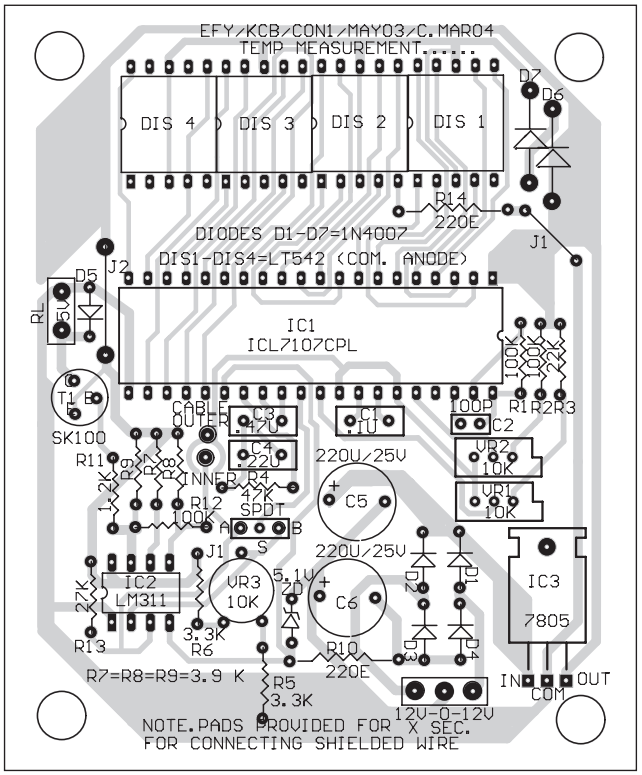


Fig. 3: Component layout for the PCB

fier (comprising diodes D1 through D4) and filter capacitors C5 and C6 at the output of transformer X (230V AC primary to 12V-0-12V, 300mA secondary). On the +12V rail, a 7805 regulator (IC3) is used to regulate the output to a stable 5V. On the –12V side, a simple zener diode (ZD) is used to generate a fairly stable –5V supply for the ADC.

Calibration

After assembling the circuit, dip sensor transistor T2 in ice-cold water at 0°C, taking care not to short the base and emitter leads of the transistor, and adjust trimpot VR2 until the display reads 000.0. After this, dip sensor transistor T2 in boiling water (100°C) and adjust trimpot VR1 such that the display reads 100.0.

Since the sensor diode takes some time to settle to the final temperature, the sen sor should be kept in ice or boiling water for at least 10 to 20 seconds before ad justing VR1 or VR2. Once calibration is done, the circuit is ready to measure the temperature of the desired medium.

Relay controller

This circuit can be used to control a heater (shown as load) using the relay contacts. The simple voltage comparator circuit built around LM311 (IC2) compares the temperature measured by sensor transistor T2 and trip-point voltage set by VR3. If the actual temperature measured by the sensor is less than the trip point, the relay remains energised. On reaching the trip point, the comparator switches off the relay, which, in turn, switches off the heater.

To monitor the trip-point temperature on display, press push-to-changeover switch S. This shifts switch S connection from position A to position B and connects the ADC input pin 30 of IC1 (IN LO) to the trip-point voltage set by VR3. The trip-point temperature is now displayed on the four 7-segment displays (DIS1 to DIS4). On releasing switch S, the ADC input pin 30 (IN LO) returns to position A, and now the ADC normally displays the sensor temperature. A positive feedback is used in the comparator to produce clean transitions at the output.

Caution

1. To ensure correct and accurate operation, select appropriate capacitors for the ADC.

2. Trimpots VR1 and VR2 must be of low-drift type. VR3 can be wire-wound type, which will greatly improve the stability of the circuit.

3. Take care while handling ICL7107CPL, as it is susceptible to electrostatic charge. Always use a shielded cable from the sensor to the PCB.

4. Analogue common (AC) and digital ground (GND) should not be connected together.

5. The load current should be within the relay contact ratings.

6. The sensor must never be kept floating in air if the air temperature is to be measured. Otherwise, the sensor will not give a stable output, leading to excessive rolling of the ADC outputs. To achieve a stable output, attach the sensor to a metallic device, which, in turn, may be exposed to the medium whose temperature is to be measured. ❑

DIGITAL CLOCK WITH SECONDS AND ALARM TIME DISPLAY

S.K. ROUSHON

Typical digital clocks using clock chips MM5387/MM5402 and MM5369 show normal time in only hours and minutes, and seconds or alarm time is visible only after pressing a push-to-on switch. Here's a digital clock using the same IC (MM5387) that shows normal time in hours, minutes, and seconds and alarm time simultaneously. For this, ten 7-segment LED displays and a few extra ICs and some discrete components are needed.

Pin details of IC MM5387 are shown in Fig. 1. IC MM5387 is a 40-pin dual-in-line package IC operated on 8V DC to 26V DC. It gets the positive DC supply voltage at its pin 28. Pin 29 is grounded. As per the datasheet of this IC, pins 28 and 29 are designated as V_{DD} and V_{SS}, respectively.

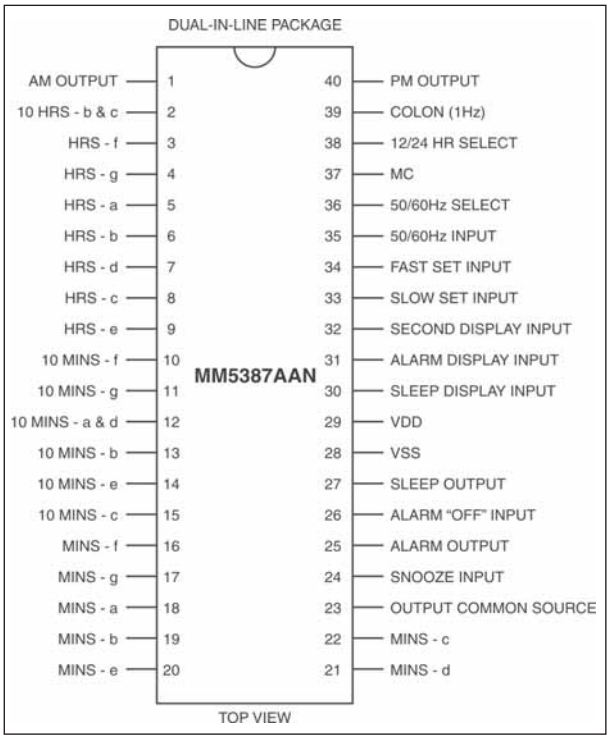


Fig. 1: Pin configuration of IC MM5387

Circuit description

When we switch a light on and off at a high frequency, the light appears 'on' all the time. This idea has been applied in this clock circuit to alternatively switch the seconds and alarm time displays on and off at a high speed. We use the CMOS decoder CD4017BE (IC1) and the multivibrator circuit using IC 555 (IC2) to do this job in the circuit. This combined circuit switches the two push-to-on switches for seconds and alarm time displays at a frequency that is same as the frequency of the multivibrator, which is quite high for our eyes to notice the changes in the display.

The outputs of IC1 at pins 32 and 31 of the clock IC are used for switching the seconds/alarm time display, as also for switching transistors T2 through T4 (BC547) at the same time, which, in turn, ground the common cathodes of the respective 7-segment LED displays (DIS1-DIS10). The three pairs in upper six 7-segment LEDs (DIS1 through DIS6) are used to display hours, minutes, and seconds, respectively.

Though this circuit has the option to select 24-hour format, we've designed it to display hours in 12-hour format only. By leaving pin 38 (12-/24-hours select pin) unconnected, the tens digit of the hours display can be programmed to provide a 12-hour display format.

Note that in the upper 7-segment LED displays, the tens digit of the hours display (DIS1) uses only two segments (b and c) of the LED to display '1' when the hours display reaches 10 and above.

The lower four 7-segment LED displays (DIS7 through DIS10) are used for the display of alarm time, where b and c segments are used to display '1' in the tens digit of the hours display (DIS7) as mentioned above. Also note that f and e segments of DIS7 are used to represent AM and PM, respectively, for the alarm time display.

PARTS LIST

Semiconductors:

IC1, IC5	- CD4017BE decade counter/ divider
IC2	- NE555 timer
IC3	- MM5387 clock chip
IC4	- CD4060 14-stage binary counter
T1-T5	- BC547 npn transistor
D1-D6	- 1N4001 rectifier diode
D7, D8	- 1N4148 fast switching diode

Resistors (all ¼-watt, ±5% carbon, unless stated otherwise):

R1	- 10-kilo-ohm
R2	- 150-ohm
R3-R7	- 15-kilo-ohm
R8	- 3.3-mega-ohm

Capacitors:

C1	- 0 .01μF ceramic disk
C2	- 4.7μF, 16V electrolytic
C3	- 1000μF, 16V electrolytic
C4	- 470μF, 16V electrolytic
C5-C7	- 0.22μF ceramic disk
C8, C9	- 47pF ceramic disk

Miscellaneous:

X_{TAL}	- 4.9152MHz crystal
LED1, LED2	- 5mm dia. red LED
DIS1-DIS10	- LT543 common-cathode 7-segment display
S1	- SPDT switch
S2-S7	- Push-to-on tactile switch
X	- 230V AC primary to 4.5V-0-4.5V secondary, 500mA transformer
	- 9V PP3 battery

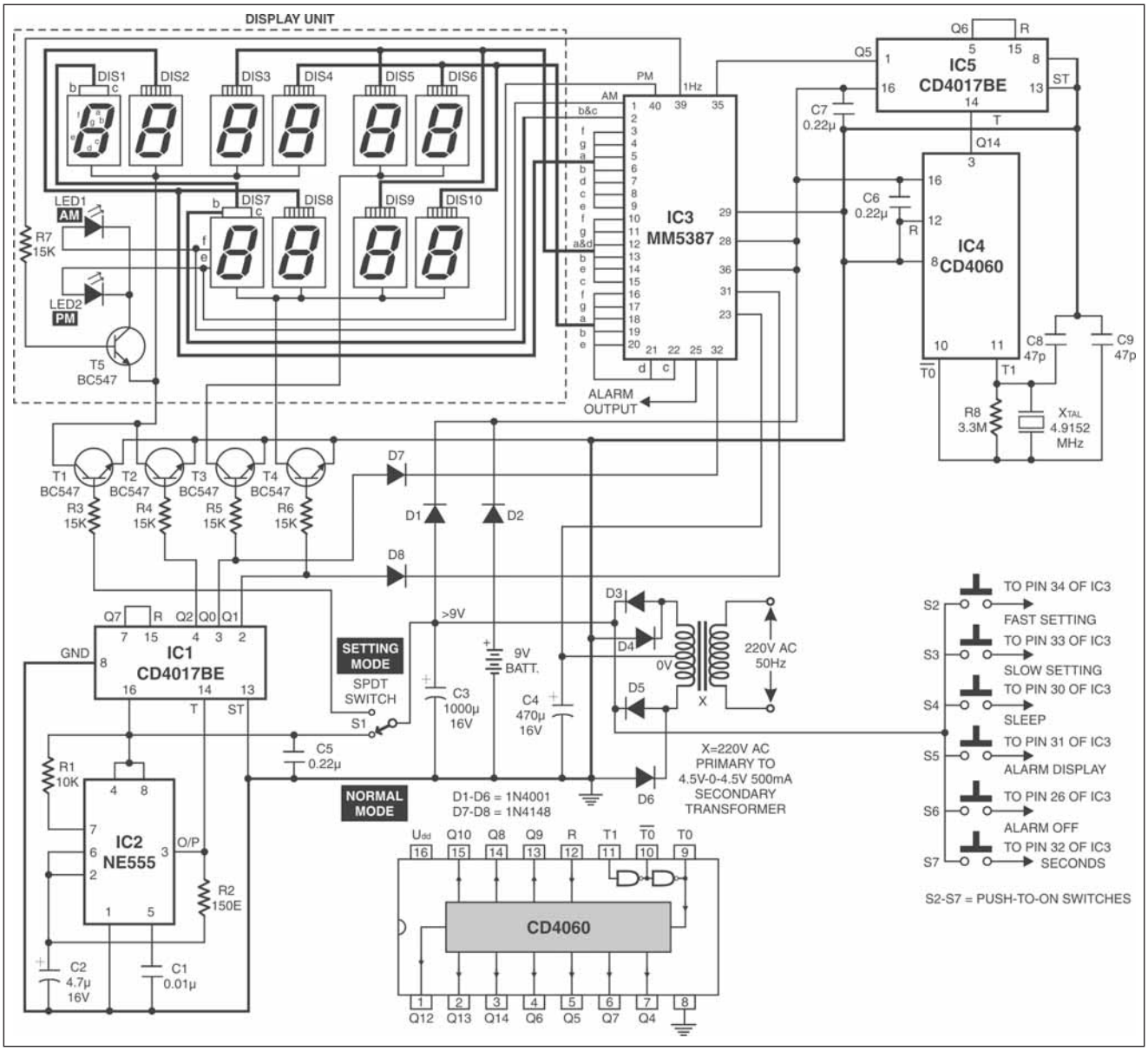


Fig. 2: The circuit of digital clock with seconds/alarm display

The time-keeping function of the clock chip (IC3) operates off a 50Hz or 60Hz input. Operation at 50 Hz is programmed by connecting pin 36 (50/60Hz select) of the clock IC to V_{SS}. To get a 50Hz clock pulse, we use the decade counter/divider CD4017BE (IC5) and the 14-stage counter CD4060 (IC4) together with a 4.9152MHz crystal (X_{TAL}). The IC CD4060 divides the crystal frequency of 4.9152 MHz by 16,384 and produces a 300Hz clock pulse, which is further divided by 6 using the CD4017BE (IC5) to get the required 50Hz pulse. This simple and low-cost time base circuit replaces the expensive IC MM5369.

The 1Hz output from pin 39 of IC3 is used to blink the colon LEDs (LED1 and LED2) for AM or PM indication. Pins 1 and 40 of IC3 are connected to f and e pins, respectively, of the tens digit of the hour of the alarm display (DIS7) and also connected to anodes of LED1 and LED2. Thus, AM/PM for alarm time and normal time (DIS1-DIS6) can be known from the blinking of f/e segment of DIS7 and the blinking of LED1/LED2, respectively.

In the schematic diagram shown in Fig. 2, the thick lines from IC3 to the 7-segment LEDs consist of several lines and the a, b, c, d, e, f, and g outputs of IC3 go to the corresponding segments of the 7-segment LEDs.

Power supply

In the power supply section, a 230V primary to 4.5V-0-4.5V, 500mA secondary step-down transformer (X) is used. The transformer must be of good quality as it will be always 'on'. We've used both the negative and positive supplies of the secondary of the transformer to reduce the price and size of the transformer. Thus, as shown in Fig. 2, IC3 gets a positive, unregulated DC voltage of around 10V at its pin 28. The centre-tapped secondary output of the transformer is 4.5V with respect to ground, which is connected to pin 23 of IC3. This makes the circuit compact.

A 9V backup battery is needed for the

clock to work during power failure. During power failure, the display is invisible but the clock continues to work. When the power resumes, the display shows the correct time.

The power supply circuit is accommodated in the PCB of the clock itself.

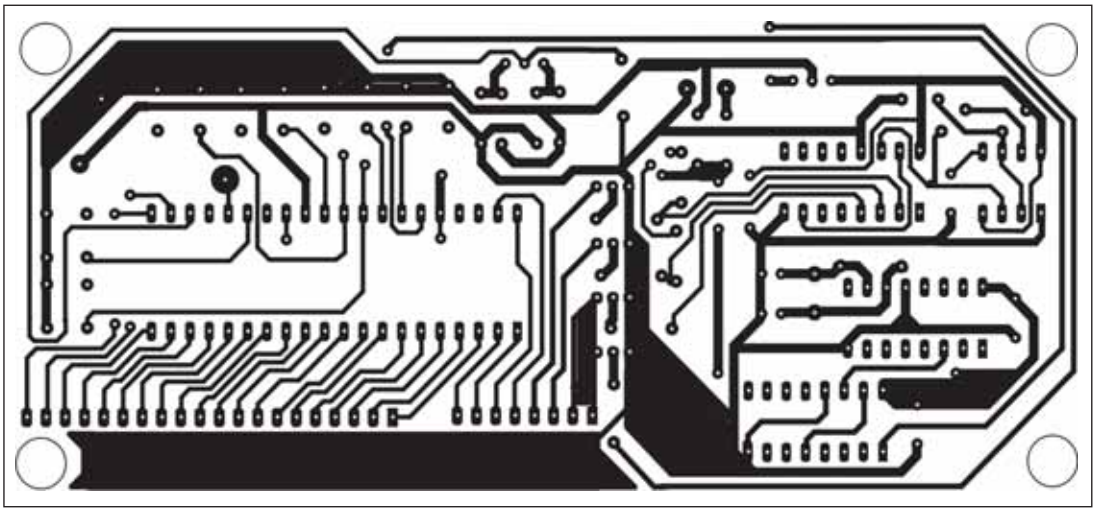


Fig. 3: Actual-size, single-side PCB for the clock circuit

Setting up the clock

After soldering is done, place the ICs in the respective bases. Now to set time, alarm, etc, keep SPDT switch S1 in Setting Mode position and switch on the power to the circuit. The clock will flash, showing only the hours and minutes display. Set the clock using switches S2 through S7. After the setting is over, change the position of S1 to Normal Mode.

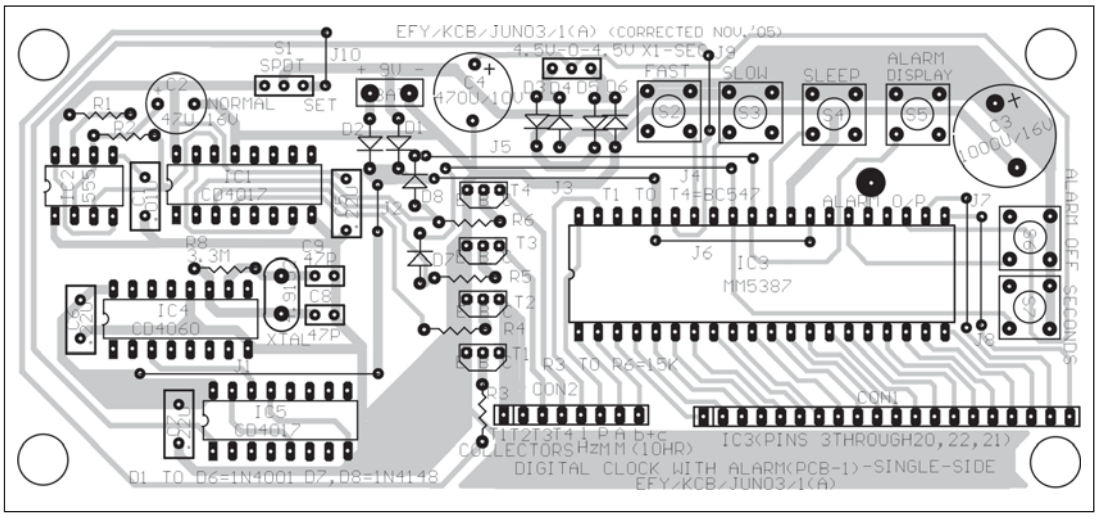


Fig. 4: Component layout for the PCB in Fig. 3

When SPDT switch S1 is in the normal mode, the display shows hours, minutes, and seconds in the upper 7-segment displays while one of AM/PM LEDs blinks according to this timing. The alarm time display also shows the same time in hours and minutes along with the blinking of either f or e segment in the lower 7-segment DIS7 unless the alarm is set to a particular time.

Normal mode. The functions of various keys (switches) used in the normal mode are as follows:

S4 (sleep): Used to display sleep timing in upper as well as lower 7-segment minutes LEDs, i.e. DIS3-DIS4 and DIS9-DIS10, simultaneously. This output can be used to control external appliances such as radios and TV sets from the sleep output.

S5 (alarm): Used to display alarm timing in upper as well as lower 7-segment LED displays simultaneously.

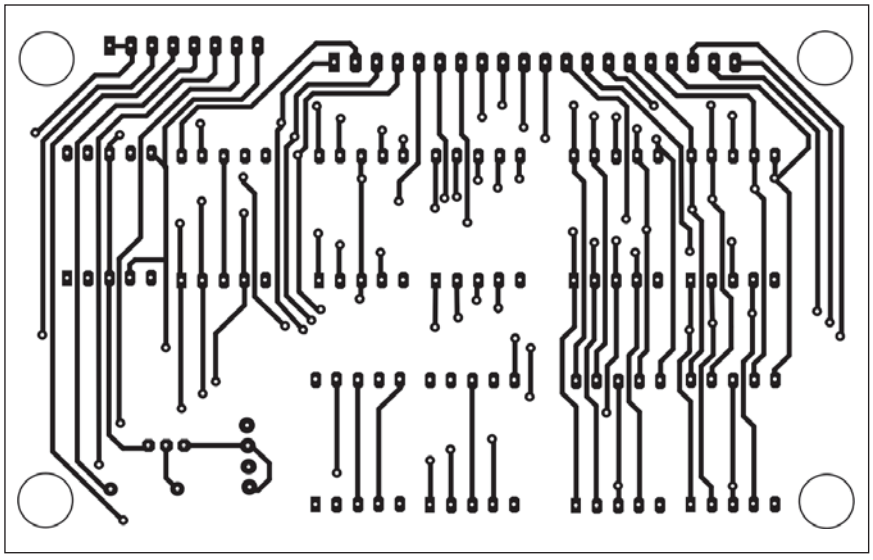


Fig. 5: Actual-size, solder-side PCB for display circuit

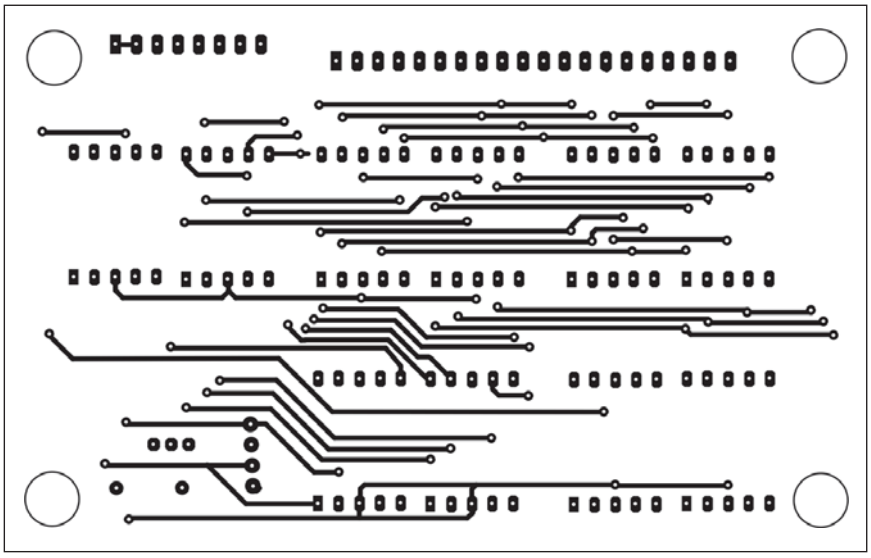


Fig. 6: Actual-size, component-side PCB for display circuit

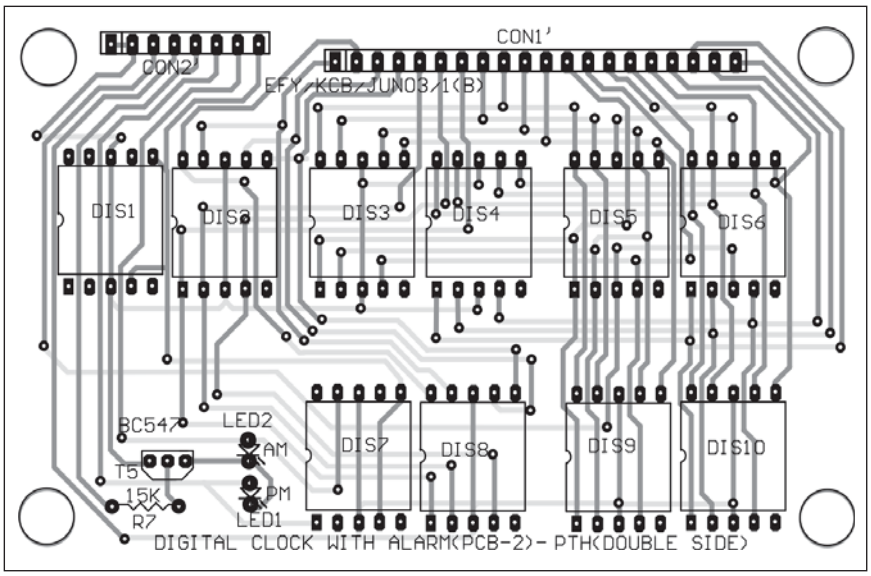


Fig. 7: Component layout for the double-side PCB of display unit

S6 (alarm off): Used to turn off the alarm timing. (It reactivates automatically after every 24 hours.)

S7 (seconds): Used to display units and tens digits of the seconds display along with the units digit of the minutes display in the upper 7-segment LEDs.

S8 (snooze): Used to put off the alarm for a short duration and activate it after every nine minutes for five times; this switch is not shown in the circuit.

Setting mode. When switch S1 is in the setting mode, the display shows only hours and minutes in the upper 7-segment displays, and seconds and alarm time displays remain off.

The functions of various keys (switches) used in the setting mode are as follows:

S2 (fast setting): Used for fast setting in increment mode only for hour, minute, and alarm time displays.

S3 (slow setting): Used for slow setting in increment mode only for hour, minute, and alarm displays.

S4 (sleep): Used to set sleep timing (in minutes). Note that it is a down-counting timer. When it reaches '00', the appliance connected to the sleep output will either turn on or off.

S6 (alarm off): Used to turn off the alarm timing. This pin should be connected to V_{SS} to make the alarm silent for a day or more.

S8 (snooze): Used to put off the alarm for a short duration and activate it after every nine minutes for five times; this switch is not shown in the circuit.

Setting of alarm

Keep switch S1 in the normal mode. Use fast/slow button switch to adjust the display to the desired alarm timing. Now move switch S1 to the setting mode and again use fast/slow button to adjust the exact normal timing in the upper display. Again move switch S1 to the normal mode.

Construction

The circuit can be assembled on a Vero board or by etching the PCB. Separate PCBs for the clock unit and the display unit are recommended so as to place them in a small cabinet. In the prototype built by the author, one PCB is placed on the top of the other, and the PCBs are fixed by nut-bolt with spacer. The display board and the main clock board are connected by a band of wires.

The actual-size, single-side PCB for the main clock circuit is shown in Fig. 3 with its component layout in Fig. 4. The actual-size, solder-side and component-side PCBs for the display circuit are shown in Fig. 5 and 6 with component layout in Fig. 7. Carefully connect the jumper wires on the display board, as a haphazard wiring may create problem while troubleshooting. The colon LEDs can be fixed on the component side of the PCB using some adhesive. Push-to-on switches can be soldered directly on the PCB of the display board if they are small enough to get accommodated.

The same idea as in the present circuit can be used to get month, day, alarm time, and sleep displays simultaneously on the display board using IC FA7317 clock. However, IC FA7317 clock chip is currently not available in the Indian market.

Note. The datasheet of IC MM5387 is included in the CD. ❑

Readers' comments:

Q1. Pin 25 of IC MM5387(clock chip) is being shown as the alarm output. I have connected an 8-ohm, 1W speaker to this output. The problem is when real time reaches the alarm time, no music or sound is produced by the speaker. What could be the reason? Please tell me the type of output at pin 25 of IC MM5387. Apart from this, the gadget is working properly.

V. Venkatesh
Chitradurga (Distt), Karnat

Q2. I have the following doubts:

1. In the circuit, the connections of 'seconds' display to IC MM5387 have not been used as per the datasheet: 7-segment displays DIS5 and DIS6 of tens and units digits of the 'seconds' display are connected to pins 10 through 15 and pins 16 through 22 of IC MM5387, respectively. In fact, these pins should be connected for the tens and units digits of the 'minutes' display, respectively.

But even then, the circuit is showing correct time and 'seconds' are incrementing correctly. How come?

2. For using a 3½-digit LCD in this project, what modifications are required? Can I connect pins of IC MM5387 directly to the LCD pins?

Venkatesh V.
Chitradurga, Karnataka

EFY: **A1.** For the connection at the alarm output, use a piezobuzzer instead of the speaker. The output at pin 25 of IC MM5387 will be a DC output and hence it cannot directly drive the speaker.

The author, S.K. Roushon., replies:

A2. First of all, I thank Mr Venkatesh for showing interest in my digital clock project. The answer to his first query can be explained by the action of switching on/off of a light bulb. This is also called 'multiplexing.'

Else, assume that you have one tap with two pipes, say, 'A' and 'B,' attached to it. 'A' is for hot water and 'B' is for cold water. Switches 'X' and 'Y' are attached to 'A' and 'B,' respectively, with the following result:

Tap	X (Hot)	Y (Cold)
Left turn	On	Off
Right turn	Off	On

So, if you turn the tap left, hot water comes out of the tap. And if you turn it right, cold water comes out. This principle has been used in my project. But in the circuit, switching takes place thousand times in a second. And this depends on the frequency of the multivibrator circuit using IC 555. I hope this answers well.

As regards the use of an LCD, I have not tried it. It is possible but extra care is needed.

PROGRAMMABLE LIGHT EFFECTS GENERATOR

SUNIL P.B.

Nowadays various types of lighting effect generators are available in the market, but these produce only two or three effects. In order to achieve a large number of lighting effects, you need to use a microprocessor-based circuit, which is quite complex and costly.

Here's an EPROM-based lighting effects generator circuit that can be used as a decorative lamp controller. It generates a number of lighting effects with variable speeds and varying intensities under software control. This is possible by the use of two preprogrammed EPROMs. As per the program, the circuit creates different effects like the dancing or running light.

Typical circuit using the up-counter

Fig. 1 shows the typical lighting effects generator circuit that controls light using the ramp signal. Here NE555 (IC13) is wired as an astable to produce clock pulse for CD4029 binary counter (IC14). IC CD4067 (IC15) is used for supplying separate 4-bit binary data for the multiplexers to control different light effects through the transistor (BC558) and UJT 2N2646.

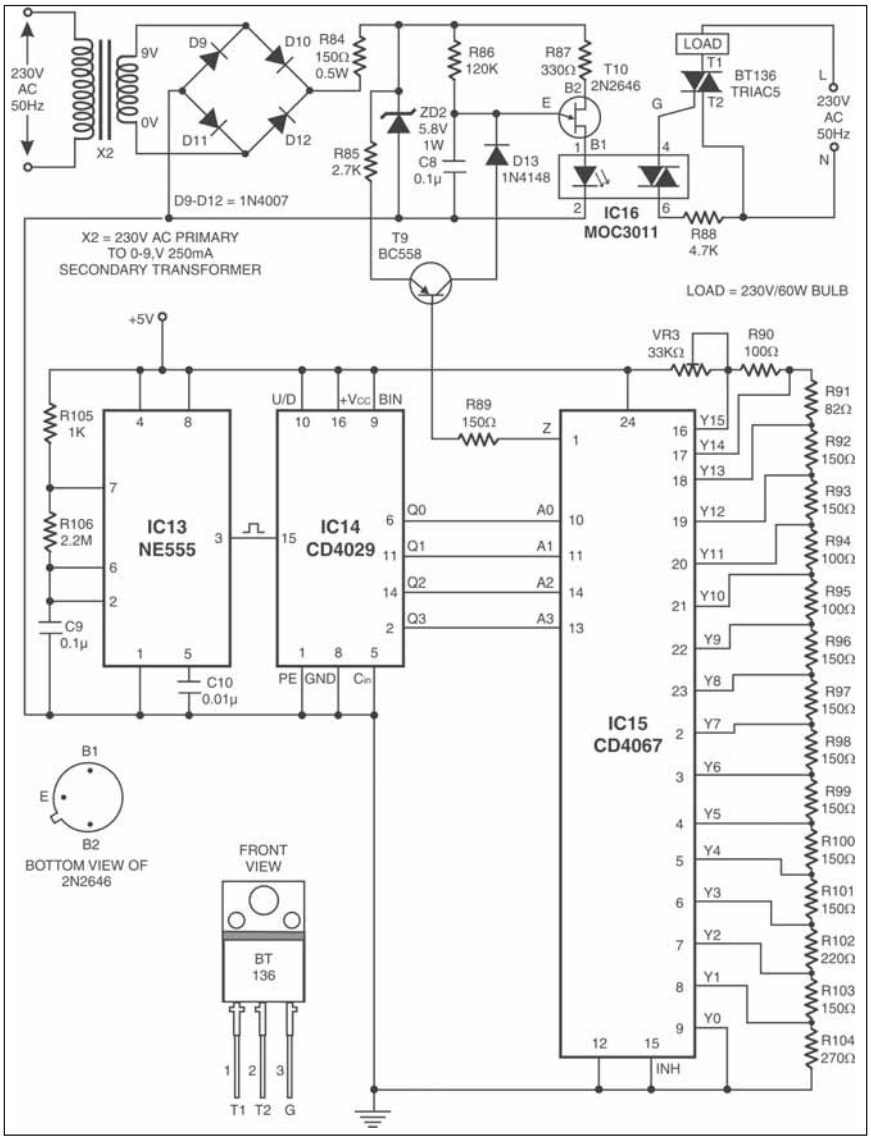


Fig. 1: Typical lighting effects generator circuit using the ramp signal

The UJT relaxation oscillator is wired to generate sawtooth pulse. Initially the UJT is in cut-off region and its internal input diode is reverse biased. As a result, capacitor C8 starts charging through resistor R86. When voltage across the capacitor becomes high enough, it forward biases internal input diode of UJT, and is discharged into the low-resistance region between the UJT's emitter and the optocoupler MOC3011 (IC16). Discharging continues until voltage across the capacitor is zero and the diode is again reverse biased. When the diode is reverse biased, capacitor C8 starts charging again. The process of charging and discharging produces a sawtooth pulse.

Triac BT136 is fired at different angles to get light intensity varying from zero to maximum. The UJT relaxation oscillator supplies short-duration current pulses for firing the triac. The short-duration pulses are coupled to the gate of triac BT136 through optocoupler MOC3011. Pedestal voltage control is used for firing the triac at different angles. The pedestal voltage is derived from a step-running ramp generator. The step-running ramp voltage is generated by the combination of multiplexer, resistor arrays, and binary counter.

When the binary counter output is 0000, the intensity of the bulb is zero. If the binary output is 1111, the bulb glows with the maximum intensity. In the intermediate binary outputs, the bulb glows with different intensities between zero and maximum.

Circuit description

The circuit in Fig. 2 uses EPROMs, in place of the binary counter, to get different effects through program control. It comprises four stages. Two EPROM ICs

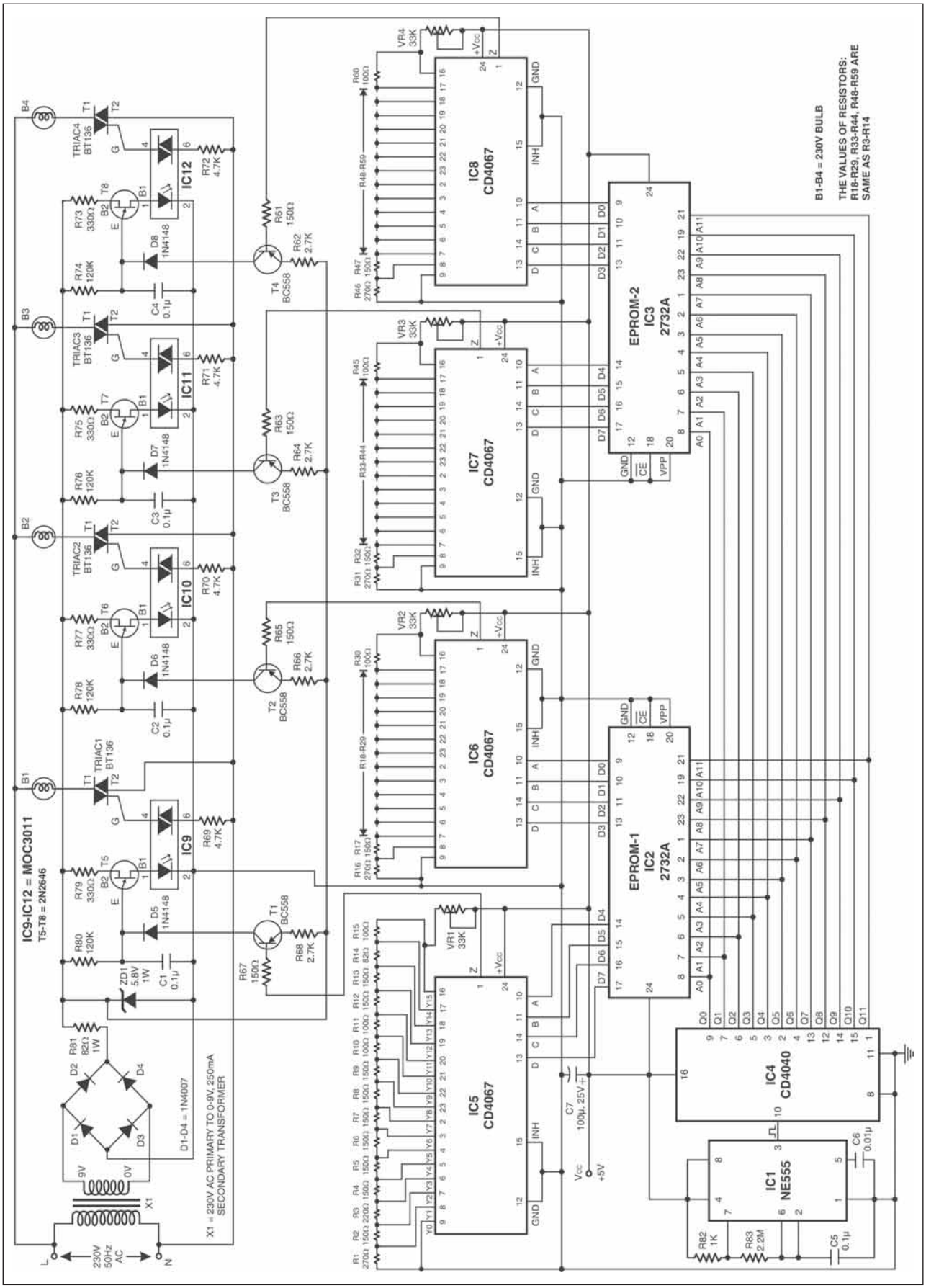


Fig. 2: EPROM-based lighting effects generator circuit

PARTS LIST

Semiconductors:

IC1	- NE555 timer
IC2, IC3	- 2732A EPROM (4k×8-bit)
IC4	- CD4040 12-stage binary counter
IC5-IC8	- CD4067 16-channel analogue multiplexer
IC9-IC12	- MOC3011 optocoupler
T1-T4	- BC558 pnp transistor
T5-T8	- 2N2646 unijunction transistor (UJT)
Triac 1-Triac 4	- BT136 triac
D1-D4	- 1N4007 rectifier diode
D5-D8	- 1N4148 switching diode
ZD1	- 5.8V, 1W zener diode

Resistors (all ¼-watt, ±5% carbon, unless stated otherwise):

R1, R16, R31, R46	- 270-ohm
R2, R4-R9, R12, R13 R17, R19-R24 R27, R28 R32, R34-R39, R42, R43, R47 R49-R54, R57, R58, R61, R63, R65, R67	- 150-ohm
R3, R18, R48, R33	- 220-ohm
R10, R11, R15, R25, R26, R30, R40, R41, R45, R55, R56, R60	- 100-ohm
R14, R29, R44, R59	
R62, R64	- 82-ohm
R66, R68	- 2.7-kilo-ohm
R69-R72	- 4.7-kilo-ohm
R73, R75, R77, R79	- 330-ohm
R74, R76, R78, R80	- 120-kilo-ohm
R81	- 82-ohm, 1-watt
R82	- 1-kilo-ohm
R83	- 2.2-mega-ohm
VR1-VR4	- 33-kilo-ohm preset

Capacitors:

C1-C5	- 0.1μF ceramic disk
C6	- 0.01μF ceramic disk
C7	- 100μF, 25V electrolytic

Miscellaneous:

X1	- 230V AC primary to 0-9V, 250mA Secondary transformer
B1-B4	- 230V, 60W bulb

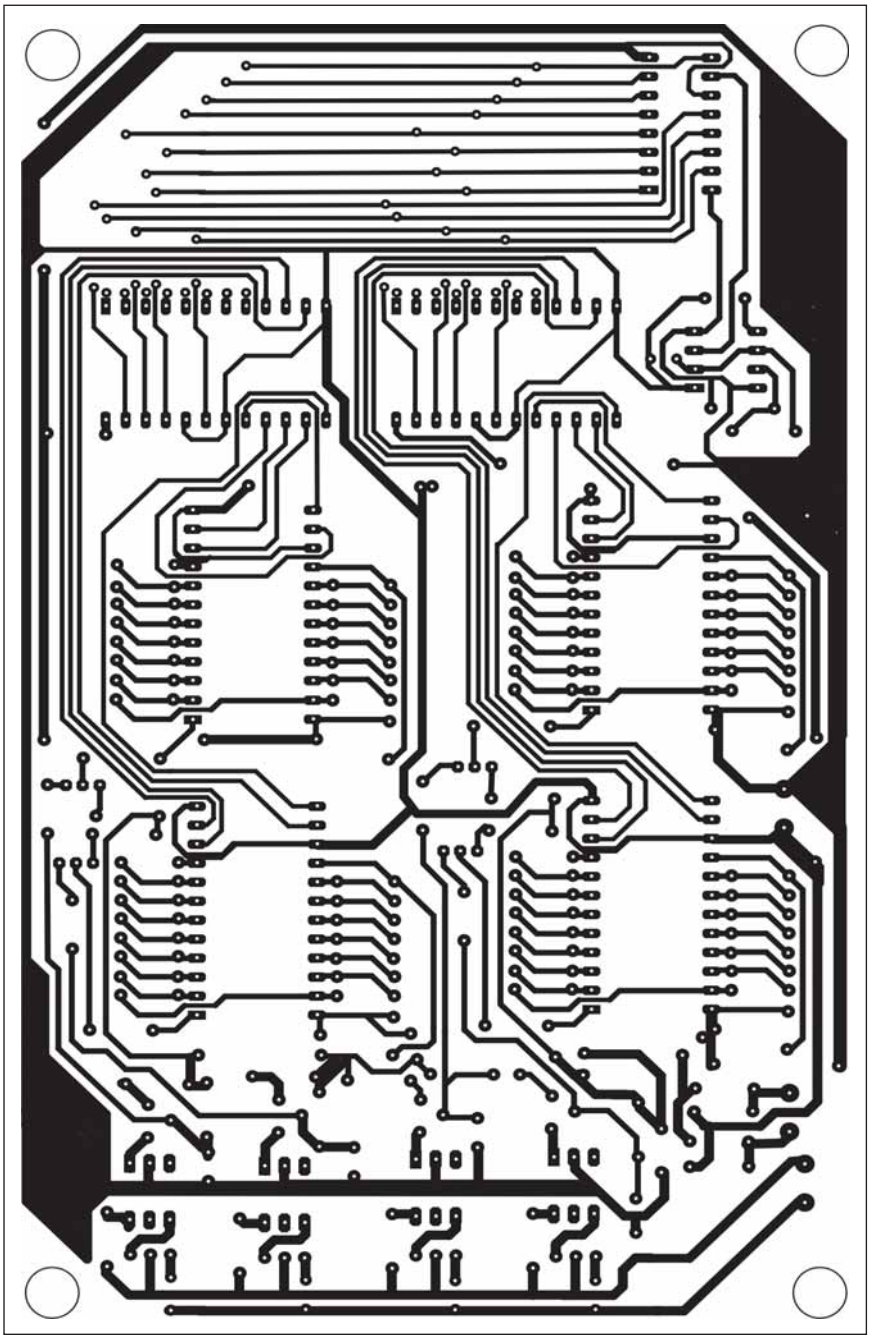


Fig. 3: Actual-size, single-side PCB for EPROM-based lighting effects generator

2732A (IC2 and IC3) hold the programs for creating different effects. IC1 (NE555) is wired as an astable multivibrator to produce clock pulse for the 12-bit binary counter CD4040 (IC4), which supplies addresses for the EPROMs. The output of IC4 increments by one at every clock pulse.

In order to get 16-bit data outputs with 4kB addresses/locations, the address lines of both EPROMs are connected in parallel. The 8-bit data lines D0 through D7 of each EPROM are divided into two separate 4-bit data outputs D0-D3 and D4-D7. The 4-bit data outputs of each EPROM are connected to the address inputs of two 16-channel analogue multiplexers (CD4067). As per the data inputs from the EPROM (IC 2732A), the multiplexer connects the corresponding input (Y0 through Y15) to the output Z. The output Z of the multiplexer controls the emitter of UJT 2N2646 through transistor BC558.

For getting a ramp voltage in steps, a group of resistors R1 through R15 is used at the output of IC5. Similarly, the groups of resistors R16 through R30, R31 through R45, and R46 through R60 are used at the outputs of IC6, IC7, and IC8, respectively. Each individual resistor drops the current in different levels. This ramp voltage output controls the firing angle of triac by using the pedestal voltage control method as explained above.

The actual-size, single-side PCB for the mains-operated light controller based on EPROM is shown in Fig. 3 with component layout in Fig. 4.

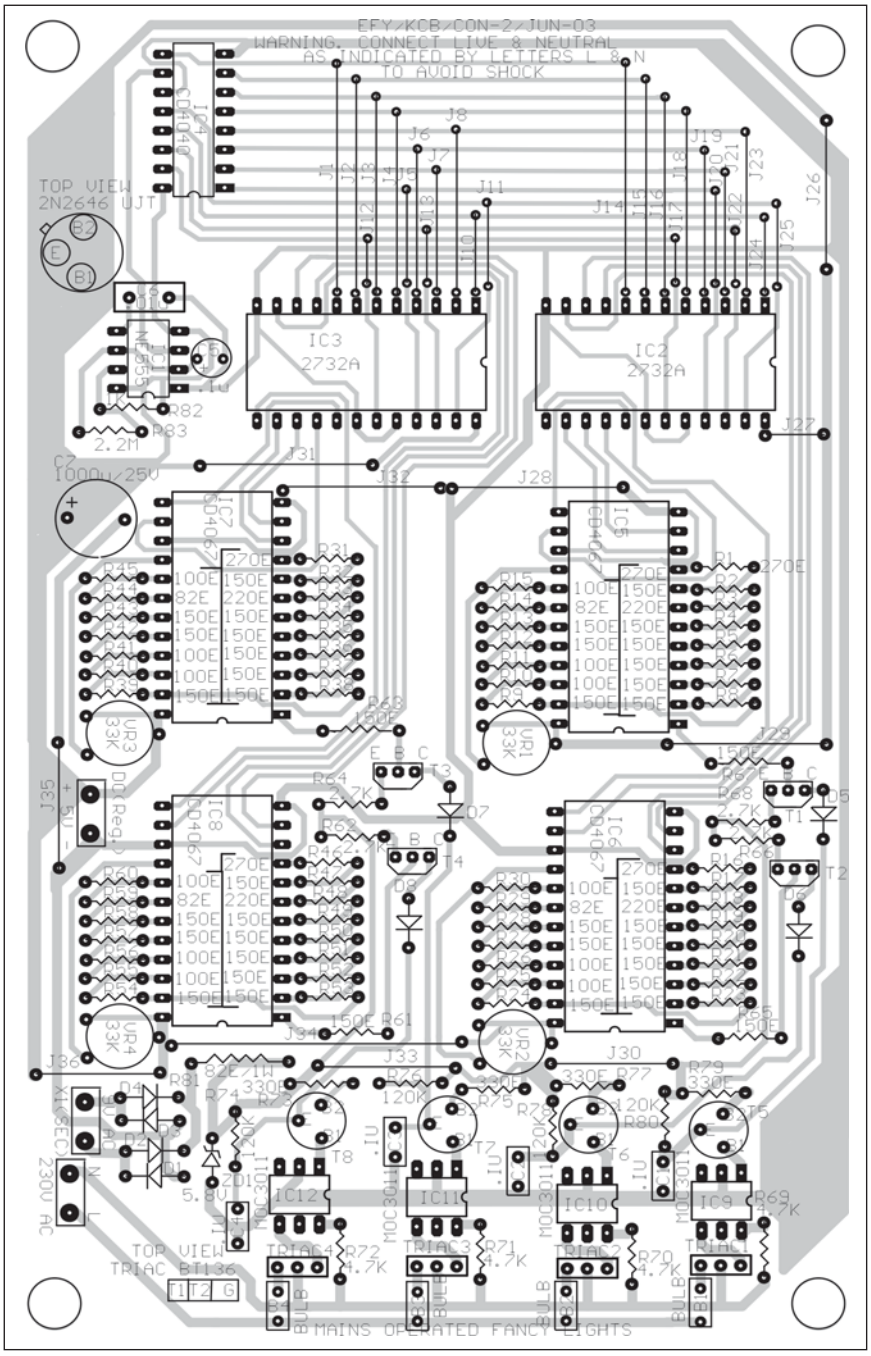


Fig. 4: Component layout for the PCB in Fig. 3

Caution. 1. Use sockets for ICs.

2. Even though the optocoupler (MOC3011) isolates the high-voltage section from the low-voltage section, care should be taken while touching the circuit to avoid shock hazards.

The program

Writing the program is very simple. The basic thing to keep in mind for writing the program is that if the control inputs to the multiplexers are 0000 the bulb goes off, and if the control inputs are 1111 the bulb glows with the maximum intensity. Any intermediate intensity is possible by making the binary inputs other than 0000 and 1111. You can easily create any effect (running, dancing, blinking etc) by writing the necessary program.

Variable lighting speeds are also possible with this circuit. Writing the same data again in the next location reduces the speed. In this way, any speed can be easily programmed for an effect without disturbing the master oscillator frequency. In all cases, for entering the code, take identical addresses for both the EPROMs.

Any type of EPROM programmer can be used for programming the EPROM, but its address location should start from 0000H. You must fill all the locations of EPROM with the necessary programs. Otherwise, leave the locations blank. The bulb glows continuously in the blanked locations.

We've tested the circuit using the sample coding shown in the table. Try to fill all the locations in the EPROMs because if any of the locations is left blank, the bulbs controlled by that EPROM glow continuously. So try to fill all the locations by repeating the effects. If you wish, you can create more effects by writing the necessary programs in the EPROM. ❑

ADDRESS CODES WITH DATA FOR VARIOUS LIGHTING EFFECTS

Address	Data for EPROM1	Data for EPROM2
Zero intensity to maximum intensity		
000H	00H	00H
001H	11H	11H
002H	22H	22H
003H	33H	33H
004H	44H	44H
005H	55H	55H
006H	66H	66H
007H	77H	77H
008H	88H	88H
009H	99H	99H
00AH	AAH	AAH
00BH	BBH	BBH
00CH	CCH	CCH
00DH	DDH	DDH
00EH	EEH	EEH
00FH	FFH	FFH
010H	00H	00H
011H	11H	11H
012H	22H	22H
013H	33H	33H
014H	44H	44H
015H	55H	55H

016H	66H	66H
017H	77H	77H
018H	88H	88H
019H	99H	99H
01AH	AAH	AAH
01BH	BBH	BBH
01CH	CCH	CCH
01DH	DDH	DDH
01EH	EEH	EEH
01FH	FFH	FFH
020H	00H	00H
021H	11H	11H
022H	22H	22H
033H	33H	33H
024H	44H	44H
025H	55H	55H
026H	66H	66H
027H	77H	77H
028H	88H	88H
029H	99H	99H
02AH	AAH	AAH
02BH	BBH	BBH
02CH	CCH	CCH
02DH	DDH	DDH
02EH	EEH	EEH
02FH	FFH	FFH

Maximum intensity to zero

030H	FFH	FFH
031H	EEH	EEH
032H	DDH	DDH
033H	CCH	CCH
034H	BBH	BBH
035H	AAH	AAH
036H	99H	99H
037H	88H	88H
038H	77H	77H
039H	66H	66H
03AH	55H	55H
03BH	44H	44H
03CH	33H	33H
03DH	22H	22H
03EH	11H	11H
03FH	00H	00H

Blinking effect with varying speed

040H	FFH	FFH
041H	00H	00H
042H	FFH	FFH
043H	00H	00H
044H	FFH	FFH
045H	00H	00H
046H	FFH	FFH
047H	FFH	FFH
048H	FFH	FFH
049H	00H	00H
04AH	FFH	FFH
04BH	FFH	FFH
04CH	FFH	FFH
04DH	00H	00H
04EH	00H	00H
04FH	FFH	FFH
050H	FFH	FFH
051H	FFH	FFH
052H	00H	00H
053H	00H	00H
054H	00H	00H
055H	FFH	FFH
056H	FFH	FFH
057H	00H	00H
058H	00H	00H
059H	00H	00H
05AH	FFH	FFH

Blinking with varying intensity

05BH	FFH	FFH
05CH	00H	00H
05DH	EEH	EEH
05EH	00H	00H
05FH	DDH	DDH
060H	00H	00H
061H	CCH	CCH
062H	00H	00H
063H	BBH	BBH
064H	00H	00H
065H	AAH	AAH
066H	00H	00H
067H	99H	99H
068H	00H	00H
069H	88H	88H

Dancing Effect

06AH	F0H	F0H
06BH	0FH	0FH
06CH	F0H	F0H
06DH	0FH	0FH
06EH	F0H	F0H
06FH	0FH	0FH
070H	F0H	F0H
071H	F0H	F0H
072H	F0H	F0H
073H	0FH	0FH
074H	0FH	0FH
075H	0FH	0FH
076H	F0H	F0H
077H	F0H	F0H
078H	F0H	F0H
079H	FFH	00H
07AH	FFH	00H
07BH	00H	FFH
07CH	00H	00H
07DH	FFH	FFH
07EH	FFH	FFH
07FH	00H	00H
080H	00H	FFH

Each bulb intensity from zero to maximum

081H	10H	00H
082h	20H	00H
083h	30H	00H
084h	40H	00H
085H	50H	00H
086H	60H	00H
087H	70H	00H
088H	80H	00H
089H	90H	00H
08AH	A0H	00H
08BH	B0H	00H
08CH	C0H	00H
08DH	D0H	00H
08EH	E0H	00H
08FH	F0H	00H
090H	F1H	00H
091H	F2H	00H
092H	F3H	00H
093H	F4H	00H
094H	F5H	00H
095H	F6H	00H
096H	F7H	00H
097H	F8H	00H
098H	F9H	00H
099H	FAH	00H
09AH	FBH	00H
09BH	FCH	00H
09CH	FDH	00H
09DH	FEH	00H
09EH	FFH	00H
09FH	FFH	10H
0A0H	FFH	20H
0A1H	FFH	30H
0A2H	FFH	40H
0A3H	FFH	50H
0A4H	FFH	60H
0A5H	FFH	70H
0A6H	FFH	80H
0A7H	FFH	90H
0A8H	FFH	A0H
0A9H	FFH	B0H
0AAH	FFH	C0H
0ABH	FFH	D0H
0ACH	FFH	E0H
0ADH	FFH	F0H
0AEH	FFH	F1H
0AFH	FFH	F2H
0B0H	FFH	F3H
0B1H	FFH	F4H
0B2H	FFH	F5H
0B3H	FFH	F6H
0B4H	FFH	F7H
0B5H	FFH	F8H
0B6H	FFH	F9H

0B7H	FFH	FAH
0B8H	FFH	FBH
0B9H	FFH	FCH
0BAH	FFH	FDH
0BBH	FFH	FEH
0BCH	FFH	FFH

Running effect

0BDH	00H	00H
0BEH	F0H	00H
0BFH	0FH	00H
0C0H	00H	F0H
0C1H	00H	0FH
0C2H	00H	00H
0C3H	00H	0FH
0C4H	00H	F0H
0C5H	0FH	00H
0C6H	F0H	00H
0C7H	00H	00H

Running hole effect

0C8H	FFH	FFH
0C9H	0FH	FFH
0CAH	F0H	FFH
0CBH	FFH	0FH
0CCH	FFH	F0H
0CDH	FFH	FFH
0CEH	FFH	F0H
0CFH	FFH	0FH
0D0H	F0H	FFH
0D1H	0FH	FFH
0D2H	FFH	FFH

Various Effect

0D3H	00H	FFH
0D4H	11H	EEH
0D5H	22H	DDH
0D6H	33H	CCH
0D7H	44H	BBH
0D8H	55H	AAH
0D9H	66H	99H
0DAH	77H	88H
0DBH	88H	77H
0DCH	99H	66H
0DDH	AAH	55H
0DEH	BBH	44H
0DFH	CCH	33H
0E0H	DDH	22H
0E1H	EEH	11H
0E2H	FFH	00H
0E3H	FFH	00H
0E4H	EEH	11H
0E5H	DDH	22H
0E6H	CCH	33H
0E7H	BBH	44H
0E8H	AAH	55H
0E9H	99H	66H
0EAH	88H	77H
0EBH	77H	88H
0ECH	66H	99H
0EDH	55H	AAH
0EEH	44H	BBH
0EFH	33H	CCH
0E0H	22H	DDH
0F1H	11H	EEH
0F2H	00H	FFH

❑

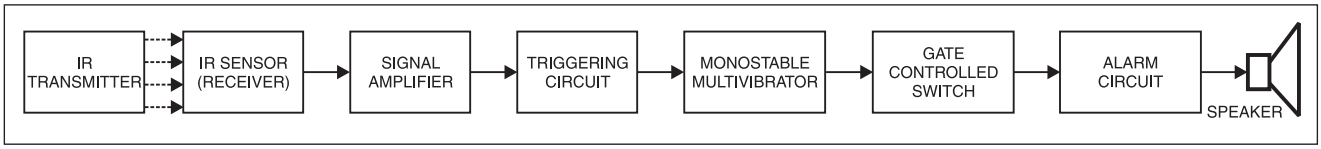
DOOR-OPENING ALARM WITH REMOTE CONTROL

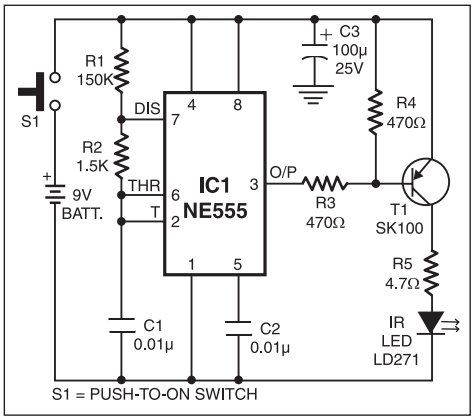
PRADEEP G.

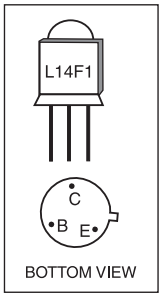
Commercially available electronic security alarms with remote control are generally very expensive. Here is a circuit, with construction details, for a low-cost door-opening alarm with remote control. However, you need not despair as here we present a low-cost door opening alarm with remote control. It uses readily available components and is easy to assemble.

The circuit

Fig. 1 shows the block diagram of the door-opening alarm with IR remote control. The circuit has two main parts, namely, a small infrared remote transmitter unit and a receiver unit with alarm.





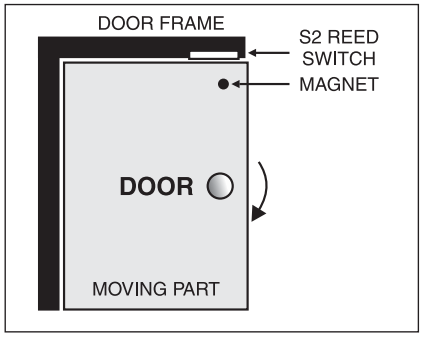


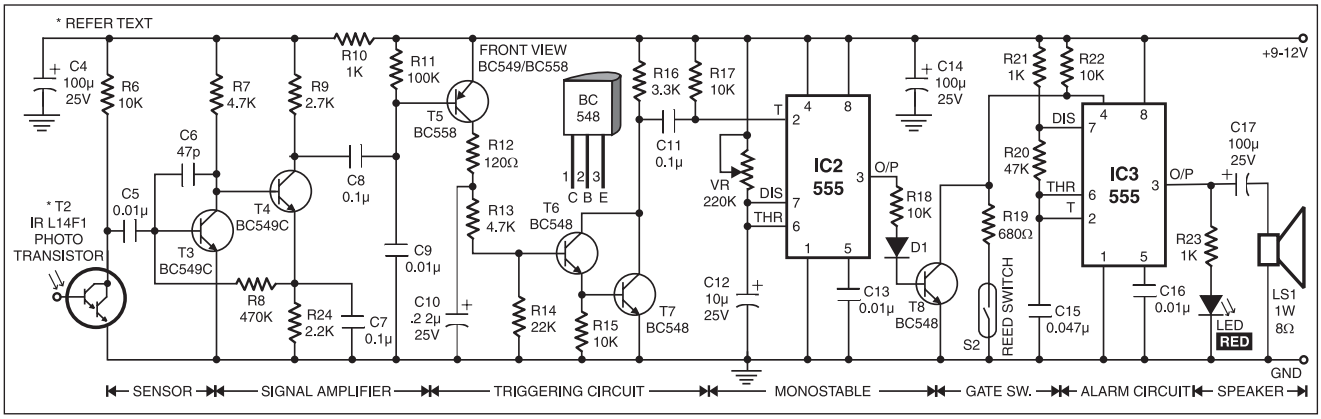
Transmitter unit. Fig. 2 shows the circuit of the remote transmitter unit. Its working is very simple. IC NE555 (IC1) is used in astable multivibator mode to operate a frequency of 1 kHz. A pnp transistor (T1) drives the IR LED.

Connect a +9V battery to the circuit. Now on pressing switch S1, the transmitter emits a modulated infrared beam up to 7 metres without the need of any lens or reflector.

Receiver unit with alarm. The circuit diagram of the IR receiver unit with alarm is shown in Fig. 4. IR signals sent by the transmitter are received by Darlington IR phototransistor L14F1 (T2) whose bottom view is shown in Fig. 3. Thus phototransistor T2 is used here as a sensor.

As IR signals are very weak, these require amplification. So the signals are amplified by the amplifier stage comprising transistors T3 and T4. Amplified signals are fed to the triggering circuit com-





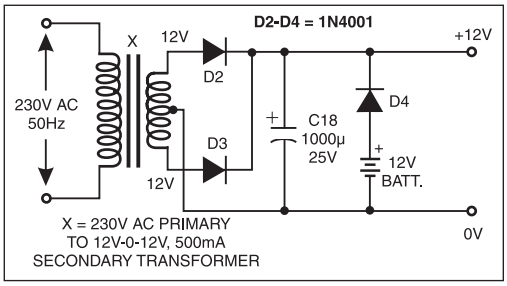


Fig. 6: Circuit of power supply with battery backup

prising transistors T5, T6, and T7 to trigger the monostable multivibrator wired around IC2.

When IC2 is triggered at pin 2, its output pin 3 goes high for 5 to 10 seconds. Time delay can be set by a 220k preset (VR). During this time, transistor T8 conducts to keep reset pin 4 of astable multivibrator IC3 low. Then the alarm gets disabled.

Within the preset time period if somebody opens the door, i.e. the magnet is moved away from reed switch S2, pin 4 of IC3 goes low due to the conduction of transistor T8 and hence the alarm is not activated.

After completion of the preset time period if somebody opens the door, reed switch S2 also gets opened and pin 4 of IC3 goes high due to non-conduction of transistor T8 and hence the alarm is activated.

The actual use of the remote control is that you can disable the alarm while you open the door. You can keep the remote control in your pocket. When you enter the room or go out from the room, simply direct remote control to the sensing phototransistor and momentarily press switch S1. Thus the alarm is disabled for 5 to 10 seconds. So during this time, you can open the door without activation of the alarm. After this time duration completes, if anyone tries to open the door, the alarm will sound.

Assembling

The door opening alarm uses a simple magnet-operated two-leads reed switch as a sensor. Reed switch S2 is fitted on the door frame using an adhesive

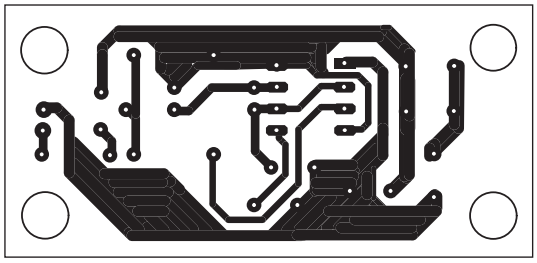


Fig. 7: Actual-size, single-side PCB of transmitter unit

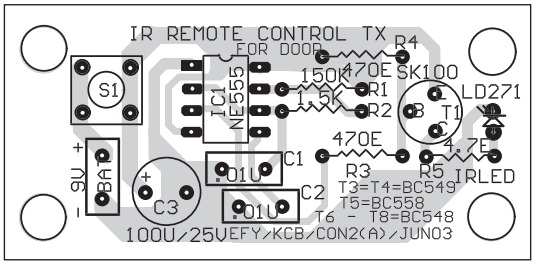


Fig. 8: Component layout for the PCB in Fig. 7

PARTS LIST

Semiconductors:

IC1-IC3	-	NE555 timer
T1, T9	-	SK100 pnp transistor
T2	-	IR L14F1 photo Darlington
T3-T4	-	BC549C npn transistor
T5	-	BC558 pnp transistor
T6-T8	-	BC548 npn transistor
T10	-	2N3054 pnp power transistor
IR-LED	-	LD271 infrared LED
LED	-	5mm red LED
D1	-	1N4148 switching diode
D2-D5	-	1N4001 rectifier diode

Resistors (all ¼-watt, ±5% carbon, unless stated otherwise):

R1	-	150-kilo-ohm
R2	-	1.5-kilo-ohm
R3, R4	-	470-ohm
R5	-	4.7-ohm
R6, R15, R17, R18, R22, R25	-	10-kilo-ohm
R7, R13	-	4.7-kilo-ohm
R8	-	470-kilo-ohm
R9	-	2.7-kilo-ohm
R10, R21, R23	-	1-kilo-ohm
R11	-	100-kilo-ohm
R12	-	120-ohm
R14	-	22-kilo-ohm
R16	-	3.3-kilo-ohm
R19	-	680-ohm
R20	-	47-kilo-ohm
R24	-	2.2-kilo-ohm
R26	-	100-ohm, 1W
VR	-	220k preset

Capacitors:

C1, C2, C5, C9, C13, C16	-	0.01µF ceramic disk
C7, C8, C11	-	0.1µF ceramic disk
C3, C4, C14, C17	-	100µF, 25V electrolytic
C6	-	47pF ceramic disk
C10	-	2.2µF, 25V electrolytic
C12	-	10µF, 25V electrolytic
C15	-	0.047µF ceramic disk
C18	-	1000µF, 25V electrolytic

Miscellaneous:

X	-	230V AC primary to 12V-0-12V, 500mA secondary transformer
S1	-	Tactile switch
S2	-	Reed switch
LS1	-	8-ohm, 1W speaker
LS2	-	5-ohm, 10W speaker
	-	Magnet
	-	IC bases
	-	+12V battery
	-	+9V battery

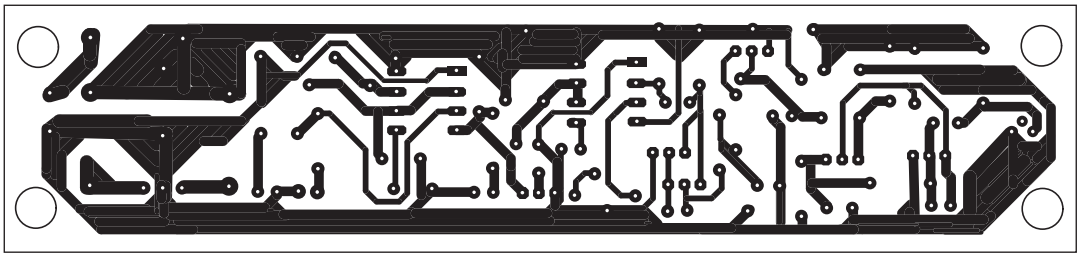


Fig. 9: Actual-size, single-side PCB for receiver unit

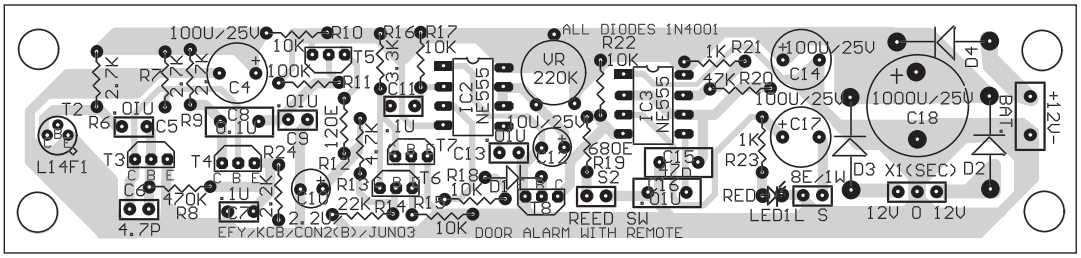


Fig. 10: Component layout of the PCB in Fig. 9

like Feviquick. The magnet is fitted on the moving part of the door as shown in Fig. 5.

When the door is fully closed, the magnet is close to switch S2 and the internal leads of the switch get shorted. When the door is opened, the magnet moves away from the reed switch and hence the internal leads of the reed switch get opened.

Assemble the transmitter unit and

the alarm unit on separate PCBs. The PCB of the transmitter should be small. All components, excluding timer IC 555, can be directly soldered on the PCB. Use 8-pin IC bases for timer IC. Try to keep the length of the wire between the IR phototransistor and the receiver PCB as small as possible. Don't overheat the sensor while soldering. Use a 25W soldering iron for soldering.

The unit requires back-up during power supply failure. Therefore use a 12V DC power supply with battery for back-up as shown in Fig. 6. Connect this power supply to the IR receiver unit with alarm.

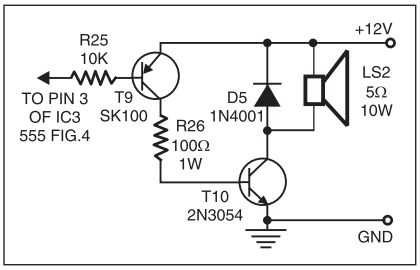


Fig. 11: Power amplifier circuit for loud sound

The actual-size, single-side PCB for the transmitter circuit (Fig. 2) is shown in Fig. 7 and its component layout in Fig. 8. The actual-size, single-side PCB for the receiver circuit with alarm (Fig. 4) and power supply (Fig. 6) is shown in Fig. 9, and its component layout in Fig. 10.

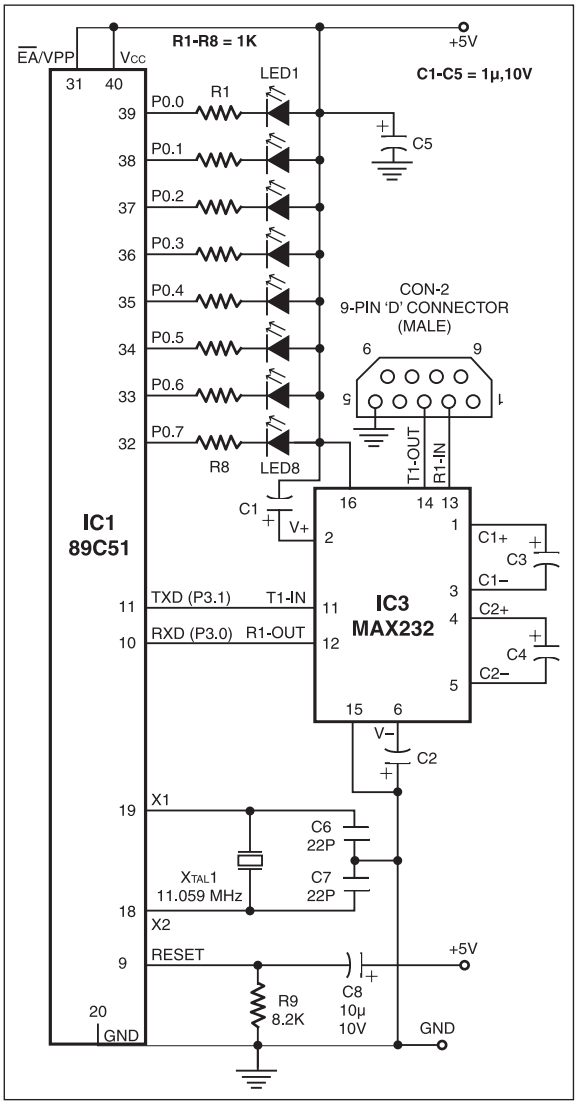
If you want the alarm to sound loudly during the unauthorised opening of the gate, use the power amplifier circuit shown in Fig. 11 with another suitable power supply. This circuit uses another power supply with a 230V AC primary to 12V-0-12V, 2A secondary transformer and two diodes of 2A rating (D2 and D3). ❑

MICROCONTROLLER-DRIVEN DATA DISPLAY

A.R. KARKARE

This project shows as to how you can use the Atmel microcontroller AT89C51 to drive an LCD display module and in turn use it as a handheld device to set the parameters of a control unit through RS-232 serial link.

Figs 1 and 2 show the circuits of a microcontroller-driven control unit and microcontroller-driven remote handheld device comprising LCD module, respectively. The circuit around IC1 (IC AT89C51) is configured as a control unit, while the circuit around IC2 (another IC AT89C51) is configured as the LCD driver unit. The two units are connected via an RS-232 serial link. The combination of an 8.2k resistor (R) and a 10μF capacitor (C) provides hardware power-on-reset to IC1 and IC2 at their pin 9. A 11.059MHz crystal is connected between pins 18 and 19 of microcontrollers IC1 and IC2 each to generate the required clock and baud rate of 9600.



Eight LEDs are connected to pins 39 (P0.0) through 32 (P0.7) of IC1, so we can see the status of each pin of port 0. Txd (pin 11) and Rxd (pin 10) are used to transmit and receive serial data through IC MAX232. IC3 and IC4 (MAX232) serve the purpose of linking the microcontrollers. Pin 14 (T1 OUT) of IC3 is connected to pin 13 (R1 IN) of IC4 and vice versa. The control unit contains the program 'contr.asm' to send and receive data to the handheld device (LCD module).

IC2 contains the program 'module.asm' to drive the LCD. A 16-character x 4-row LCD display is used to display the day-month-year. The LCD module is interfaced through 8-bit data bus of IC2 on its port 2 (pins 21 through 28). These pins are pulled high through the 10k resistor network. Internal registers of the LCD module are selected by pin 1 (P1.0) of IC2. The Write and Chip-Enable signals of LCD module are connected to pins 2 (P1.1) and 3 (P1.2) of IC2, respectively.

Backlight current (intensity) is controlled through series resistor R12 at pin 16 of the LCD module. The contrast and viewing angle are controlled through preset VR1 at pin 3 of the LCD module.

Four pins of port 1 (pins 4 through 7) are used to sense which key has been pressed. The keys are Esc, Ok, Up, and Down. Usually, pins 4 through 7 are held high through 4.7k resistors, but any of the pins can be pulled down using the corresponding switches S1 through S4.

RS-232 link between the two circuits serves the purpose of transferring serial data from one microcontroller to the other.

It is assumed that the control unit has some basic data, say, someone's birthday,

PARTS LIST

Semiconductors:

IC1, IC2	- AT89C51 microcontroller
IC3, IC4	- MAX232, RS-232 level converter
LCD module	- 16-character×4-line type
LED1-LED8	- Red LED

Resistors (all ¼-watt, ±5% carbon, unless stated otherwise):

R1-R8	- 1-kilo-ohm
R9, R10	- 8.2-kilo-ohm
R11-R14	- 4.7-kilo-ohm
R15	- 5-kilo-ohm
R16	- 18-ohm
RNW1	- 10-kilo-ohm×8 *SIL9 resistor network
RNW2	- 4.7-kilo-ohm×4 SIL5 resistor network
VR1	- 5-kilo-ohm preset

(***Note.*** *Serial-in-line 9-pin resistor, where pin 1 is a common pin.)

Capacitors:

C1-C5, C9-C13	- 1μF, 10V electrolytic
C6, C7, C14, C15	- 22pF ceramic disk
C8, C16	- 10μF, 10V electrolytic

Miscellaneous:

X_{TAL1}, X_{TAL2}	- 11.059MHz
S1-S4	- Push-to-on tactile switch
Con-1	- 9-pin 'D' female connector
Con-2	- 9-pin 'D' male connector

stored in it. The day, month, and the year data are stored at 30H, 31H, and 32H RAM locations, respectively.

When the remote handheld device (LCD module) is connected to the control unit through RS-232 link (IC MAX232), IC2 is reset to start functioning. The data stored in the control unit is displayed on the LCD screen. The user can then select the data (day, month or year). To change the data, increment or decrement it using Up or Down key, and then transfer the data back to the control unit.

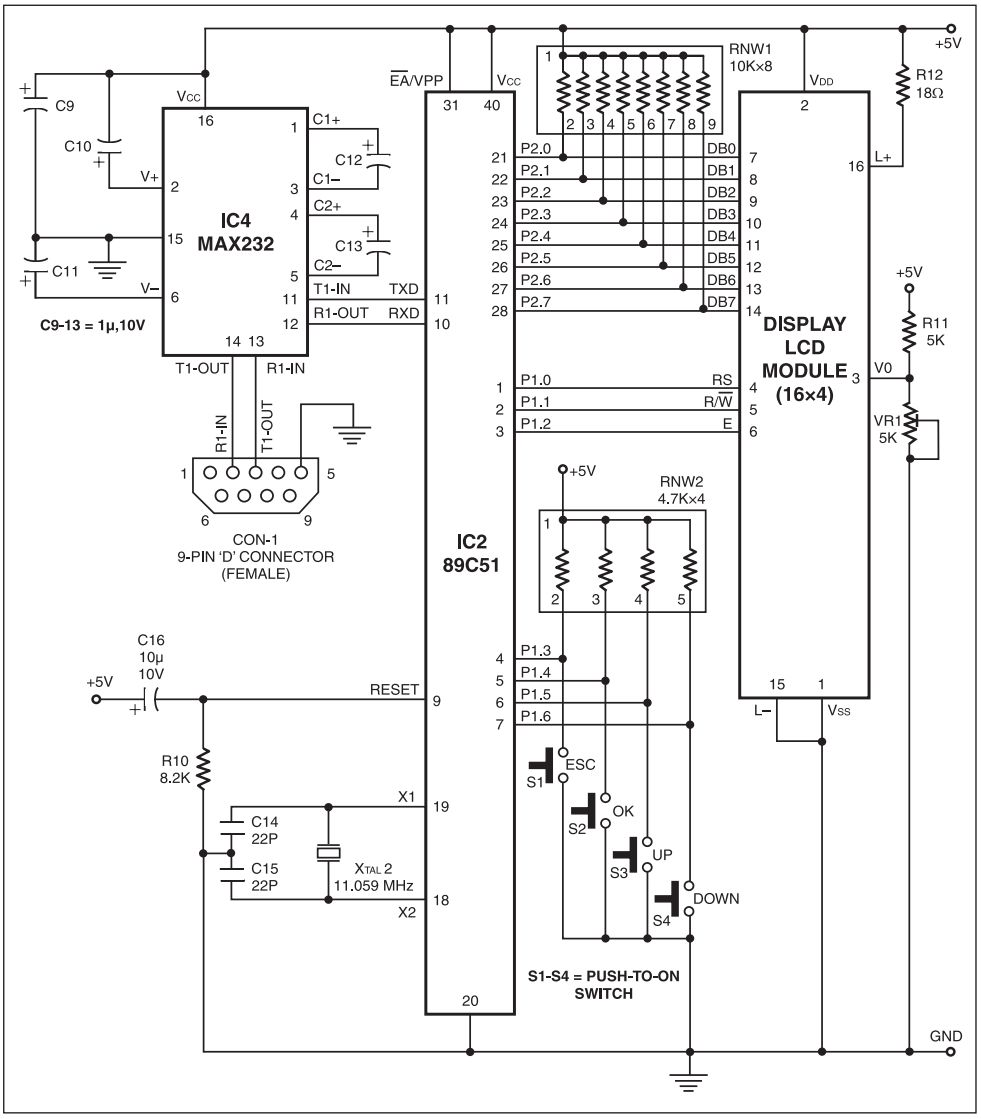


Fig. 2: Handheld unit comprising LCD module

Software

In the beginning section of the assembly file (refer Fig. 3 and the module.lst file), RAM locations are reserved for saving various variables such as the day's units and tens digits.

One location (45H) has been defined for sensing the flag to find whether serial port has been interrupted or not. Port pins connected to pins 4 through 6 of the LCD module are defined as 'rs', 'rw', and 'en'. Keys Esc, Ok, Up, and Down are defined as Port 1, which are connected to pins 4 through 7 of IC2, respectively.

The main program starts at location 0000H, while a jump instruction has been set at location 0023H for the serial port interrupt service routine (ISR). Whenever the serial port is interrupted, the program is automatically branched to location 0023H.

Start. The main program starts at location 0030H. Initially the stack pointer is initialised to some safe location where it will not get disturbed by normal routines of the program. Timer 1 is set as a NOT-gated timer for 8-bit auto-reload function mode. The reload value of timer 1 is set for generating a baud rate of 9600 bits per second. The SCON register is set for Mode 1 operation and is kept ready for reception.

Start timer 1 and set the required interrupt request bits as enabled. The interrupt flag is kept cleared to start. Now proceed as per the flowchart shown in Fig. 3.

A few steps after the 'clr intflg' instructions and before step1 are for initialising the LCD module.

Step 1. Screen 1, screen 2, etc to be displayed on the LCD module are predefined as scr1, scr2, etc at respective locations. As the program enters step 1, it first sets the data pointer to point at the first screen to be displayed. The setup subroutine displays the screen. The first screen displayed is a welcome massage. The program waits for the user to press Ok key to come out from the welcome screen display. When the user presses Ok key, the program control passes to Step 2.

Step 2. The program now displays the birthday screen, indicating day, month, and year. A small arrow pointer (>) indicator gets added at LCD location C0H, so the arrow points at 'day', indicating that the parameter 'day' is being selected.

The first character of each line on the LCD module has a unique address: The first character of first, second, third, and fourth lines has address as 80H, C0H, 90H, and D0H, respectively.

As the program executes the add_day, add_month, and add_year subroutines, the day, month, and year data is retrieved from the master IC 89C51 (IC1), converted into proper ASCII format, and saved at LCD locations. The display now shows the day, month, and year also on the LCD screen.

If the user wishes to select month or year, he needs to press Down key and shift the arrow pointer to the required selection place. On pressing Down key, the arrow pointer shifts down.

Similarly, on pressing Up key, the arrow pointer shifts up. This way the user can select the parameter he wishes to

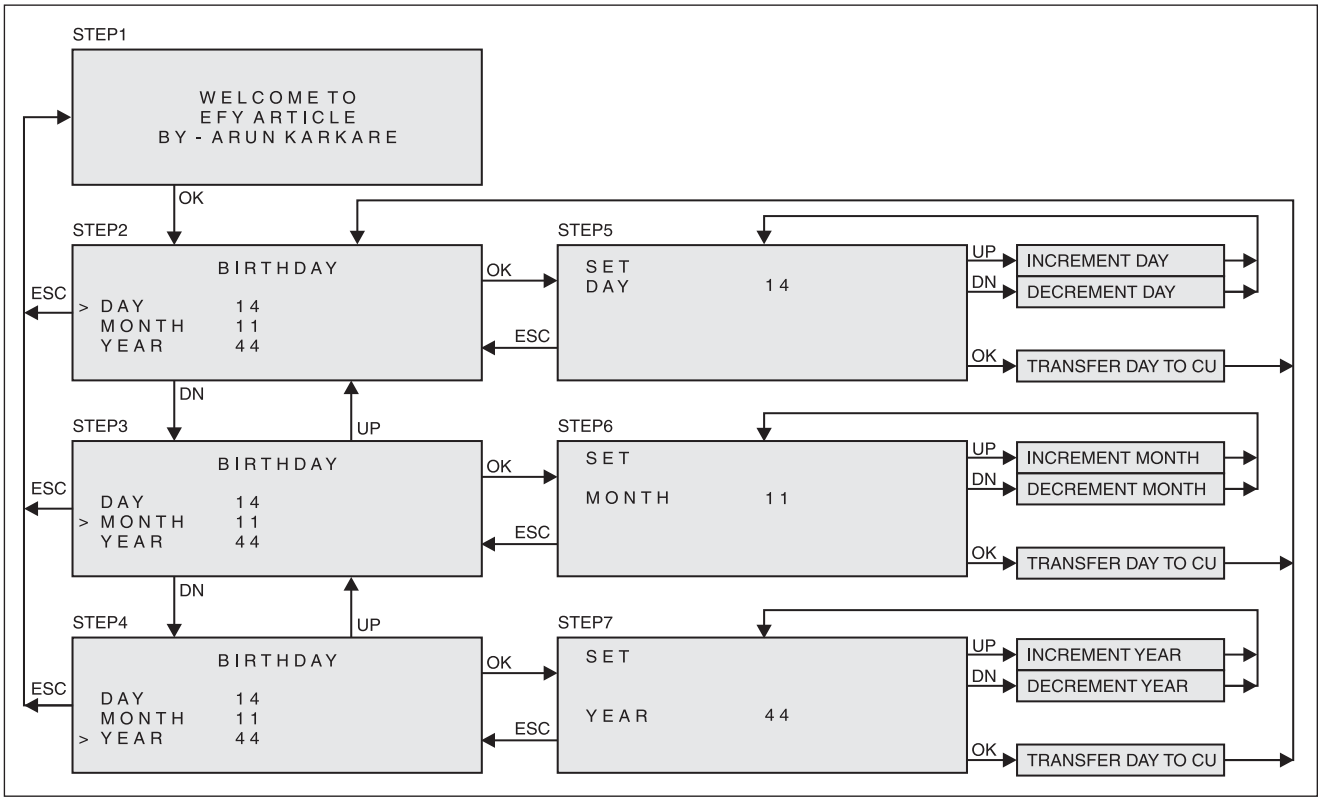


Fig. 3: Flowchart of microcontroller-driven data display

change. In case no parameter is to be selected, by simply pressing Esc key, the user can go back to Step 1, which is the welcome screen. Once the user has selected the parameter, pressing Ok key takes the program to the next step.

Step 3. Here the screen displays all the birthday characters, except the arrow has been shifted to indicate month.

Step 4. Here also the screen displays all the birthday characters, except the arrow has been shifted to indicate year.

Step 5. Depending upon the user's selection of day or month or year, the program branches to Step 5 or Step 6 or Step 7, where the screen displays 'set day' or 'set month' or 'set year', respectively.

On screen 5, the LCD displays 'set day'. The day then gets added on the screen. At key5 label, the program checks, which key is pressed. As long as no key is pressed, the program keeps looping back to key5 label.

When the user presses Up key, the parameter increments, as the 'advance day' and 'display day' subroutines are called in. Similarly, by pressing Down key, the parameter decrements.

During the 'advance day' subroutine, the program first checks whether the day is already 31. If so, it resets the day to 01, and doesn't allow it to increment to 32. Similarly, the month doesn't go beyond 12 and the year doesn't go beyond 99. However, if the user is decrementing the day parameter, the program first checks whether the day is already 01. If so, it resets the day to 31, the month to 12, and the year to 99.

Whenever the desired value of the day is seen on the screen, pressing Ok key takes the program to transfer the day data to the master IC 89C51 (IC1). The trfr_day subroutine transfers the value to the appropriate RAM location in the control unit and returns to the step2 screen.

Steps 6 and 7 are similar to Step 5.

As soon as the control unit of IC1 sends some data to the serial port, the serial interrupt at location 0023H gets activated and the program control is passed to the serial port by the spint ISR (serial port interrupt program).

spint subroutine. First, all the interrupts are disabled, since we do not want any interrupt while serving this subroutine. Pushing the program status word (psw) on the stack saves any useful information on the psw and accumulator. The sbuf register is then read and the same is stored at register B. 'ri' bit is then cleared for receiving the next character; flag is set to indicate the interrupt had occurred, and finally the program returns from the subroutine.

Send subroutine. The program first disables all the interrupts and clears the transmission completion flag. Then it loads the buffer register to start the transmission from IC2 to the control unit (IC1). As long as 'ti' bit remains low, we need to wait. When the transmission is over, 'ti' bit goes high. The program then enables the interrupt and returns to the main control.

Setup subroutine. The program first sets the address pointer (register r2) to the first-line, first-column position (80H) of the LCD. It writes this address to the LCD using the wi subroutine. The program then gets the character from the screen data library and writes data to the LCD using the wd subroutine. The setup subroutine displays the character on the LCD screen.

Both the data pointer and the address pointer (register r2) are then incremented. The program checks whether the first line of LCD has been written. If so, it modifies the address pointer to the second line, which is C0H. Similarly, when the second line is over, the third-line, first-character address is set, and then fourth-line, first-character address is set as address pointer.

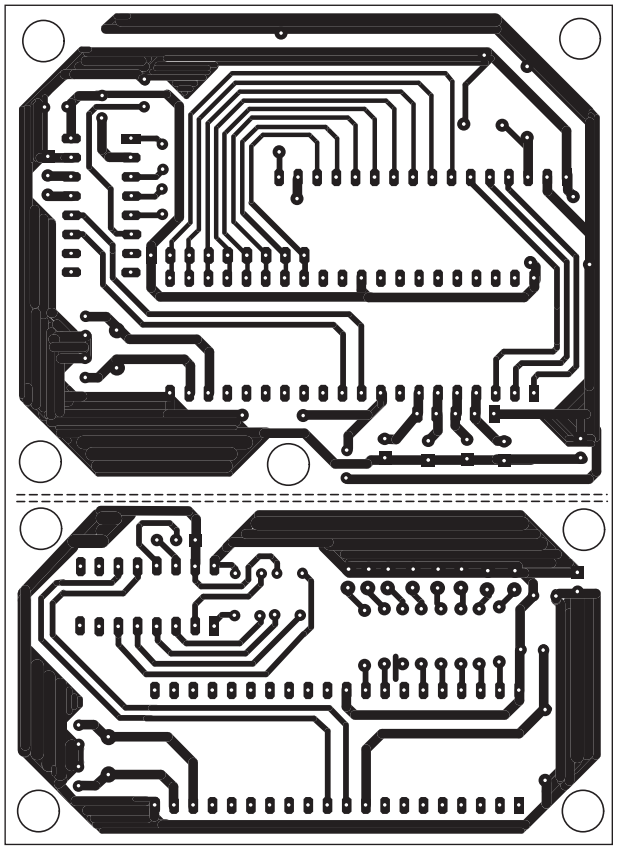


Fig. 4: Actual-size, single-side PCB layout for the handheld unit comprising LCD module (above) and the control unit (below)

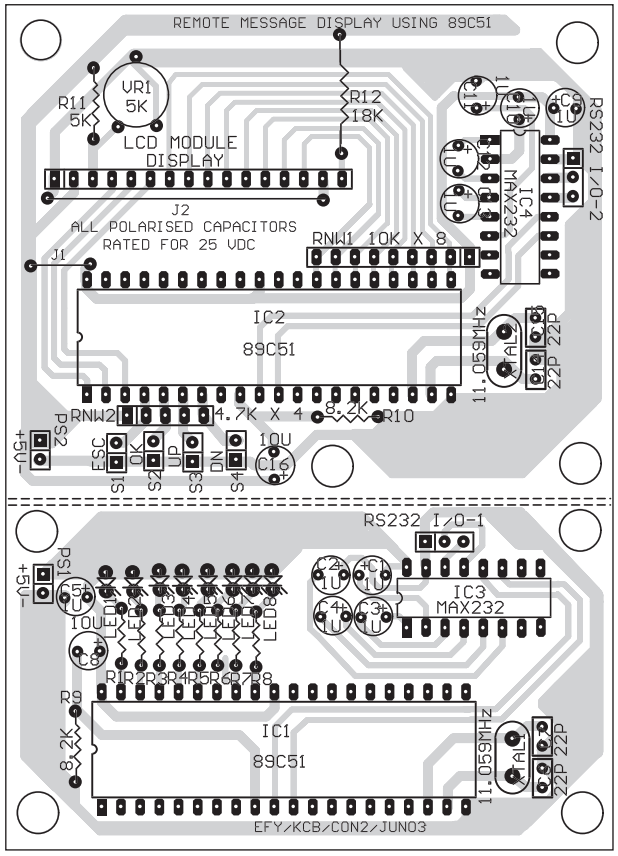


Fig. 5: Component layout for the PCB shown in Fig. 4

wi subroutine. This subroutine is used for transferring control instructions to the LCD. It first sets up the LCD for writing instructions (rw=0, en=0, rs=0) and then moves the data to Port 2 (P2.0 through P2.7) from the accumulator. It then reads the busy bit at the rdbusy subroutine and waits until the writing process is completed, and finally returns to the main program.

wd subroutine. This subroutine is used for transferring data to the LCD. It first sets the LCD for writing data (rw=0, en=0, rs=1) and moves data to Port 2 from the accumulator. It then reads the busy bit by the rdbusy subroutine and waits until the writing process is completed, and finally returns to the main program.

rdbusy subroutine. This subroutine is used for testing the busy bit during the writing operation to the LCD. It first selects the read set-up for the LCD (rw=1, en=0, rs=0). Then it sets Port 2-bit 7 (P2.7) and waits until this bit becomes low after successfully writing to the LCD. Finally, it returns to the main program.

del1m to del100m subroutines. These are just time delay subroutines.

add_day subroutine. The accumulator is set as a pointer for the control unit where the day information is stored. The send_con subrou-tine gets the data from the address pointer of the control unit. This data is now directly available in the two-digit ASCII format for the tens and units digits of day. The tens and units digits of the day are stored and then displayed at LCD locations C7H and C8H, resp-ectively. The add_ month and add_year subroutines are similar to the add_day subroutine.

keyprs? subrou-tine. This subroutine checks which key (Esc, Ok, Up, or Down) has been pressed. If no key is pressed, the subrou-tine returns with the accumulator containing FFH. Key switches are connected to Port 1 (P1.3 through P1.6). Pins P1.3 through P1.6 usually remain high until a key is pressed.

If any key is sensed low, the program jumps to confirm whether it was an unintentional low or it really happ-ened by keypress. For confirming so, the program waits for the bounce period of 10 milliseconds and then checks for the low again on the same key.

If the key is not sensed low now, it is assumed to be an accidental low and the subroutine returns as if no key was pressed.

But if the key is sensed low for the second time also, the program accepts the key and waits for the user to release the key in about 300 miliseconds. After 300 miliseconds, even if the user does not release the key, the program repeats the action as if the key is being pressed again and again. The program control returns with a code in the accumulator.

Codes for the keys are:

- '01' for pressing Esc key
- '02' for pressing Ok key
- '03' for pressing Up key
- '04' for pressing Down key

trfr_day subroutine. This subroutine transfers the day data to the appropriate location in the control unit. When this subroutine is called, the data is available as two digits (tens and units) in the ASCII format. As the data needs to be stored at one RAM location in the hex format, the program has to convert the two ASCII digits into a single hex digit by the asci_hex subroutine. At the end of the asci_hex subroutine, an equivalent hex number is available as hex variable.

The program now starts sending the characters. First, start code 02H is sent to the control unit, signaling it to get ready as the data is coming. Second, the address 30H is sent, where the day data is to be stored. Finally, the hex variable is sent, which is the current day data. The trfr_month and trfr_year are similar subroutines.

Only the address where the data is to be stored is different in each case.

hex_asci subroutine. First the units and tens digits are reset to ASCII zero. Then check whether the hex number is already zero. If yes, simply return. Else, advance the units. If the units digit has crossed ASCII 9, we need to reset the units digit to zero and advance the tens digit. Simultaneously, the hex number has to be decremented. The process keeps repeating until hex number becomes zero. The accumulated tens and units are equivalent to the hex number originally loaded.

asci_hex subroutine. Here the process is almost opposite to what we did while converting hex into ASCII. First, the hex number is reset to zero. Then we check whether both the units and tens digits are zero. If so, we simply return. Otherwise, we have to advance the hex number. Simultaneously, the units and tens digits are to be decremented. The process keeps repeating itself until units and tens digits become zero.

adv_day subroutine. This subroutine advances the day data, but ensures that it does not go beyond 31. The first part checks whether the day's units digit is 1 (decimal) and tens digit is 3 (decimal). If so, the program sets the units digit to 1 and the tens digit to 0 before returning.

The second part of the subroutine advances the day's units digit until it crosses 9 (ASCII 39). After 9, the day's units digit is reset to 0 and the tens digit is advanced. Similarly, if the tens digit crosses 9, the program sets it to 0.

dec_day subroutine. This subroutine decrements the day value. The first part checks whether the day's units digit is 1 (decimal) and the tens digit is 0 (decimal). If so, the program sets the tens digit to 3 before returning.

In the second part, as the day's units digit is decremented, the program tests whether it has gone below zero. (When ASCII 30h decrements, it will become ASCII 2fh.) The program sets the units digit to 9 (ASCII 39h) and decrements the tens digit. As the tens digit is decremented, the program tests whether it has gone below zero. If so, the program sets the tens digit to 9.

The adv_month and adv_year subroutines are similar to the adv_day subroutine, and the dec_month and dec_year subroutines are similar to the dec_day subroutine.

send_con subroutine. This subroutine first sends the address to the control unit and waits for the interrupt flag to go high. This means the data from the control unit is to be received at the specified address. After receiving the data, the interrupt flag gets cleared for the next instruction. The data is received from register B, saved as the hex variable, and converted into the ASCII code that is required for the LCD module.

The actual-size, single-side PCB layout for the handheld unit comprising LCD module and control unit is shown in Fig. 4 and its component layout in Fig. 5.

The combined PCB can be cut along the dotted lines to separate the control unit and handheld unit comprising LCD module.

Note. The software program of module.lst and contr.lst are included in the CD.

LCD DRIVER UNIT (MODULE.LST)

```
PAGE 1
 1            $mod51
 2            ;program for 'LCD' module
 3            ;article written for EFY
 4            ;programmer-AR Karkare
 5            ;Date last modified- 5th May 2003
 6            ;uses 89c51 micro-controller with 11.059 mhz crystal
 7            ;CU means Control Unit
 8            ;modifications included.
 9            ;1.incrementing/decrementing of day/month/year in continuous step(not in
single step).
 10           ;2.while advancing, limiting day/month/year to 30,12 and 99.
 11           ;3.while decrementing, resetting the day to 30,month to 12,year to 99 after
01.
 12           ;file name - module.asm
 13           ;RESERVED LOCATIONS
 0030           14    day_t   equ   30h        ;day tens
 0031           15    day_u   equ   31h        ;day units
 0032           16    mon_t   equ   32h        ;month tens
 0033           17    mon_u   equ   33h        ;month units
 0034           18    yer_t   equ   34h        ;year tens
 0035           19    yer_u   equ   35h        ;year units
 0036           20    units   equ   36h        ;temp storage of units
 0037           21    tens    equ   37h        ;temp storage of tens
 0038           22    hex     equ   38h        ;temp storage of hex byte
                23                             ;LIST OF I/O
 0045           24    intflg  bit   45h        ;interrupt flag
                25                             ;FOR LCD MODULE
 0090           26    rs      equ   p1.0       ;0=instructions,1=data
 0091           27    rw      equ   p1.1       ;0=write,1=read
 0092           28    en      equ   p1.2       ;0=disabled,1=enabled
                29                             ;p2 port for data from cpu to
                                               LCD module
                30                             ;FOR KEYBOARD
 0093           31    esc     equ   p1.3       ;Escape key
 0094           32    ok      equ   p1.4       ;Ok key
 0095           33    up      equ   p1.5       ;Plus key
 0096           34    dn      equ   p1.6       ;Minus key
                35                             ;FOR RS232 COMMUNICATION
 0000           36    org     0000h            ;
 0000 802E      37    sjmp    start            ;jump to main program
 0023           38    org     0023h            ;
 0023 21A0      39    ajmp    spint            ;jump to serial port int. program
 0030           40    org     0030h            ;initialization of registers
 0030 758160    41    start:  mov   sp, #60h   ;set stack pointer
                42                             ;initialise SFRs
 0033 758700    43    mov     pcon, #00h       ;smod=0
 0036 758920    44    mov     tmod, #20h       ;timer1(gate=0,c/t=0,mode=8  bit
                                                auto reload)
 0039 758BFD    45    mov     tl1, #0fdh       ;reload value for 9.6k baud rate
 003C 758DFD    46    mov     th1, #0fdh       ;
 003F 759850    47    mov     scon, #50h       ;mode 1,reception enabled
 0042 D28E      48    setb    tr1              ;start timer
 0044 D2AF      49    setb    ea               ;global int. on
 0046 D2AC      50    setb    es               ;serial int. on
 0048 D2BC      51    setb    ip.4             ;high priority to serial port int.
 004A D2B0      52    setb    rxd              ;float pin
 004C D2B1      53    setb    txd              ;float pin
 004E C245      54    clr     intflg           ;clear interrupt flag
                55                             ;initialise LCD module
 0050 120245    56    lcall   del10m           ;power on delay
 0053 120245    57    lcall   del10m           ;
 0056 7430      58    mov     a, #30h          ;select in write mode-three times
 0058 1201F5    59    lcall   wi               ;write instruction
 005B 120238    60    lcall   del4m            ;
 005E 7430      61    mov     a, #30h          ;select in write mode-three times
 0060 1201F5    62    lcall   wi               ;write instruction
 0063 120238    63    lcall   del4m            ;
 0066 7430      64    mov     a, #30h          ;select in write mode-three times
 0068 1201F5    65    lcall   wi               ;write instruction
 006B 7438      66    mov     a, #38h          ;select 8-bit operation,
                67                             ;number of lines 2,font 5x7 dots
 006D 1201F5    68    lcall   wi               ;
 0070 740C      69    mov     a, #00001100b    ;display on,cursor off,blinking off
 0072 1201F5    70    lcall   wi               ;
 0075 7401      71    mov     a, #01h          ;clear display,cursor to home
                                                position
 0077 1201F5    72    lcall   wi               ;
 007A 7406      73    mov     a, #06h          ;set incrementing dd ram, no shift
                                                mode
 007C 1201F5    74    lcall   wi               ;
 007F 9004C5    75    step1:  mov   dptr, #scr1 ;set pointer
 0082 1201C3    76    lcall   setup            ;display "Welcome to EFY Article"
 0085 1202C5    77    key1:   lcall   keyprs?  ;key pressed?
 0088 B402FA    78    cjne    a, #02h,key1     ;keep waiting till 'ok' key pressed
 008B 8000      79    sjmp    step2            ;if ok key-go to step 2
 008D 900505    80    step2 : mov   dptr, #scr2 ;set pointer
 0090 1201C3    81    lcall   setup            ;display "Birthday/day/month/y
                                                year"
 0093 74C0      82    mov     a, #0c0h         ;set address line2/col1
```

```
0095 1201F5  83   lcall   wi                  ;write address on LCD module
0098 743E    84   mov     a, #3eh;set arrow pointer
009A 120208  85   lcall   wd                  ;write data on LCD module
009D 12025C  86   lcall   add_day             ;add day
00A0 12027F  87   lcall   add_month           ;add month
00A3 1202A2  88   lcall   add_year            ;add year
00A6 1202C5  89   key2:   lcall   keyprs?     ;key pressed?
00A9 B40102  90   cjne    a, #01h,_022?       ;
00AC 80D1    91   sjmp    step1               ;if escape key-go back to step 1
00AE B40202  92   _022?:  cjne    a, #02h,_042? ;
00B1 8060    93   sjmp    step5               ;if ok key-go to step 5
00B3 B404F0  94   _042?:  cjne    a, #04h,key2  ;if none-go back to key2
00B6 8000    95   sjmp    step3               ;if down key-go to step 3
00B8 900505  96   step3:  mov     dptr, #scr2   ;set pointer
00BB 1201C3  97   lcall   setup               ;display "Birthday/day/month/
                                               year"
00BE 7490    98   mov     a, #90h             ;set address line3/col1
00C0 1201F5  99   lcall   wi                  ;
00C3 743E    100  mov     a, #3eh;set arrow pointer
00C5 120208  101  lcall   wd                  ;
00C8 12025C  102  lcall   add_day             ;add day
00CB 12027F  103  lcall   add_month           ;add month
00CE 1202A2  104  lcall   add_year            ;add year
00D1 1202C5  105  key3:   lcall   keyprs?     ;key pressed?
00D4 B40102  106  cjne    a, #01h,_023?       ;
00D7 80A6    107  sjmp    step1               ;if escape key-go back to step 1
00D9 B40202  108  _023?:  cjne    a, #02h,_033? ;
00DC 8064    109  sjmp    step6               ;if ok key-go to step6
00DE B40302  110  _033?:  cjne    a, #03h,_043?  ;
00E1 80AA    111  sjmp    step2               ;if up key-go to step2
00E3 B404EB  112  _043?:  cjne    a, #04h,key3 ;if none-go back to key3
00E6 8000    113  sjmp    step4               ;if down key-go back to step4
00E8 900505  114  step4:  mov     dptr, #scr2   ;set pointer
00EB 1201C3  115  lcall   setup               ;display "Birthday/day/month/
                                               year"
00EE 74D0    116  mov     a, #0d0h            ;add arrow pointer
00F0 1201F5  117  lcall   wi                  ;
00F3 743E    118  mov     a, #3eh;
00F5 120208  119  lcall   wd                  ;
00F8 12025C  120  lcall   add_day             ;add day
00FB 12027F  121  lcall   add_month           ;add month
00FE 1202A2  122  lcall   add_year            ;add year
0101 1202C5  123  key4:   lcall   keyprs?     ;key pressed?
0104 B40102  124  cjne    a, #01h,_024?       ;
0107 017F    125  ajmp    step1               ;if escape key-go back to step 1
0109 B40202  126  _024?:  cjne    a, #02h,_034? ;
010C 8063    127  sjmp    step7               ;if ok key-go to step7
010E B403F0  128  _034?:  cjne    a, #03h,key4 ;if none-go back to key4
0111 80A5    129  sjmp    step3               ;if up key-go back to step3
0113 900545  130  step5:  mov     dptr, #scr5   ;set pointer
0116 1201C3  131  lcall   setup               ;display "set/day"
0119 12025C  132  lcall   add_day             ;add day
011C 1202C5  133  key5:   lcall   keyprs?     ;key pressed?
011F B40102  134  cjne    a, #01h,_025?       ;
0122 018D    135  ajmp    step2               ;if escape key-go to step2
0124 B40205  136  _025?:  cjne    a, #02h,_035? ;
0127 12032C  137  lcall   trfr_day            ;if ok key-transfer day value
012A 018D    138  ajmp    step2               ;back to display
012C B40308  139  _035?:  cjne    a, #03h,_045? ;
012F 1203D7  140  lcall   adv_day             ;if up key-advance day value
0132 12026A  141  lcall   disp_day            ;display new day
0135 80E5    142  sjmp    key5                ;back
0137 B404E2  143  _045?:  cjne    a, #04h,key5 ;none-back to key5
013A 1203FB  144  lcall   dec_day             ;if down key-decrement day value
013D 12026A  145  lcall   disp_day            ;display new day
0140 80DA    146  sjmp    key5                ;back
0142 900585  147  step6:  mov     dptr, #scr6   ;set pointer
0145 1201C3  148  lcall   setup               ;display "set/month"
0148 12027F  149  lcall   add_month           ;add month
014B 1202C5  150  key6:   lcall   keyprs?     ;key pressed?
014E B40102  151  cjne    a, #01h,_026?       ;
0151 01B8    152  ajmp    step3               ;if escape key-go to step3
0153 B40205  153  _026?:  cjne    a, #02h,_036? ;
0156 120348  154  lcall   trfr_month           ;if ok key-transfer month value
0159 018D    155  ajmp    step2               ;back to display
015B B40308  156  _036?:  cjne    a, #03h,_046? ;
015E  12041F 157  lcall   adv_month           ;if up key-advance month value
0161 12028D  158  lcall   disp_mnth          ;display new month
0164 80E5    159  sjmp    key6                ;back
0166 B404E2  160  _046?:  cjne    a, #04h,key6 ;none-back to key6
0169 120449  161  lcall   dec_month           ;if down key-decrement month
                                               value
016C 12028D  162  lcall   disp_mnth          ;display new month
016F 80DA    163  sjmp    key6                ;back
0171 9005C5  164  step7:  mov     dptr,#scr7    ;set pointer
0174 1201C3  165  lcall   setup               ;display "set/year"
0177 1202A2  166  lcall   add_year            ;add year
017A 1202C5  167  key7:   lcall   keyprs?     ;key pressed?
017D B40102  168  cjne    a, #01h,_027?       ;
0180 01E8    169  ajmp    step4               ;if escape key-go to step4
0182 B40205  170  _027?:  cjne    a, #02h,_037? ;
0185 120364  171  lcall   trfr_year           ;if ok key-transfer year value
0188 018D    172  ajmp    step2               ;back to display
018A B40308  173  _037?:  cjne    a, #03h,_047? ;
018D 12046D  174  lcall   adv_year           ;if up key-advance year value
0190 1202B0  175  lcall   disp_year          ;display new year
0193 80E5    176  sjmp    key7                ;back
0195 B404E2  177  _047?:  cjne    a, #04h,key7 ;none-back to key7
0198 120491  178  lcall   dec_year           ;if down key-decrement year value
019B 1202B0  179  lcall   disp_year          ;display new year
019E 80DA    180  sjmp    key7                ;back
             181                              ;SUBROUTINES
01A0 C2AF    182  spint:  clr     ea          ;disable all interrupts
01A2 C0D0    183  push    psw                 ;
01A4 C0E0    184  push    acc                 ;
01A6 C3      185  clr     c                   ;
01A7 8599F0  186  mov     b, sbuf             ;save in reg. B
01AA C298    187  out:    clr     ri          ;
01AC D0E0    188  pop     acc                 ;
01AE D0D0    189  pop     psw                 ;
01B0 D2AF    190  setb    ea                  ;enable all interrupts
01B2 D245    191  setb    intflg              ;set interrupt flag
01B4 32      192  reti                        ;return from interrupt
01B5 C2AF    193  send:   clr     ea          ;disable all interrupts
01B7 C299    194  clr     ti                  ;pull ti flag low
01B9 F599    195  mov     sbuf, a             ;load sbuf
01BB 3099FD  196  waitt:  jnb     ti, waitt   ;wait till ti flag goes high
01BE C299    197  clr     ti                  ;pull ti flag low
01C0 D2AF    198  setb    ea                  ;enable all interrupts
01C2 22      199  ret                         ;return
01C3 7A80    200  setup:  mov     r2, #80h    ;line 1-column 0 position
01C5 EA      201  mov     a, r2               ;
01C6 1201F5  202  lcall   wi                  ;write instruction
01C9         203  display:
01C9 E4      204  clr     a                   ;
01CA 93      205  movc    a, @a+dptr          ;
01CB 120208  206  lcall   wd                  ;write data
01CE A3      207  inc     dptr                ;
01CF 0A      208  inc     r2                  ;
01D0 BA9008  209  cjne    r2, #090h,_2ndln    ;
01D3 7AC0    210  mov     r2, #0c0h           ;
01D5 EA      211  mov     a, r2               ;
01D6 1201F5  212  lcall   wi                  ;
01D9 80EE    213  sjmp    display;
01DB BAD008  214  _2ndln:  cjne    r2, #0d0h,_3rdln ;
01DE 7A90    215  mov     r2, #090h           ;
01E0 EA      216  mov     a, r2               ;
01E1 1201F5  217  lcall   wi                  ;
01E4 80E3    218  sjmp    display;
01E6 BAA008  219  _3rdln:  cjne    r2, #0a0h,_4thln ;
01E9 7AD0    220  mov     r2, #0d0h           ;
01EB EA      221  mov     a, r2               ;
01EC 1201F5  222  lcall   wi                  ;
01EF 80D8    223  sjmp    display;
01F1 BAE0D5  224  _4thln:  cjne    r2, #0e0h,display ;
01F4 22      225  ret                         ;
01F5 C291    226  wi:     clr     rw          ;select write
01F7 C292    227  clr     en                  ;transfer disabled
01F9 C290    228  clr     rs                  ;select instruction
01FB F5A0    229  mov     p2, a               ;set data to port
01FD D292    230  setb    en                  ;transfer disabled
01FF 12022B  231  lcall   del1m               ;
0202 C292    232  clr     en                  ;
0204 12021B  233  lcall   rdbusy              ;test busy flag
0207 22      234  ret                         ;return
0208 C291    235  wd:     clr     rw          ;select write
020A C292    236  clr     en                  ;transfer disabled
020C D290    237  setb    rs                  ;select data
020E F5A0    238  mov     p2, a               ;set data to port
0210 D292    239  setb    en                  ;
0212 12022B  240  lcall   del1m               ;
0215 C292    241  clr     en                  ;transfer disabled
0217 12021B  242  lcall   rdbusy              ;
021A 22      243  ret                         ;return
021B C292    244  rdbusy:  clr     en          ;
021D C290    245  clr     rs                  ;
021F D291    246  setb    rw                  ;
0221 D2A7    247  setb    p2.7                ;
0223 D292    248  setb    en                  ;
0225 20A7FD  249  wt:     jb      p2.7, wt    ;
0228 C292    250  clr     en                  ;
022A 22      251  ret                         ;
022B 755004  252  del1m:  mov     50h, #04h   ;delay of 1 milisec.
022E 755153  253  loopa:  mov     51h ,#53h   ;
0231 D551FD  254  loopb:  djnz    51h, loopb  ;
0234 D550F7  255  djnz    50h, loopa          ;
0237 22      256  ret                         ;
0238 12022B  257  del4m:  lcall   del1m       ;delay of 4 milisec.
```

```
023B 12022B  258  lcall   del1m              ;
023E 12022B  259  lcall   del1m              ;
0241 12022B  260  lcall   del1m              ;
0244 22      261  ret                        ;
0245 755028  262  del10m: mov    50h, #28h   ;delay of 10 milisec.
0248 755153  263  loopc:  mov    51h, #53h   ;
024B D551FD  264  loopd:  djnz   51h, loopd  ;
024E D550F7  265  djnz    50h, loopc        ;
0251 22      266  ret                        ;
0252         267  del100m:
0252 75560A  268  mov     56h, #0ah          ;delay of 100 milisec.
0255 120245  269  looph:  lcall   del10m     ;
0258 D556FA  270  djnz    56h, looph        ;
025B 22      271  ret                        ;
025C         272  add_day:
025C 120245  273  lcall   del10m             ;wait for stabilisation
025F 7430    274  mov     a, #30h            ;Day value is stored at 30h in CU
0261 1204B5  275  lcall   send_con           ;send and convert
0264 853730  276  mov     day_t,tens         ;save new tens
0267 853631  277  mov     day_u,units        ;save new units
026A         278  disp_day:
026A 74C7    279  mov     a, #0c7h ;set address in LCD module(line2/
                                             col8)
026C 1201F5  280  lcall   wi                 ;
026F E537    281  mov     a, tens            ;print tens
0271 120208  282  lcall   wd                 ;
0274 74C8    283  mov     a, #0c8h           ;set address in LCD module
0276 1201F5  284  lcall   wi                 ;
0279 E536    285  mov     a, units           ;print units
027B 120208  286  lcall   wd                 ;
027E 22      287  ret                        ;
027F         288  add_month:
027F 120245  289  lcall   del10m             ;wait for stabilisation
0282 7431    290  mov     a, #31h            ;month value is stored at 31h in
                                             CU
0284 1204B5  291  lcall   send_con           ;send and convert
0287 853732  292  mov     mon_t, tens        ;save new tens
028A 853633  293  mov     mon_u, units       ;save new units
028D         294  disp_mnth:
028D 7497    295  mov     a, #97h            ;set address in LCD module(line3/
                                             col8)
028F 1201F5  296  lcall   wi                 ;
0292 E537    297  mov     a, tens            ;print tens
0294 120208  298  lcall   wd                 ;
0297 7498    299  mov     a, #98h            ;set address in LCD module
0299 1201F5  300  lcall   wi                 ;
029C E536    301  mov     a, units           ;print units
029E 120208  302  lcall   wd                 ;
02A1 22      303  ret                        ;
02A2         304  add_year:
02A2 120245  305  lcall   del10m             ;wait for stabilisation
02A5 7432    306  mov     a, #32h            ;year value is stored at 32h in CU
02A7 1204B5  307  lcall   send_con           ;send and convert
02AA 853734  308  mov     yer_t, tens        ;save new tens
02AD 853635  309  mov     yer_u, units       ;save new units
02B0         310  disp_year:
02B0 74D7    311  mov     a, #0d7h           ;set address in LCD module(line4/
                                             col8)
02B2 1201F5  312  lcall   wi                 ;
02B5 E537    313  mov     a, tens            ;print tens
02B7 120208  314  lcall   wd                 ;
02BA 74D8    315  mov     a, #0d8h           ;set address in LCD module
02BC 1201F5  316  lcall   wi                 ;
02BF E536    317  mov     a, units           ;print units
02C1 120208  318  lcall   wd                 ;
02C4 22      319  ret                        ;
02C5         320  keyprs?:
02C5 D293    321  setb    esc                ;set escape pin high
02C7 D294    322  setb    ok                 ;set ok pin high
02C9 D295    323  setb    up                 ;set up pin high
02CB D296    324  setb    dn                 ;set down pin high
02CD 30930C  325  jnb     esc, con_esc       ;escape key low-confirm
02D0 30941D  326  jnb     ok, con_ok         ;ok key low-confirm
02D3 30952E  327  jnb     up, con_up         ;up key low-confirm
02D6 30963F  328  jnb     dn, con_dn         ;down key low-confirm
02D9 74FF    329  retmt:  mov     a, #0ffh   ;no key-make accu. all high
02DB 22      330  ret                        ;return empty
02DC         331  con_esc:
02DC 120245  332  lcall   del10m             ;debounce delay
02DF 309302  333  jnb     esc, accesc        ;escape key low-accept escape
02E2 80F5    334  sjmp    retmt              ;otherwise return empty
02E4 120252  335  accesc: lcall   del100m    ;wait for 300 milisec
02E7 120252  336  lcall   del100m           ;
02EA 120252  337  lcall   del100m           ;
02ED 7401    338  mov     a, #01h            ;set accu. to 01
02EF 22      339  ret                        ;return
02F0 120245  340  con_ok: lcall   del10m     ;debounce delay
02F3 309402  341  jnb     ok, accok          ;ok key low-accept ok
02F6 80E1    342  sjmp    retmt              ;otherwise return empty
02F8 120252  343  accok:  lcall   del100m    ;wait for 300 milisec
02FB 120252  344  lcall   del100m           ;
02FE 120252  345  lcall   del100m           ;
0301 7402    346  mov     a, #02h            ;set accu. to 02
0303 22      347  ret                        ;return
0304 120245  348  con_up: lcall   del10m     ;debounce delay
0307 309502  349  jnb     up, accup          ;up key low-accept up
030A 80CD    350  sjmp    retmt              ;otherwise return empty
030C 120252  351  accup:  lcall   del100m    ;wait for 300 milisec
030F 120252  352  lcall   del100m           ;
0312 120252  353  lcall   del100m           ;
0315 7403    354  mov     a, #03h            ;set accu. to 03
0317 22      355  ret                        ;return
0318 120245  356  con_dn: lcall   del10m     ;debounce delay
031B 309602  357  jnb     dn, accdn          ;down key low-accept down
031E 80B9    358  sjmp    retmt              ;otherwise return empty
0320 120252  359  accdn:  lcall   del100m    ;wait for 300 milisec
0323 120252  360  lcall   del100m           ;
0326 120252  361  lcall   del100m           ;
0329 7404    362  mov     a, #04h            ;set accu. to 04
032B 22      363  ret                        ;return
032C         364  trfr_day:
032C 1203A8  365  lcall   asci_hex          ;convert ascii characters to hex
032F 120245  366  lcall   del10m            ;wait for stabilisation
0332 7402    367  mov     a, #02h            ;load start character <STX>
0334 1201B5  368  lcall   send              ;send to CU
0337 120245  369  lcall   del10m            ;wait for stabilisation
033A 7430    370  mov     a, #30h            ;address in CU where to store the
                                             data
033C 1201B5  371  lcall   send              ;send address
033F 120245  372  lcall   del10m            ;wait for stabilisation
0342 E538    373  mov     a, hex             ;get day value-hex format
0344 1201B5  374  lcall   send              ;send day value
0347 22      375  ret                        ;return
0348         376  trfr_month:
0348 1203A8  377  lcall   asci_hex          ;convert ascii characters to hex
034B 120245  378  lcall   del10m            ;wait for stabilisation
034E 7402    379  mov     a, #02h            ;load start character <STX>
0350 1201B5  380  lcall   send              ;send to CU
0353 120245  381  lcall   del10m            ;wait for stabilisation
0356 7431    382  mov     a, #31h            ;address in CU where to store the d
                                             data
0358 1201B5  383  lcall   send              ;send address
035B 120245  384  lcall   del10m            ;wait for stabilisation
035E E538    385  mov     a, hex             ;get month value in hex format
0360 1201B5  386  lcall   send              ;send month value
0363 22      387  ret                        ;return
0364         388  trfr_year:
0364 1203A8  389  lcall   asci_hex          ;convert ascii characters to hex
0367 120245  390  lcall   del10m            ;wait for stabilisation
036A 7402    391  mov     a, #02h            ;load start character <STX>
036C 1201B5  392  lcall   send              ;send to CU
036F 120245  393  lcall   del10m            ;wait for stabilisation
0372 7432    394  mov     a, #32h            ;address in CU where to store the
                                             data
0374 1201B5  395  lcall   send              ;send address
0377 120245  396  lcall   del10m            ;wait for stabilisation
037A E538    397  mov     a, hex             ;get year value in hex format
037C 1201B5  398  lcall   send              ;send year value
037F 22      399  ret                        ;return
0380         400  hex_asci:
0380 C3      401  clr     c                  ;clear carry
0381 E4      402  clr     a                  ;clear accu.
0382 753630  403  mov     units, #30h        ;ascii zero in units
0385 753730  404  mov     tens, #30h         ;ascii zero in tens
0388 E538    405  zero?:  mov     a, hex     ;get hex value
038A B40001  406  cjne    a, #00h,adv_units  ;
038D 22      407  ret                        ;if hex value is zero-return
038E         408  adv_units:
038E 0536    409  inc     units              ;advance units
0390 E536    410  mov     a, units           ;get new units
0392 B43A0F  411  cjne    a, #3ah,dcr_hex    ;if units exceeding ascii 39
0395 753630  412  mov     units, #30h        ;set zero in units
0398 0537    413  inc     tens               ;advance tens
039A E537    414  mov     a, tens            ;get new tens
039C B43A05  415  cjne    a, #3ah,dcr_hex    ;if tens exceeding ascii 39
039F 753730  416  mov     tens, #30h         ;set zero in tens
03A2 80FE    417  sjmp    $                  ;this is illegal stage-so hang up
03A4         418  dcr_hex:
03A4 1538    419  dec     hex                ;decrement hex value
03A6 80E0    420  sjmp    zero?              ;check is it zero yet?
             421                             ;this subroutine converts ascii
                                             characters
             422                             ;to equivalent hex format
03A8         423  asci_hex:
03A8 C3      424  clr     c                  ;clear carry for safety
03A9 E4      425  clr     a                  ;clear accu.
03AA 753800  426  mov     hex, #00h          ;clear destination hex register
             427                             ;remember ascii zero=30h
```

```
03AD E536  428  allzro:  mov    a, units        ;gets units
03AF B43006 429  cjne     a, #30h,adv_hex ;is it zero?
03B2 E537  430  mov      a, tens         ;get tens
03B4 B43001 431  cjne     a, #30h,adv_hex ;is it zero?
03B7 22    432  ret                      ;all zero-then return
03B8       433  adv_hex:
03B8 0538  434  inc      hex             ;advance hex count
03BA E538  435  mov      a, hex          ;get hex count
03BC B40002 436  cjne     a, #00h,dcr_asc ;crossed ff?
03BF 80FE  437  sjmp     $               ;this is illegal-so hang up
03C1       438  dcr_asc:
03C1 1536  439  dec      units           ;decrement units
03C3 E536  440  mov      a, units        ;get units
03C5 B42FE5 441  cjne     a, #2fh,allzro  ;crossed zero?
03C8 753639 442  mov      units, #39h     ;if crossed-set units to 9(ascii39)
03CB 1537  443  dec      tens            ;decrement tens
03CD E537  444  mov      a, tens         ;get tens
03CF B42FDB 445  cjne     a, #2fh,allzro  ;crossed zero?
03D2 753739 446  mov      tens, #39h      ;set tens to 9-illegal state
03D5 80FE  447  sjmp     $               ;hang operation
03D7       448  adv_day:
03D7 E531  449  mov      a, day_u        ;get current day units
03D9 B4310C 450  cjne     a, #31h,cont_ad ;test-is it ascii one?
03DC E530  451  mov      a, day_t        ;yes-get current day tens, no-
                                         continue advnacing day
03DE B43307 452  cjne     a, #33h,cont_ad ;test-is it ascii three?
03E1 753631 453  mov      units, #31h     ;yes-set units-ascii one
03E4 753730 454  mov      tens, #30h      ;set tens-ascii zero
03E7 22    455  ret                      ;return
03E8       456  cont_ad:
03E8 E531  457  mov      a, day_u        ;continue advancing day
03EA 04    458  inc      a               ;
03EB B43A55 459  cjne     a, #3ah,save_units ;
03EE 753630 460  mov      units, #30h     ;
03F1 E530  461  mov      a, day_t        ;
03F3 04    462  inc      a               ;
03F4 B43A4F 463  cjne     a, #3ah,save_tens ;
03F7 753730 464  mov      tens, #30h      ;
03FA 22    465  ret                      ;
03FB       466  dec_day:
03FB E531  467  mov      a, day_u        ;get current day units
03FD B4310C 468  cjne     a, #31h,cont_dd ;test-is it ascii one?
0400 E530  469  mov      a, day_t        ;yes-get current day tens,no-
                                         continue decrementing day
0402 B43007 470  cjne     a, #30h,cont_dd ;test-is it ascii zero?
0405 753631 471  mov      units, #31h     ;yes-set units-ascii one, no-continue
                                         decrementing day;;;;
0408 753733 472  mov      tens, #33h      ;set tens-ascii three
040B 22    473  ret                      ;return
040C       474  cont_dd:
040C E531  475  mov      a, day_u        ;continue decrementing day
040E 14    476  dec      a               ;
040F B42F31 477  cjne     a, #2fh,save_units ;
0412 753639 478  mov      units, #39h     ;
0415 E530  479  mov      a, day_t        ;
0417 14    480  dec      a               ;
0418 B42F2B 481  cjne     a, #2fh, save_tens ;
041B 753739 482  mov      tens, #39h      ;
041E 22    483  ret                      ;
041F       484  adv_month:
041F E533  485  mov      a, mon_u        ;get current month units
0421 B4320C 486  cjne     a, #32h, cont_am ;test-is it ascii two?
0424 E532  487  mov      a, mon_t        ;yes-get current month tens,no-
                                         continue advancing month
0426 B43107 488  cjne     a, #31h, cont_am ;test-is it ascii one?
0429 753631 489  mov      units, #31h     ;yes-set units-ascii one
042C 753730 490  mov      tens, #30h      ;set tens-ascii zero
042F 22    491  ret                      ;return
0430       492  cont_am:
0430 E533  493  mov      a, mon_u        ;continue advancing month
0432 04    494  inc      a               ;
0433 B43A0D 495  cjne     a, #3ah, save_units ;
0436 753630 496  mov      units, #30h     ;
0439 E532  497  mov      a, mon_t        ;
043B 04    498  inc      a               ;
043C B43A07 499  cjne     a, #3ah,save_tens ;
043F 753730 500  mov      tens, #30h      ;
0442 22    501  ret                      ;
0443       502  save_units:
0443 F536  503  mov      units, a        ;
0445 22    504  ret                      ;
0446       505  save_tens:
0446 F537  506  mov      tens, a         ;
0448 22    507  ret                      ;
0449       508  dec_month:
0449 E533  509  mov      a, mon_u        ;get current month units
044B B4310C 510  cjne     a, #31h,cont_md ;test-is it ascii one?
044E E532  511  mov      a, mon_t        ;yes-get current month tens,no-
                                         continue decrementing month
0450 B43007 512  cjne     a, #30h,cont_md ;test-is it ascii zero?
0453 753632 513  mov      units, #32h     ;yes-set units-ascii two,no-continue
                                         decrementing month
0456 753731 514  mov      tens, #31h      ;set tens-ascii one
0459 22    515  ret                      ;return
045A       516  cont_md:
045A E533  517  mov      a, mon_u        ;continue decrementing month
045C 14    518  dec      a               ;
045D B42FE3 519  cjne     a, #2fh,save_units ;
0460 753639 520  mov      units, #39h     ;
0463 E532  521  mov      a, mon_t        ;
0465 14    522  dec      a               ;
0466 B42FDD 523  cjne     a, #2fh,save_tens ;
0469 753739 524  mov      tens, #39h      ;
046C 22    525  ret                      ;
046D       526  adv_year:
046D E535  527  mov      a, yer_u        ;get current year units
046F B4390C 528  cjne     a, #39h, cont_ay ;test-is it ascii nine?
0472 E534  529  mov      a, yer_t        ;yes-get current year tens, no-
                                         continue advancing year
0474 B43907 530  cjne     a, #39h, cont_ay ;test-is it ascii nine?
0477 753630 531  mov      units, #30h     ;yes-set units-ascii zero,no continue
                                         advancing year
047A 753730 532  mov      tens, #30h      ;set tens-ascii zero
047D 22    533  ret                      ;return
047E       534  cont_ay:
047E E535  535  mov      a, yer_u        ;continue advancing year
0480 04    536  inc      a               ;
0481 B43ABF 537  cjne     a, #3ah,save_units ;
0484 753630 538  mov      units, #30h     ;
0487 E534  539  mov      a,y er_t        ;
0489 04    540  inc      a               ;
048A B43AB9 541  cjne     a, #3ah,save_tens ;
048D 753730 542  mov      tens, #30h      ;
0490 22    543  ret                      ;
0491       544  dec_year:
0491 E535  545  mov      a, yer_u        ;get current year units
0493 B4300C 546  cjne     a, #30h, cont_yd ;test-is it ascii one?
0496 E534  547  mov      a, yer_t        ;yes-get current year tens, no-
                                         continue decrementing year
0498 B43007 548  cjne     a, #30h,cont_yd ;test-is it ascii zero?
049B 753639 549  mov      units, #39h     ;yes-set units-ascii nine, no-continue
                                         decrementing year
049E 753739 550  mov      tens, #39h      ;set tens-ascii nine
04A1 22    551  ret                      ;return
04A2       552  cont_yd:
04A2 E535  553  mov      a, yer_u        ;continue decrementing year
04A4 14    554  dec      a               ;
04A5 B42F9B 555  cjne     a, #2fh,save_units ;
04A8 753639 556  mov      units, #39h     ;
04AB E534  557  mov      a, yer_t        ;
04AD 14    558  dec      a               ;
04AE B42F95 559  cjne     a, #2fh, save_tens ;
04B1 753739 560  mov      tens, #39h      ;
04B4 22    561  ret                      ;
04B5       562  send_con:
04B5 1201B5 563  lcall    send            ;send request to CU
04B8 3045FD 564  wait:    jnb      intflg, wait ;wait till serial data comes in
04BB C245  565  clr      intflg          ;clear indicator flag
           566                           ;on serial inturrupt,
           567                           ;the 'spint' subroutine works
           568                        ;data received is stored in register b
04BD E5F0  569  mov      a, b            ;get what has been received
04BF F538  570  mov      hex, a          ;save hex value
           571                           ;LCD module needs acsii input
04C1 120380 572  lcall    hex_asci        ;so, convert received hex data to ascii
04C4 22    573  ret                      ;return
04C5 57    574  scr1:    db       'W'    ;
04C6 45    575  db       'E'            ;
04C7 4C    576  db       'L'            ;
04C8 43    577  db       'C'            ;
04C9 4F    578  db       'O'            ;
04CA 4D    579  db       'M'            ;
04CB 45    580  db       'E'            ;
04CC 20    581  db       ' '            ;
04CD 54    582  db       'T'            ;
04CE 4F    583  db       'O'            ;
04CF 20    584  db       ' '            ;
04D0 20    585  db       ' '            ;
04D1 20    586  db       ' '            ;
04D2 20    587  db       ' '            ;
04D3 20    588  db       ' '            ;
04D4 20    589  db       ' '            ;
04D5 45    590  db       'E'            ;
04D6 46    591  db       'F'            ;
04D7 59    592  db       'Y'            ;
04D8 20    593  db       ' '            ;
04D9 20    594  db       ' '            ;
04DA 20    595  db       ' '            ;
```

```
04DB 20     596   db      ' '             ;
04DC 20     597   db      ' '             ;
04DD 20     598   db      ' '             ;
04DE 20     599   db      ' '             ;
04DF 20     600   db      ' '             ;
04E0 20     601   db      ' '             ;
04E1 20     602   db      ' '             ;
04E2 20     603   db      ' '             ;
04E3 20     604   db      ' '             ;
04E4 20     605   db      ' '             ;
04E5 41     606   db      'A'             ;
04E6 52     607   db      'R'             ;
04E7 54     608   db      'T'             ;
04E8 49     609   db      'I'             ;
04E9 43     610   db      'C'             ;
04EA 4C     611   db      'L'             ;
04EB 45     612   db      'E'             ;
04EC 20     613   db      ' '             ;
04ED 20     614   db      ' '             ;
04EE 20     615   db      ' '             ;
04EF 20     616   db      ' '             ;
04F0 20     617   db      ' '             ;
04F1 20     618   db      ' '             ;
04F2 20     619   db      ' '             ;
04F3 20     620   db      ' '             ;
04F4 20     621   db      ' '             ;
04F5  20    622   db      ' '             ;
04F6 20     623   db      ' '             ;
04F7 20     624   db      ' '             ;
04F8 20     625   db      ' '             ;
04F9 20     626   db      ' '             ;
04FA  20    627   db      ' '             ;
04FB 20     628   db      ' '             ;
04FC 20     629   db      ' '             ;
04FD 20     630   db      ' '             ;
04FE 20     631   db      ' '             ;
04FF 20     632   db      ' '             ;
0500 20     633   db      ' '             ;
0501 20     634   db      ' '             ;
0502 20     635   db      ' '             ;
0503 20     636   db      ' '             ;
0504 20     637   db      ' '             ;
0505 42     638   scr2:   db      'B'     ;
0506 49     639   db      'I'             ;
0507 52     640   db      'R'             ;
0508 54     641   db      'T'             ;
0509 48     642   db      'H'             ;
050A 44     643   db      'D'             ;
050B 41     644   db      'A'             ;
050C 59     645   db      'Y'             ;
050D 20     646   db      ' '             ;
050E 20     647   db      ' '             ;
050F 20     648   db      ' '             ;
0510 20     649   db      ' '             ;
0511 20     650   db      ' '             ;
0512 20     651   db      ' '             ;
0513 20     652   db      ' '             ;
0514 20     653   db      ' '             ;
0515 20     654   db      ' '             ;
0516 44     655   db      'D'             ;
0517 41     656   db      'A'             ;
0518 59     657   db      'Y'             ;
0519 20     658   db      ' '             ;
051A 20     659   db      ' '             ;
051B 20     660   db      ' '             ;
051C 20     661   db      ' '             ;
051D 20     662   db      ' '             ;
051E 20     663   db      ' '             ;
051F 20     664   db      ' '             ;
0520 20     665   db      ' '             ;
0521 20     666   db      ' '             ;
0522 20     667   db      ' '             ;
0523 20     668   db      ' '             ;
0524 20     669   db      ' '             ;
0525 20     670   db      ' '             ;
0526 4D     671   db      'M'             ;
0527 4F     672   db      'O'             ;
0528 4E     673   db      'N'             ;
0529 54     674   db      'T'             ;
052A 48     675   db      'H'             ;
052B 20     676   db      ' '             ;
052C 20     677   db      ' '             ;
052D 20     678   db      ' '             ;
052E 20     679   db      ' '             ;
052F 20     680   db      ' '             ;
0530 20     681   db      ' '             ;
0531 20     682   db      ' '             ;
0532 20     683   db      ' '             ;
0533 20     684   db      ' '             ;
0534 20     685   db      ' '             ;
0535 20     686   db      ' '             ;
0536 59     687   db      'Y'             ;
0537 45     688   db      'E'             ;
0538 41     689   db      'A'             ;
0539 52     690   db      'R'             ;
053A 20     691   db      ' '             ;
053B 20     692   db      ' '             ;
053C 20     693   db      ' '             ;
053D 20     694   db      ' '             ;
053E 20     695   db      ' '             ;
053F 20     696   db      ' '             ;
0540 20     697   db      ' '             ;
0541 20     698   db      ' '             ;
0542 20     699   db      ' '             ;
0543 20     700   db      ' '             ;
0544 20     701   db      ' '             ;
0545 53     702   scr5:   db      'S'     ;
0546 45     703   db      'E'             ;
0547 54     704   db      'T'             ;
0548 20     705   db      ' '             ;
0549 20     706   db      ' '             ;
054A 20     707   db      ' '             ;
054B 20     708   db      ' '             ;
054C 20     709   db      ' '             ;
054D 20     710   db      ' '             ;
054E 20     711   db      ' '             ;
054F 20     712   db      ' '             ;
0550 20     713   db      ' '             ;
0551 20     714   db      ' '             ;
0552 20     715   db      ' '             ;
0553 20     716   db      ' '             ;
0554 20     717   db      ' '             ;
0555 20     718   db      ' '             ;
0556 44     719   db      'D'             ;
0557 41     720   db      'A'             ;
0558 59     721   db      'Y'             ;
0559 20     722   db      ' '             ;
055A 20     723   db      ' '             ;
055B 20     724   db      ' '             ;
055C 20     725   db      ' '             ;
055D 20     726   db      ' '             ;
055E 20     727   db      ' '             ;
055F 20     728   db      ' '             ;
0560 20     729   db      ' '             ;
0561 20     730   db      ' '             ;
0562 20     731   db      ' '             ;
0563 20     732   db      ' '             ;
0564 20     733   db      ' '             ;
0565 20     734   db      ' '             ;
0566 20     735   db      ' '             ;
0567 20     736   db      ' '             ;
0568 20     737   db      ' '             ;
0569 20     738   db      ' '             ;
056A 20     739   db      ' '             ;
056B 20     740   db      ' '             ;
056C 20     741   db      ' '             ;
056D 20     742   db      ' '             ;
056E 20     743   db      ' '             ;
056F 20     744   db      ' '             ;
0570 20     745   db      ' '             ;
0571 20     746   db      ' '             ;
0572 20     747   db      ' '             ;
0573 20     748   db      ' '             ;
0574 20     749   db      ' '             ;
0575 20     750   db      ' '             ;
0576 20     751   db      ' '             ;
0577 20     752   db      ' '             ;
0578 20     753   db      ' '             ;
0579 20     754   db      ' '             ;
057A 20     755   db      ' '             ;
057B 20     756   db      ' '             ;
057C 20     757   db      ' '             ;
057D 20     758   db      ' '             ;
057E 20     759   db      ' '             ;
057F 20     760   db      ' '             ;
0580 20     761   db      ' '             ;
0581 20     762   db      ' '             ;
0582 20     763   db      ' '             ;
0583 20     764   db      ' '             ;
0584 20     765   db      ' '             ;
0585 53     766   scr6:   db      'S'     ;
0586 45     767   db      'E'             ;
0587 54     768   db      'T'             ;
0588 20     769   db      ' '             ;
0589 20     770   db      ' '             ;
058A 20     771   db      ' '             ;
058B 20     772   db      ' '             ;
058C 20     773   db      ' '             ;
```

```
058D 20          774   db      ' '             ;
058E 20          775   db      ' '             ;
058F 20          776   db      ' '             ;
0590 20          777   db      ' '             ;
0591 20          778   db      ' '             ;
0592 20          779   db      ' '             ;
0593 20          780   db      ' '             ;
0594 20          781   db      ' '             ;
0595 20          782   db      ' '             ;
0596 20          783   db      ' '             ;
0597 20          784   db      ' '             ;
0598 20          785   db      ' '             ;
0599 20          786   db      ' '             ;
059A 20          787   db      ' '             ;
059B 20          788   db      ' '             ;
059C 20          789   db      ' '             ;
059D 20          790   db      ' '             ;
059E 20          791   db      ' '             ;
059F 20          792   db      ' '             ;
05A0 20          793   db      ' '             ;
05A1 20          794   db      ' '             ;
05A2 20          795   db      ' '             ;
05A3 20          796   db      ' '             ;
05A4 20          797   db      ' '             ;
05A5 20          798   db      ' '             ;
05A6 4D          799   db      'M'             ;
05A7 4F          800   db      'O'             ;
05A8 4E          801   db      'N'             ;
05A9 54          802   db      'T'             ;
05AA 48          803   db      'H'             ;
05AB 20          804   db      ' '             ;
05AC 20          805   db      ' '             ;
05AD 20          806   db      ' '             ;
05AE 20          807   db      ' '             ;
05AF 20          808   db      ' '             ;
05B0 20          809   db      ' '             ;
05B1 20          810   db      ' '             ;
05B2 20          811   db      ' '             ;
05B3 20          812   db      ' '             ;
05B4 20          813   db      ' '             ;
05B5 20          814   db      ' '             ;
05B6 20          815   db      ' '             ;
05B7 20          816   db      ' '             ;
05B8 20          817   db      ' '             ;
05B9 20          818   db      ' '             ;
05BA 20          819   db      ' '             ;
05BB 20          820   db      ' '             ;
05BC 20          821   db      ' '             ;
05BD 20          822   db      ' '             ;
05BE 20          823   db      ' '             ;
05BF 20          824   db      ' '             ;
05C0 20          825   db      ' '             ;
05C1 20          826   db      ' '             ;
05C2 20          827   db      ' '             ;
05C3 20          828   db      ' '             ;
05C4 20          829   db      ' '             ;
05C5 53          830   scr7:   db      'S'     ;
05C6 45          831   db      'E'             ;
05C7 54          832   db      'T'             ;
05C8 20          833   db      ' '             ;
05C9 20          834   db      ' '             ;
05CA 20          835   db      ' '             ;
05CB 20          836   db      ' '             ;
05CC 20          837   db      ' '             ;
05CD 20          838   db      ' '             ;
05CE 20          839   db      ' '             ;
05CF 20          840   db      ' '             ;
05D0 20          841   db      ' '             ;
05D1 20          842   db      ' '             ;
05D2 20          843   db      ' '             ;
05D3 20          844   db      ' '             ;
05D4 20          845   db      ' '             ;
05D5 20          846   db      ' '             ;
05D6 20          847   db      ' '             ;
05D7 20          848   db      ' '             ;
05D8 20          849   db      ' '             ;
05D9 20          850   db      ' '             ;
05DA 20          851   db      ' '             ;
05DB 20          852   db      ' '             ;
05DC 20          853   db      ' '             ;
05DD 20          854   db      ' '             ;
05DE 20          855   db      ' '             ;
05DF 20          856   db      ' '             ;
05E0 20          857   db      ' '             ;
05E1 20          858   db      ' '             ;
05E2 20          859   db      ' '             ;
05E3 20          860   db      ' '             ;
05E4 20          861   db      ' '             ;
05E5 20          862   db      ' '             ;
05E6 20          863   db      ' '             ;
05E7 20          864   db      ' '             ;
05E8 20          865   db      ' '             ;
05E9 20          866   db      ' '             ;
05EA 20          867   db      ' '             ;
05EB 20          868   db      ' '             ;
05EC 20          869   db      ' '             ;
05ED 20          870   db      ' '             ;
05EE 20          871   db      ' '             ;
05EF 20          872   db      ' '             ;
05F0 20          873   db      ' '             ;
05F1 20          874   db      ' '             ;
05F2 20          875   db      ' '             ;
05F3 20          876   db      ' '             ;
05F4 20          877   db      ' '             ;
05F5 20          878   db      ' '             ;
05F6 59          879   db      'Y'             ;
05F7 45          880   db      'E'             ;
05F8 41          881   db      'A'             ;
05F9 52          882   db      'R'             ;
05FA 20          883   db      ' '             ;
05FB 20          884   db      ' '             ;
05FC 20          885   db      ' '             ;
05FD 20          886   db      ' '             ;
05FE 20          887   db      ' '             ;
05FF 20          888   db      ' '             ;
0600 20          889   db      ' '             ;
0601 20          890   db      ' '             ;
0602 20          891   db      ' '             ;
0603 20          892   db      ' '             ;
0604 20          893   db      ' '             ;
                 894                           end
VERSION 1.2k ASSEMBLY COMPLETE, 0 ERRORS FOUND
```

CONTROL UNIT (CONTR.LST)

```
PAGE 1

1    $mod51
2    ;this is a small test program for CONTROL UNIT (CU)
3    ;sending/receiving data to/from the "LCD" module
4    ;article written for EFY
5    ;programmer-AR Karkare
6    ;date written-14 April 2002
7    ;uses 89c51 micro-controller with 11.059 mhz crystal
8     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9    ;RESERVED LOCATIONS
10
0045             11    intflg  bit     45h             ;interrupt indicator
                 12                                    ;LIST OF I/O
                 13
                 14                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
0000              15    org             0000h           ;
0000 802E         16    sjmp            start           ;skip all interrupt vactors
0023              17    org             0023h           ;
0023 8075         18    sjmp            spint           ;serial port program
0030              19    org             0030h           ;initialization of registers
0030 758160       20    start:          mov sp, #60h    ;set stack pointer
                  21                                    ;set SFRs
0033 758700       22    mov             pcon,#00h       ;smod=0
0036 758920       23    mov             tmod,#20h       ;timer1
                  24                                    ;gate=0,c/t=0,mode=8
                                                        bit,auto reload
0039 758BFD       25    mov             tl1, #0fdh      ;reload value for 9.6k baud rate
003C 758DFD       26    mov             th1, #0fdh      ;
003F 759850       27    mov             scon, #50h      ;mode 1,transmission/reception
                                                        enabled
0042 D28E         28    setb            tr1             ;start timer
0044 C2AF         29    clr             ea              ;global int. off
0046 D2AC         30    setb            es              ;serial int. on
0048 D2BC         31    setb            ip.4            ;high priority to serial port int.
004A D2B0         32    setb            rxd             ;float pin
004C D2B1         33    setb            txd             ;float pin
004E C245         34    clr             intflg          ;keep interrupt flag low
0050 7580FF       35    mov             p0, #0ffh       ;float all pins of port p0
                  36                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                  37                                    ;save some trial values
0053 75300E       38    mov             30h, #14        ;Day=14
0056 75310B       39    mov             31h, #11        ;Month=11
0059 75322C       40    mov             32h, #44        ;Year=1944
                  41                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                  42                                    ;wait till interrupt flag goes
```

```
                                                high
                43                                      ;clear interruot flag
                44                                      ;get what address received
                45                                      ;show on port p0
                46                                      ;if start character <STX>
                                                        received,
                47                                      ;get ready to receive address
                                                        and data
                48                                      ;otherwise treat character
                                                        received as address
                49                                      ;send data contents of the
                                                        address
                50                                      ;while sending
                51                                      ;wait for a while for stabilising
                52                                      ;send the data for the asked
                                                        address
005C D2AF       53      setb    ea                      ;global interrupt on
005E 3045FD     54      wait1: jnb      intflg, wait1   ;wait for serial port flag
0061 C245       55      clr     intflg                  ;clear flag
0063 E5F0       56      mov     a, b                    ;get  received character
0065 F580       57      mov     p0, a                   ;show on port p0 LEDs
0067 B40202     58      cjne    a, #02h, transmit ;if character is 02 <STX>
                                                        received,
CNTRNEW PAGE 2

006A 800A       59      sjmp    receive         ;jump to receiver program
                60                              ;otherwise accu. contains RAM
                                                address
006C            61      transmit:
006C 1200BC     62      lcall   del10m          ;stabilisation delay
006F F8         63      mov     r0, a           ;set address
0070 E6         64      mov     a, @r0          ;retrieve data
0071 12008C     65      lcall   send            ;send to RS232 port
0074 80E8       66      sjmp    wait1           ;wait for the next call
0076            67      receive:
0076 3045FD     68      wait2: jnb      intflg, wait2   ;wait for serial port flag
0079 C245       69      clr     intflg          ;clear flag
007B E5F0       70      mov     a, b            ;get received character
007D F580       71      mov     p0, a           ;show on port p0 LEDs
007F F8         72      mov     r0, a           ;save address at r0
0080 3045FD     73      wait3: jnb      intflg, wait3   ;wait for serial port flag
0083 C245       74      clr     intflg          ;clear flag
0085 E5F0       75      mov     a, b            ;get received character
0087 F580       76      mov     p0, a           ;show on port p1 LEDs
0089 F6         77      mov     @r0, a          ;save whatever data received
008A 80D2       78      sjmp    wait1           ;back to wait for next
                                                character
                79                              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
008C C2AF       80      send:   clr     ea      ;disable all interrupts
008E C299       81      clr     ti              ;pull ti flag low
0090 F599       82      mov     sbuf, a         ;load sbuf
0092 3099FD     83      waitt: jnb      ti, waitt       ;wait till ti flag goes high
0095 C299       84      clr     ti              ;pull ti flag low
0097 D2AF       85      setb    ea              ;enable all interrupts
0099 22         86      ret                     ;return
                87                              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
009A C2AF       88      spint:  clr     ea      ;disable all interrupts
009C C0D0       89      push    psw             ;save status
009E C0E0       90      push    acc             ;save accu.
00A0 C3         91      clr     c               ;clear carry
00A1 8599F0     92      mov     b, sbuf         ;save received character in B
00A4 C298       93      clr     ri              ;clear RI bit
00A6 D0E0       94      pop     acc             ;get back accu.
00A8 D0D0       95      pop     psw             ;get back status
00AA D2AF       96      setb    ea              ;enable all interrupts
00AC D245       97      setb    intflg          ;set interruppt flag
00AE 32         98      reti                    ;return from interrupt
                99                              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00AF 755004     100     del1m:  mov     50h, #04        ;delay of 1 milisec.
00B2 755153     101     loopa:  mov     51h, #83        ;
00B5 D551FD     102     loopb:  djnz    51h, loopb      ;
00B8 D550F7     103     djnz    50h, loopa      ;
00BB 22         104     ret                     ;
00BC 755028     105     del10m: mov     50h, #40        ;delay of 10 milisec.
00BF 755153     106     loopc:  mov     51h, #83        ;
00C2 D551FD     107     loopd:  djnz    51h, loopd      ;
00C5 D550F7     108     djnz    50h, loopc      ;
00C8 22         109     ret                     ;
00C9 75520A     110     del100m: mov    52h, #10        ;delay of 100 milisec.
00CC 1200BC     111     loope:  lcall   del10m  ;
00CF D552FA     112     djnz    52h, loope      ;
00D2 22         113     ret                     ;
00D3 75530A     114     del1s:  mov     53h, #10        ;delay of 1 sec.
00D6 1200C9     115     loopf:  lcall   del100m ;
00D9 D553FA     116     djnz    53h, loopf      ;
00DC 22         117     ret                     ;
                118                             end

VERSION 1.2k ASSEMBLY COMPLETE, 0 ERRORS FOUND
```

❑

SIMPLE MULTICHANNEL REMOTE CONTROL SYSTEM

KAUSHIK HAZARIKA

Multichannel remote control units let you access various features of stereo systems and switch between loads in lighting systems from a distance. Electronics hobbyists would certainly love to make such a remote control themselves. But they may find it difficult due to complex coding and decoding of remote signals and unavailability of microcontrollers and programmers for the unit.

The multichannel remote control system given here overcomes the aforesaid problems by using a simple frequency counting technique and decade counters. It can function up to a distance of 7 metres.

Block diagram

The block diagram of the multichannel remote control in Fig. 1 explains the basic operation of the circuit.

The transmitter section is shown within dotted lines. This unit based on a timer IC is a variable frequency oscillator. The keypad consists of tactile switches, which are used as inputs to generate different modulating frequencies. The signal generator consists of a timer IC to generate modulating frequency signals. Another timer IC is used as a carrier generator-cum-modulator.

The carrier generator oscillates at a frequency of 38 kHz. The modulating signal is mixed with the carrier signal. The modulated signal is then transmitted through the infrared LEDs. The modulated IR beam is demodulated by the IR module in the receiver section. This signal is inverted by an inverter and fed to the following section.

At the positive edge of the clock signal, the signal detector is enabled. The detector, in turn, triggers the monostable IC 555. If the clock signal is not present, the detector will check for the rising edge of the next incoming pulse to ensure correct counting of the received pulses.

Once triggered, the monostable goes high for a preset period decided by the timing components. During this period, the transistor switch allows the signal to go to counters 1 and 2 for counting.

When the internal clock of counter 2 goes high, both the counters are reset. As counter 1 counts, it gives carry-out (Co) pulse for every ten counts. For every Co pulse received from counter 1, the outputs of receiver go high and then low sequentially.

When the clock input at pin 14 of counter 2 goes low, the counting stops. Only one of its outputs remains high depending upon the the number of clock pulses received and the corresponding tristate switch gets enabled and the high output pulse triggers the respective flip-

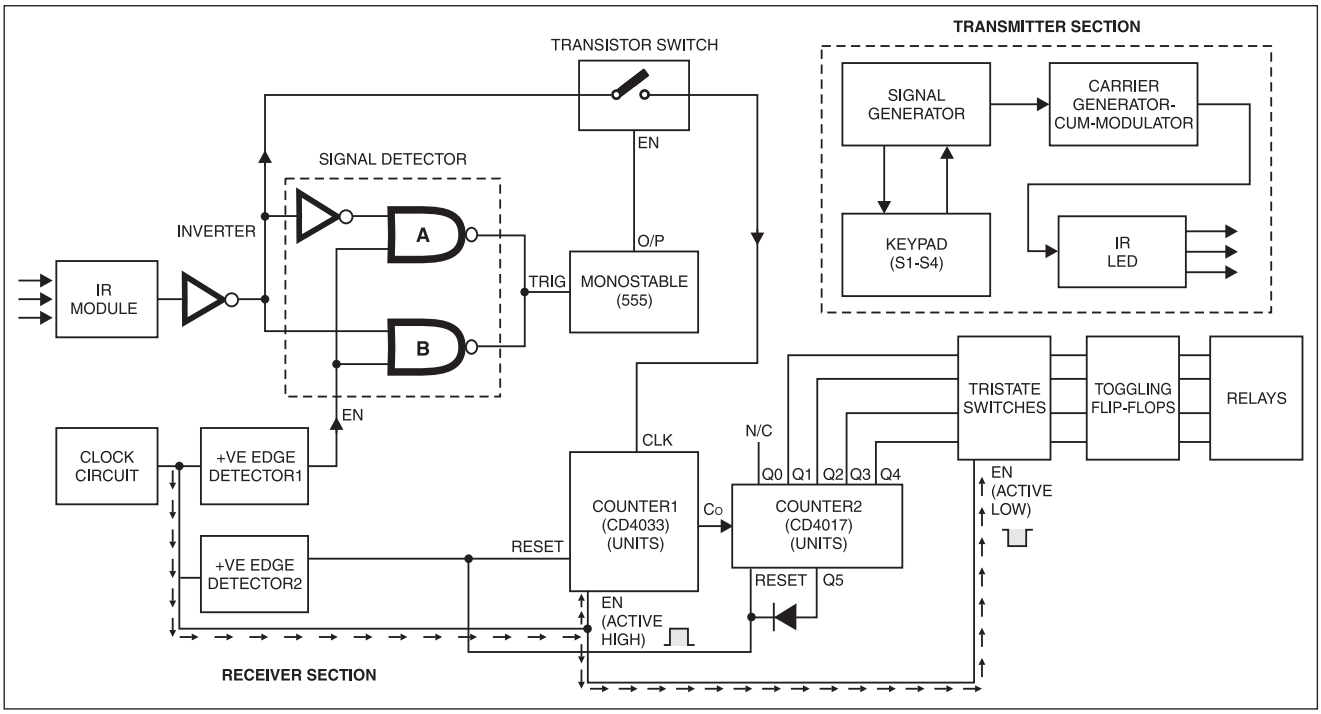


Fig. 1: Block diagram of multichannel remote control system

PARTS LIST

Semiconductors:

IC1, IC2, IC3, IC4 - NE555 timer
IC5 - CD4033 decade counter
IC6 - CD4017 decade counter
IC7, IC8 - CD4013 dual D-type flip-flop
T1 - SK100 pnp transistor
T2, T5, T7-T10 - BC558 pnp transistor
T3, T4, T6, T11-T14 - BC548 npn transistor
IR LED1, IR LED2 - Infrared LED
LED1 - Red LED for power-on
LED2 - Red LED for signal
LED3 - Red LED for clock pulse
LED4-LED7 - Red LED for counter output
LED8-LED11 - Red LED for appliances 1 through 4
D1 - 1N4148 diode
D2-D5 - 1N4001 diode

Resistors (all ¼-watt, ±5% carbon, unless stated otherwise):

R1 - 56-ohm
R2 - 820-ohm
R3 - 2.2-kilo-ohm
R4, R13 - 47-kilo-ohm
R5 - 1-kilo-ohm
R6 - 47-ohm
R7 - 22-ohm
R8 - 470-ohm
R9, R15 - 33-kilo-ohm
R10, R11, R20 - 22-kilo-ohm
R12, R22, R27 - 4.7-kilo-ohm
R14 - 1.5-kilo-ohm
R16, R21 - 100-kilo-ohm
R17 - 680-ohm
R18 - 150-kilo-ohm
R19 - 470-kilo-ohm
R23-R26, R29, R32, R34, R28, R31, R33 - 3.3-kilo-ohm
R35, R36 - 56-kilo-ohm
R30, R35 - 220-ohm
VR1-VR4 - 10-kilo-ohm preset
VR5 - 2.2-kilo-ohm
VR6 - 1-mega-ohm

Capacitors:

C1, C3, C4, C8, C11-C13, C14, C15, C16 - 0.01μF ceramic
C2, C9 - 1μF, 25V electrolytic
C5, C7 - 220pF ceramic
C6, C12 - 0.02μF ceramic
C10 - 2.2μF, 25V electrolytic

Miscellaneous:

S1-S4 - Tactile switch
S5 - On/off slide switch
RL1-RL4 - 6V, 200-ohm, 1C/O relay
Power supply - 9V battery, 6V DC regulated

flop to drive a relay.

The transmitter

The transmitter circuit, shown in Fig. 2, is powered by a 9V battery through 56-ohm resistor R1. LED1 glows when power switch S5 is closed.

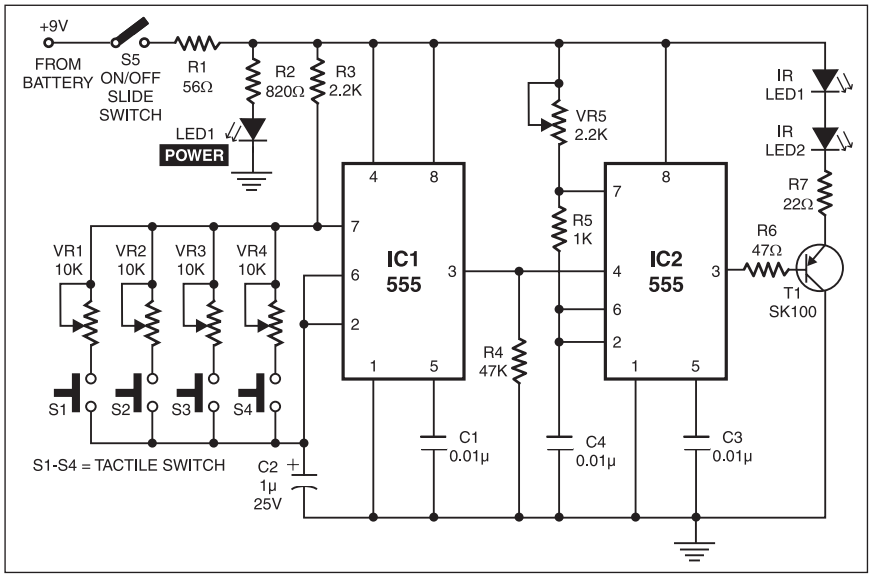


Fig. 2: Transmitter circuit

The transmitter consists of two NE555 ICs (IC1 and IC2), keypad switches S1 through S4, and IR LEDs. Both the ICs are wired in astable mode. IC1 functions as the modulating signal generator, while IC2 functions as the carrier generator-cum-modulator.

The modulating signal is generated when the switch connected to pin 7 of IC1 is pressed. Presets VR1 through VR4 in series with switches S1 through S4, respectively, are used to generate different modulating frequencies from 200 to 500 Hz. The preset, along with the specific switch pressed, is tuned to a particular modulating frequency such that a particular output (relay) is turned on or off at the receiver end. For example, to turn on output 1 (O1), adjust 10k potmeter VR1 while keeping switch S1 depressed until the output 1 is turned on. If the output doesn't turn on, release S1 and then depress it again continuously until it is on.

The output of IC1 at pin 3 is connected to pin 4 of IC2. IC2 generates a carrier frequency of 38 kHz. The modulated signal from pin 3 of IC2 is fed to the base of transistor T1 (SK100) and then transmitted through infrared LEDs (IR LED1 and IR LED2). The output frequency (f) at pin 3 of timer IC 555 is given by:

$$f = 1.443/(R_a + 2R_b)C$$

where R_a is the resistance between $+V_{cc}$ and pin 7, R_b is the resistance between pins 6 and 7, and C is the capacitance between pin 6 and ground.

The receiver

Fig. 3 shows the receiver circuit. The modulated signal is received by the IR sensor module (TSOP1738) of the receiver section. The negative pulse of the demodulated signal from the sensor is inverted by transistor T2 (BC558).

Initially, output pin 3 of the IR module is high at 6V. When the module detects the IR signal, its pin 3 goes from high to low state. As a result, LED2 is forward biased and it glows to indicate that the signal is being received by the sensor. When LED2 glows, transistor T2 conducts as the potential at its base is low. Thus the transistor conducts when the sensor output goes low.

This inverting action is depicted by the NOT gate in the block diagram. Transistors T2 through T4 form the signal detector. This detector is enabled by the positive voltage at the base of transistor T4. This voltage is given as a pulse by 220pF capacitor C7 only at the rising edge of the clock (Ck) generated by IC4 (NE555).

When transistor T2 conducts, about 6V becomes available at the base of transistor T3 (BC548) and at the collector of transistor T6 (BC548) simultaneously. Transistor T3 conducts to pull input pin 2 of IC3 (NE555) low. This negative going pulse at pin 2 triggers monostatble multivibrator IC3 to generate a clock pulse of about 1 Hz.

The base of transistor T6 receives the clock pulse from output pin 3 of IC3 and the transistor conducts. Due to the conduction of transistor T6, IC5 (CD4033) receives the strobe signal at its pin 2.

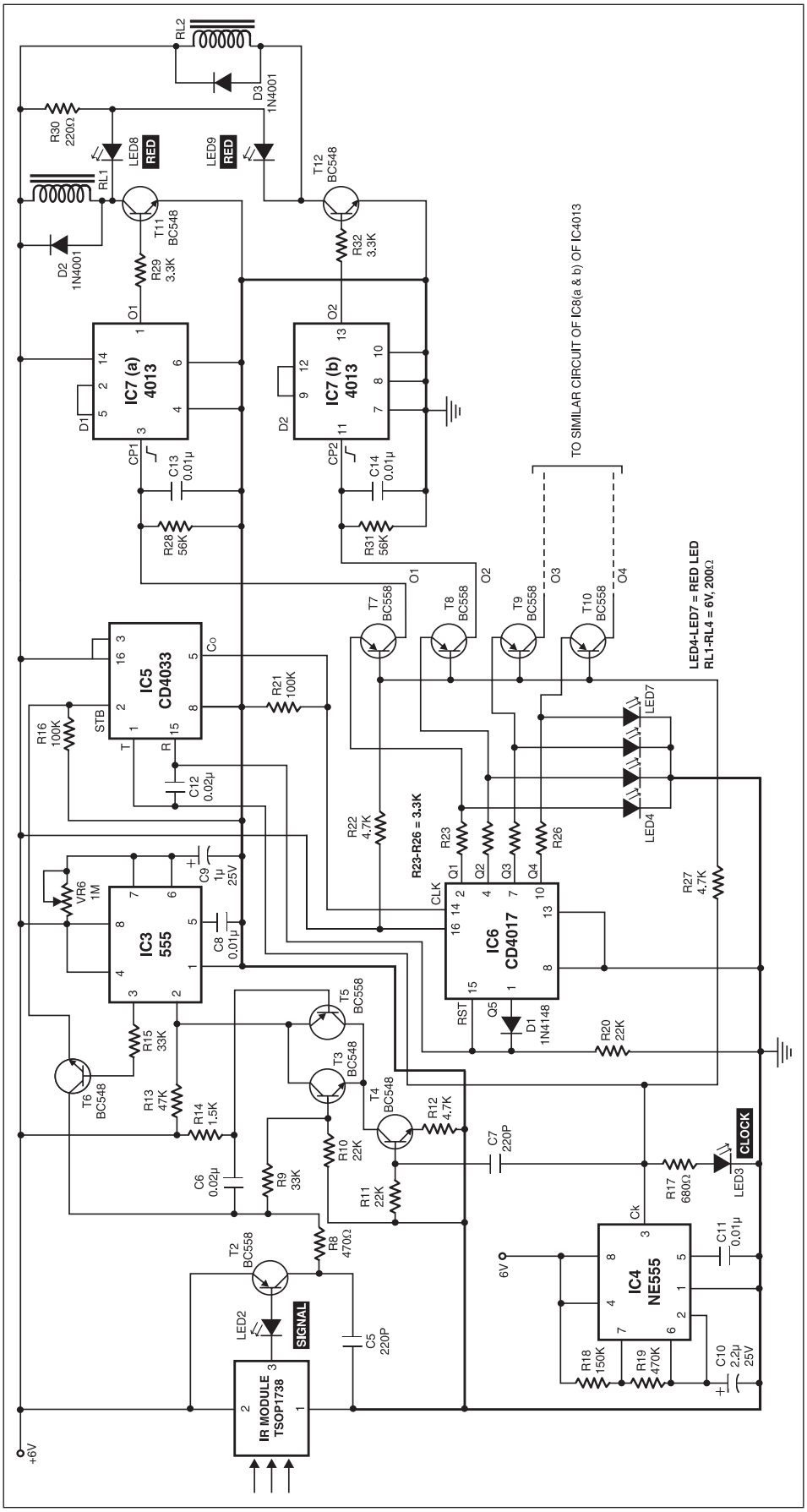


Fig. 3: Receiver circuit

Thus transistor T6 acts as a switch between the received signal and the strobe signal at pin 2 of IC5. It allows the demodulated signal to enter the counter section of the circuit for a preset period decided by 1µF capacitor C9 and 1-mega-ohm potmeter VR6. During this period, the counter ICs (IC5 and IC6) count the input frequency.

IC4 (NE555) is configured in astable mode and functions as a clock generator. It continuously generates a clock pulse of 1.6-second duration at its pin 3. This pulse is simultaneously fed to pin 1 of IC5 and the bases of transistors T7 through T10 (BC558) via 4.7k resistor R27.

IC5 (CD4033) is a decade counter that provides the required clock pulse to the second counter (IC6). It counts the units and after every ten counts sends a carry-out at its pin 5.

IC6 (CD4017) is also a decade counter. It receives the clock from pin 5 of IC5 at its input pin 14. Output pins 2, 4, 7, and 10 of IC6 are connected to the emitters of transistors T7 through T10 via resistors R23 through R26 (each 3.3 kio-ohms), respectively. As Q1 output of IC6 is normally high, it is not used. Thus only nine out of possible ten outputs can be used. Here we've used only four outputs.

When the clock (Ck) goes low, counting stops and only the last high output is passed to the flip-flop circuit through the corresponding tristate switch (T7, T8, T9, or T10). The collectors of transistors T7 and T8 are connected to pins 3 and 11 of IC7, respectively. IC7 and IC8 each comprises two flip-flops. The total four flip-flops, namely, IC7(a), IC7(b), IC8 (a), and IC8(b), are wired in toggle mode. These flip-flops change state for every positive-going pulse appearing at their inputs. The flip-flop outputs are fed to relay driver transistors.

IC7(CD4013) is configured as a latch dual D-type flip-flop.

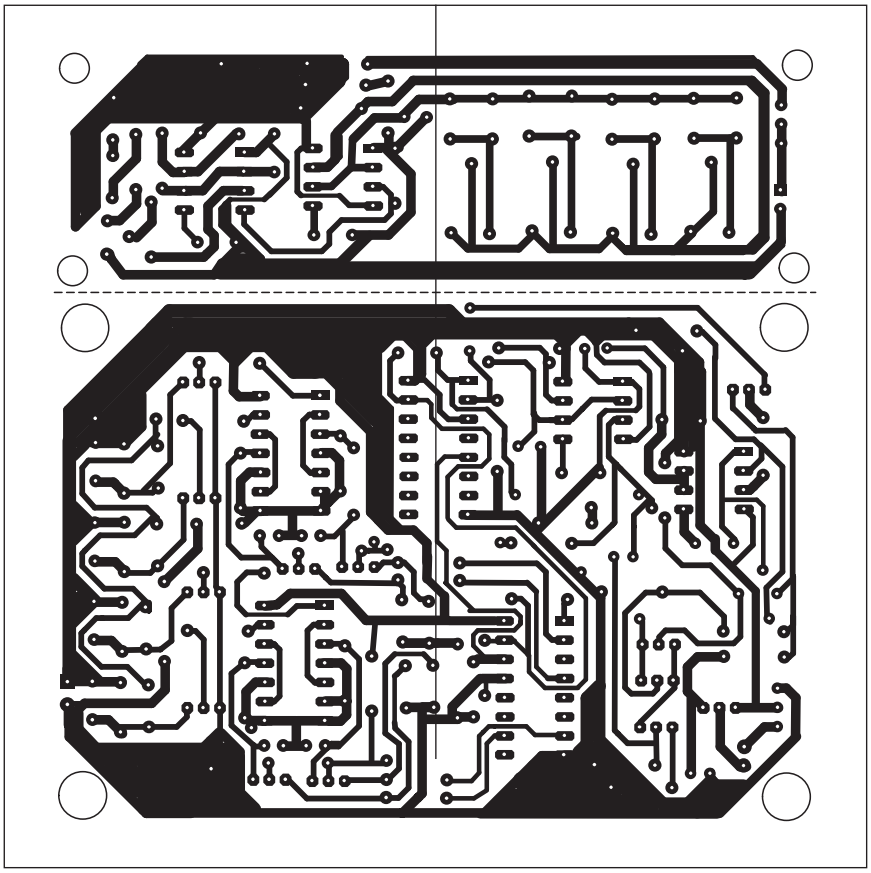


Fig. 4 :The actual-size, solder-side PCB of the multichannel remote control system comprising transmitter (above) and receiver (below) sections

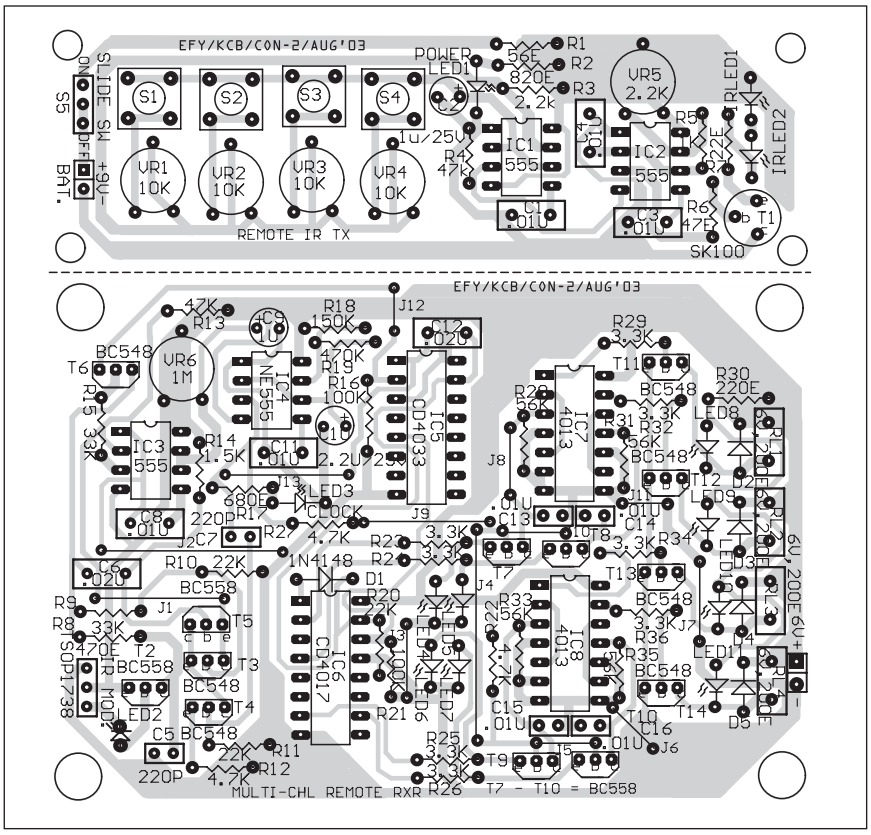


Fig. 5: Component layout for the PCB

Its output pin 1 is latched when a positive-going transition clock pulse is received at pin 3. Similar is the case at output pin 13 when a positive-going transition pulse is received at pin 11 of this IC. Output pins 1 and 13 of IC7 are connected to relay driver transistors T11 and T12 (BC548) via resistors R29 and R32 (each 3.3 kilo-ohms), respectively. The glowing of LED8 and LED9 indicates energisation of relays RL1 and RL2.

The collectors of transistors T9 and T10 (BC558) are fed to pins 3 and 11 of IC8 (CD4013), respectively. Output pins 1 and 13 of IC8 are connected to relay driver transistors T13 and T14 (BC558) via resistors R34 and R36, respectively. The glowing of LED10 and LED11 indicates energisa-tion of relays RL3 and RL4.

LED4 through LED7 connected via resistors R23 through R26, respectively, indicate the status of the output of IC6. The glowing of LED4 indicates that output pin 2 (Q1) of IC6 is high, which means that the appliance connected through relay RL1 will be turned on or off. Similarly, the glowing of LED5 indicates that output pin 4 of IC6 is high and relay RL2 is activated.

Calibration

After all the connections are done, switch on both the transmitter and the receiver. Place the IR LEDs and the IR sensor facing each other about 10 cm apart.

Now on pressing any of switches S1 through S4 in the transmitter section, LED2 in the receiver section should glow. During the high input clock pulse (Ck), you'll observe a light running at the outputs of IC6. At low input clock pulse, the flip-flop toggles. So you will see one of the LEDs connected to the relay driver (LED8 through LED11) glowing for every positive-going clock (CP).

In the transmitter section, adjust any of presets VR1 through VR4 and press the switch connected in series with it such that the relay connected to the appliance you want to turn on/off gets activated. The relay activation is indicated by the glowing of the corresponding LED (LED8 through LED11). Release the switch once the desired load is turned on/off.

The actual-size, solder-side PCB of the multichannel remote control system comprising transmitter (above) and receiver (below) sections is shown in Fig. 4 and its

component layout in Fig. 5. The combined PCB can be cut along the dotted lines to separate the remote transmitter unit and the receiver unit.

Cautions. 1. The switch (S1, S2, S3, or S4) must be kept depressed and the transmitter oriented towards the receiver sensor until the desired load turns on/off. Release the button during the low period of the clock. This can be easily achieved by observing the output LEDs (LED8-LED11).

2. It may take 2 to 3 seconds to turn on/off a load. Break of the signal during this period may cause switching of a different load. ❑

Readers' comments:

Q1. If any of switches S1 through S4 in the transmitter section is pressed when preset VR6 is at a low resistance, LED2 in the receiver section blinks. But there is no light running through LED4 through LED7 and LED8 through LED11 are always in 'on' condition. When preset VR6 is at a high resistance, LED2 glows continuously.

Q2. After I made a small correction in the receiver circuit as shown in Fig. 1 here, I observed light running through LED4 through LED7. But still LED8 through LED11 are always in 'on' condition. What could be the reason?

R. Senthil Kumar
Ranipet, Tamil Nadu

The author, Kaushik Hazarika, replies:

A1. VR6 determines the duration of timing pulse given out by IC3 (NE555) configured as monostable circuit. This pulse activates transistor T6 (BC548), which is acting as a switching device, and allows the incoming signal to go to the counters. A very low-output frequency setting will not result in any count as the incoming signal will be blocked during the cut-off period of the transistor (see Fig. 1 in the article). For proper working of the counters, set VR6 such that the output of the monostable is approx. 1 second.

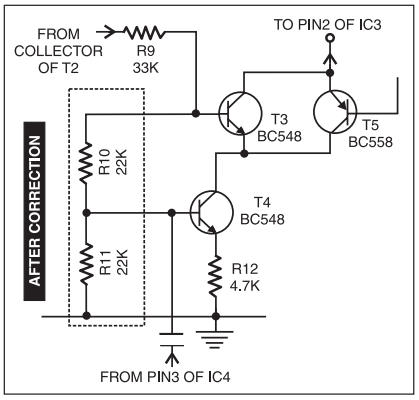


Fig. 1: Modified circuit of simple multichannel remote control system

A2. The suggested modification may result in spurious signal count. Simply replace 33k resistor R9 with around 15k resistor, instead. The fault may be due to Hfe variations between different makes of transistors of the same number. The outputs are high due to false triggering of flip-flops caused by the noisy power supply. Use the power-on-reset circuit by disconnecting pins 4 and 10 from GND (see Fig. 1 in the article). The outputs should remain low now. Otherwise, check flip-flops (CD4013).

MICROPROCESSOR-CONTROLLED THERMOMETER

BHASKAR BANERJEE

Here's a microprocessor-controlled thermometer that displays the maximum as well as the minimum temperatures over a time period. It can also display the temperature at any instant by momentary push of a button. It has the following features:

1. The thermometer is based on 80C85 microprocessor.
2. The current temperature can be viewed at the press of a button.
3. The system has a resolution of 0.5°C
4. The readings are displayed alphanumerically on an LCD.

Description

Fig. 1 shows the circuit of the microprocessor-controlled thermometer. It comprises a microprocessor, temperature sensor, analogue-to-digital converter, memory devices, and LCD module.

The microprocessor (IC 80C85). The main component of the circuit is the 80C85 microprocessor, which takes the input from an analogue-to-digital converter, does all the calculations, stores the result in respective memory locations, and shows it on the LCD, besides generating all the control signals for the system. The maximum temperature is displayed in the first line and the minimum temperature is displayed in the second line of the LCD.

When switch S2 is pressed, a positive-going pulse is given to pin 7 (RST7.5) of IC 80C85 (IC1) and the processor jumps to appropriate interrupt service routine (ISR) to show the present temperature on the display. After some time, the processor returns to the normal display (maximum and minimum temperature display).

IC1 is driven by a 2MHz clock oscillator. IC 80C85 is a CMOS version of 8085, which consumes less power than IC 8085.

Temperature sensor (IC LM35). IC LM35 is used as the temperature sensor. It is very easy to use and its output voltage is linearly proportional to the temperature in centigrade scale. The scale factor of the IC is 10 mV/°C. The sensor's output can be directly fed to the analogue-to-digital converter, requiring no zero adjustment or calibration. It is preferable to use TO46 metal-can package version such as IC LM35 DH.

Analogue-to-digital converter (IC ADC0804). The analogue-to-digital converter is 8-bit, microprocessor-compatible IC ADC0804 (IC6). This IC does not require any zero adjustment. And we only need to set the voltage at pin 9 ($V_{Ref}/2$) to 0.64 volt to obtain a full scale deflection at 128°C. The IC has an on-chip clock generator running at a frequency of about 560 kHz. Its $\overline{RD}$, $\overline{WR}$, and $\overline{CS}$ pins directly interface with the processor.

To start the conversion, the processor writes a binary number to the ADC through the bidirectional data bus that is directly connected to the system data bus. Typically, the IC takes 100 μs for the conversion. After conversion, an INTR pulse is generated at pin 5 to interrupt the processor. But here, a time delay is given before taking the output from the ADC. The ADC has a resolution of 0.5°C.

EEPROM (IC AT28C64). All the software for the system is stored in EEPROM IC AT28C64 (IC3). We've used this particular EEPROM instead of EPROM because it is easy to program and requires no special programming voltage and no UV light source, and also data bytes are erasable. However, you can safely use an EPROM like IC 2716.

The binary data taken from the analogue-to-digital converter is converted into a decimal number equivalent to the temperature by the processor. For this, a look-up table (given on page 86 of the article) is stored in the monitor (IC3) from page 02 onwards. The look-up table is given for temperatures from 0.000°C to 99.9°C.

RAM (IC 6116). RAM IC 6116 (IC4) temporarily stores display information, stack for the processor, etc. When the system is switched on, the display information table stored in IC3 gets transferred to the RAM. Then all the subsequent values of the temperature get stored in some particular memory locations and are taken from there whenever they need to be displayed.

PARTS LIST

Semiconductors:

IC1	- 80C85 microprocessor
IC2	- 74LS373, 8-bit latch
IC3	- AT28C64 EEPROM
IC4	- 6116 RAM
IC5	- 74LS139 demultiplexer
IC6	- ADC0804 analogue-to-digital converter
IC7	- 74LS00 NAND gate (N1-N4)
IC8	- 74LS02 NOR gate (N5-N6)
IC9	- LM35 temperature sensor

Resistors (all ¼-watt, ±5% carbon, unless stated otherwise):

R1, R2	- 10-kilo-ohm
R3	- 100-ohm
R4	- 4.7-kilo-ohm
VR1, VR2	-10-kilo-ohm preset

Capacitors:

C1, C2	- 22pF ceramic disk
C3	- 10μF, 16V electrolytic
C4	- 1μF, 16V electrolytic
C5	- 68pF ceramic disk

Miscellaneous:

LCD	- 16-character×2-line LCD module
S1, S2	- Tactile switch
X_{TAL}	- 2MHz crystal oscillator
Power supply	- 5V regulated

LCD module. The LCD module used has two lines, each having 16 characters. An LCD module with backlight is preferable. For backlight, connect pin 15 to $+V_{cc}$ and pin 16 to ground. A soft switch may be connected between 5V and pin 15 to switch on the backlight only when it is required.

Pin 2 is the positive supply terminal and pin 1 is the ground terminal of the LCD module. Pins 7 through 14 are data pins and the data is transferred through these pins. Pin 6 is enable pin and pin 5 is read/write pin. The data is written to the LCD when pin 6 is low and the data is read when pin 6 is high.

Construction

The circuit can be assembled on any general-purpose PCB. However, a proper PTH double-side PCB is recommended for its proper working. Utmost care

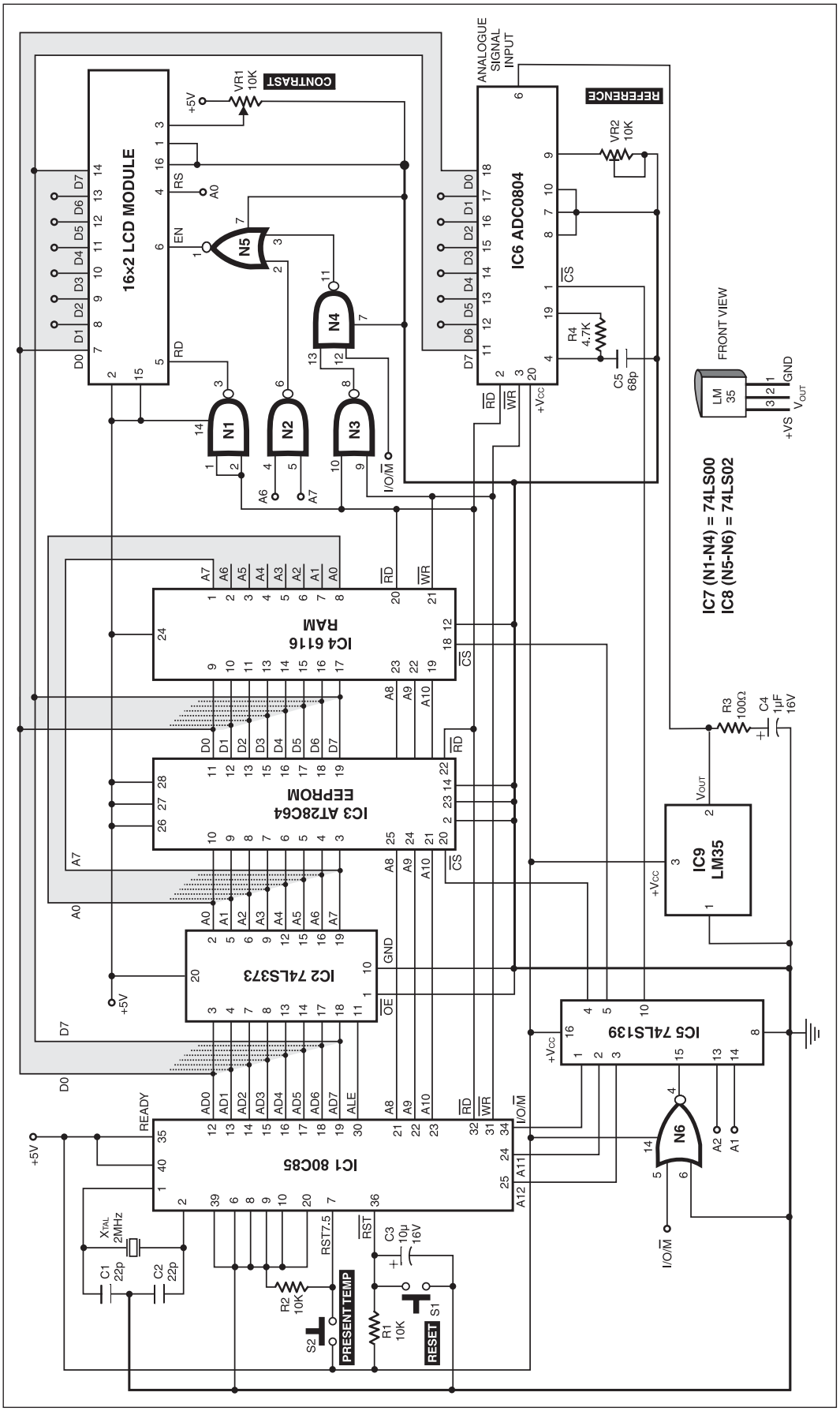


Fig. 1: Circuit of the microprocessor-controlled thermometer

should be taken while making connections because any short circuit between data lines, address lines, and control bus will cause malfunctioning. Handle the LCD module carefully as it is very expensive.

After the construction is done, the display must show the maximum temperature in the first line and the minimum temperature in the second line. Now press switch S2 to display the present temperature. After some time, the display will return to the maximum and minimum display.

If at any step, a different or ambiguous data is displayed, or the display shows nothing, check the system carefully. If there is no hardware fault, check the monitor (IC3) and the software loaded in it for correct entry of data.

If the system is working, set the voltage at pin 9 of IC6 to 0.64 volt with the help of 10-kilo-ohm trim-pot VR2. Press reset switch S1 to get a proper display of temperature, initially. The system draws a current of about 50 mA when the LCD backlight is off and about 180 mA when the backlight is on.

The actual-size, solder-side and component-side track layouts for the microprocessor-controlled thermometer are

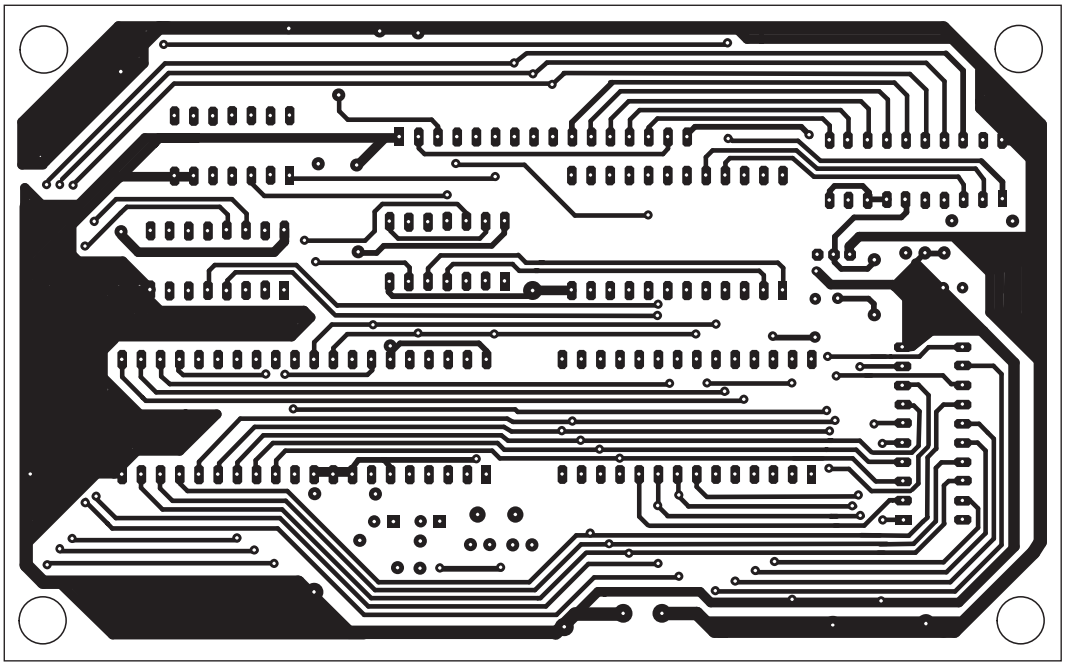


Fig. 2: Actual-size, solder-side track PCB layout of the microprocessor-controlled thermometer

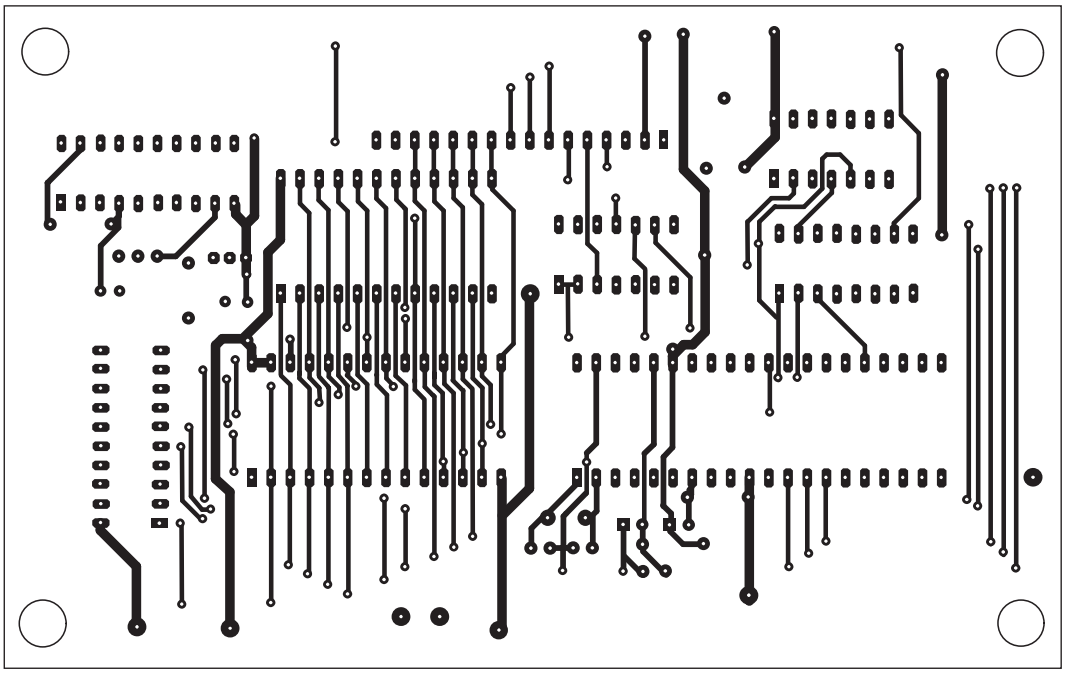


Fig. 3: Actual-size, component-side track PCB layout of the microprocessor-controlled thermometer

shown in Figs 2 and 3, with their component layout in Fig. 4.

Software program

The program is assembled using 8085 cross-assembler version 3.00b (supplied with 'Learn to Use Microprocessors' book published by EFY). The assembly program 'thermo.asm' can be written using any text editor. The listing file 'thermo.1st' is given at the end of the article. The program is self-explanatory. The flow-charts for quick understanding of the overall program and the main subroutine are shown in Figs 5 and 6 respectively.

The program begins at the memory location at address 0000H. After the interrupts of the processor are initialised, the program goes to subroutine BEGIN using jump instruction. Then the main program starts at memory location 0100H and loads the display information from ROM to RAM. If the data is correctly loaded, the program initialises the LCD and calculates the maximum and minimum temperatures for display.

DLY1 and DLY2 are delay subroutines, whereas INADC1 and INADC2 are data-input subroutines for the analogue-to-digital converter. The subroutine INTLCD initialises the LCD. The DISM1 subroutine is for maximum and minimum temperature displays, and the DISM2 subroutine is for present temperature display.

After getting the data from the look-up table, the HDSTS subroutine displays the value of the maximum temperature, the LDSTS subroutine displays the minimum temperature, and the CDSTS subroutine displays the present temperature. The label DIT in the program indicates the location where the alphanumeric display information table is stored.

Operation

When the power is switched on, the LCD shows *??.?'C Maximum* in the first line and *??.?'C Minimum**- in the second line. After a few seconds, it displays the exact maximum and minimum temperatures in place of the question marks. If nothing appears or ambiguous data is displayed, press reset switch S1 for proper display.

On the display, the maximum temperature keeps on increasing as the temperature rises and the minimum temperature keeps on decreasing as the temperature falls. The displayed temperature values don't change any fur-

ther once they reach maximum and minimum values. And you'll have to press S1 again to display new readings (say, on the next day).

When switch S2 is pressed, the display shows *****??.?'C***** in the first line and ***At Present*** in the second line. After about 6 seconds, it returns to Maximum and Minimum display again. If the temperature sensor is disconnected, '00.0' appears in place of '??.?' in the temperature display.

Note. All relevant files including cross assembler, Thermo.asm file, the look-up table etc are included in the CD.

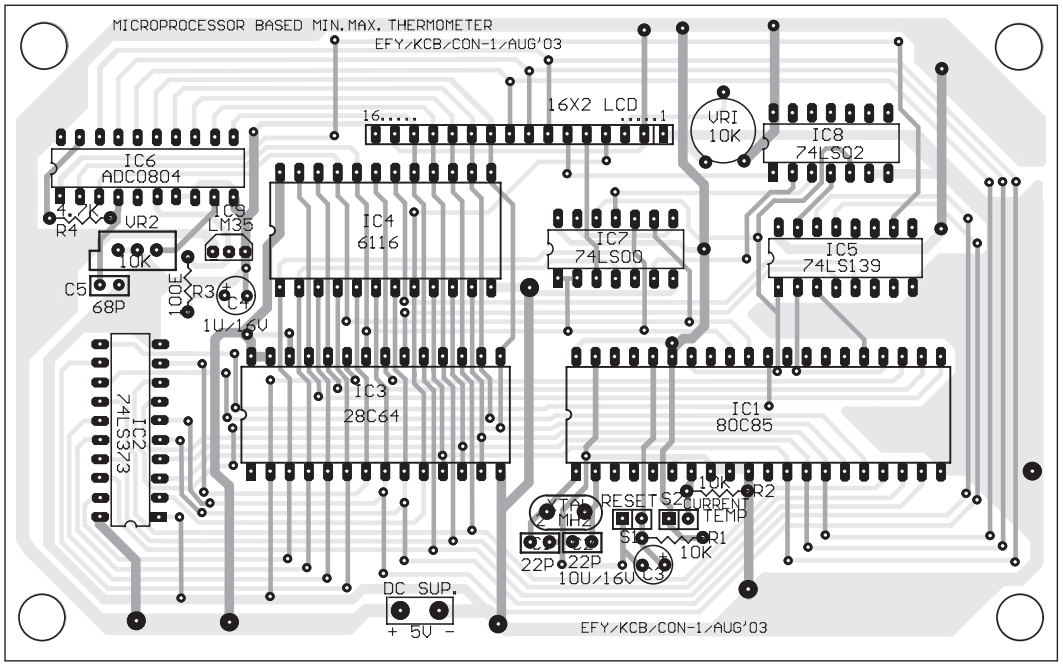


Fig. 4: Component layout for the PCBs in Figs 2 and 3

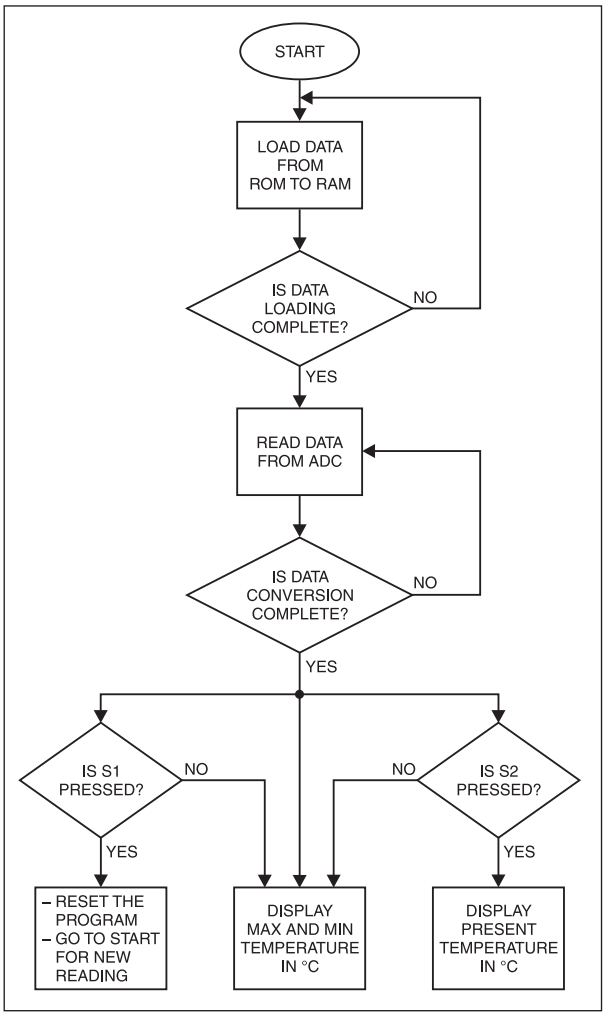


Fig. 5: Flow-chart of the program

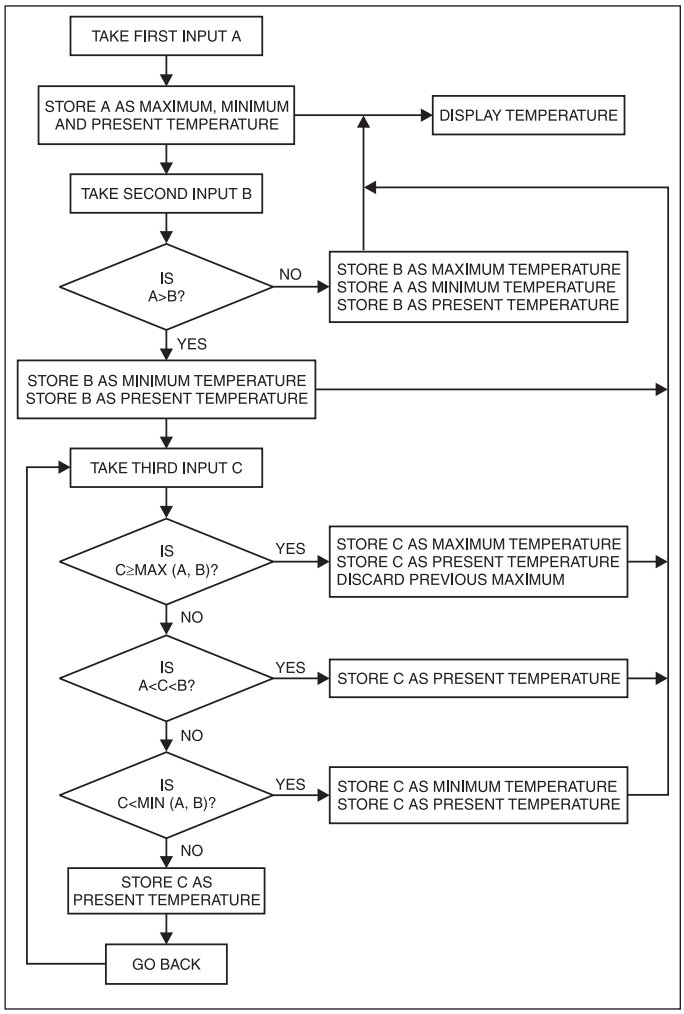


Fig. 6: Flow-chart of main routine

Look-Up Table

0200	30	30	30	30	30	35	30	31	30	30	31	35	30	32	30	30
0210	32	35	30	33	30	30	33	35	30	34	30	30	34	35	30	35
0220	30	30	35	35	30	36	30	30	36	35	30	37	30	30	37	35
0230	30	38	30	30	38	35	30	39	30	30	39	35	31	30	30	31
0240	30	35	31	31	30	31	31	35	31	32	30	31	32	35	31	33
0250	30	31	33	35	31	34	30	31	34	35	31	35	30	31	35	35
0260	31	36	30	31	36	35	31	37	30	31	37	35	31	38	30	31
0270	38	35	31	39	30	31	39	35	32	30	30	32	30	35	32	31
0280	30	32	31	35	32	32	30	32	32	35	32	33	30	32	33	35
0290	32	34	30	32	34	35	32	35	30	32	35	35	32	36	30	32
02A0	36	35	32	37	30	32	37	35	32	38	30	32	38	35	32	39
02B0	30	32	39	35	33	30	30	33	30	35	33	31	30	33	31	35
02C0	33	32	30	33	32	35	33	33	30	33	33	35	33	34	30	33
02D0	34	35	33	35	30	33	35	35	33	36	30	33	36	35	33	37
02E0	30	33	37	35	33	38	30	33	38	35	33	39	30	33	39	35
02F0	34	30	30	34	30	35	34	31	30	34	31	35	34	32	30	34
0300	32	35	34	33	30	34	33	35	34	34	30	34	34	35	34	35
0310	30	34	35	35	34	36	30	34	36	35	34	37	30	34	37	35
0320	34	38	30	34	38	35	34	39	30	34	39	35	35	30	30	35
0330	30	35	35	31	30	35	31	35	35	32	30	35	32	35	35	33
0340	30	35	33	35	35	34	30	35	34	35	35	35	30	35	35	35
0350	35	36	30	35	36	35	35	37	30	35	37	35	35	38	30	35
0360	38	35	35	39	30	35	39	35	36	30	30	36	30	35	36	31
0370	30	36	31	35	36	32	30	36	32	35	36	33	30	36	33	35
0380	36	34	30	36	34	35	36	35	30	36	35	35	36	36	30	36
0390	36	35	36	37	30	36	37	35	36	38	30	36	38	35	36	39
03A0	30	36	39	35	37	30	30	37	30	35	37	31	30	37	31	35
03B0	37	32	30	37	32	35	37	33	30	37	33	35	37	34	30	37
03C0	34	35	37	35	30	37	35	35	37	36	30	37	36	35	37	37
03D0	30	37	37	35	37	38	30	37	38	35	37	39	30	37	39	35
03E0	38	30	30	38	30	35	38	31	30	38	31	35	38	32	30	38
03F0	32	35	38	33	30	38	33	35	38	34	30	38	34	35	38	35
0400	30	38	35	35	38	36	30	38	36	35	38	37	30	38	37	35
0410	38	38	30	38	38	35	38	39	30	38	39	35	39	30	30	39
0420	30	35	39	31	30	39	31	35	39	32	30	39	32	35	39	33
0430	30	39	33	35	39	34	30	39	34	35	39	35	30	39	35	35
0440	39	36	30	39	36	35	39	37	30	39	37	35	39	38	30	39
0450	38	35	39	39	30	39	39	35	39	39	39	FF	FF	FF	FF	FF

ASSEMBLY PROGRAM (THERMO.LST)

```
2500 A.D. 8085 CROSS ASSEMBLER  -  VERSION 3.00b
-------------------------------------------------------------------------------
INPUT  FILENAME :  THERMO.ASM
OUTPUT FILENAME :  THERMO.LST
 1                                                  ; ASSEMBLY LANGUAGE FOR
                                                    MICROPROCESSOR BASED
                                                    THERMOMETER */
 2
 3    0000
 4    0000    ORG 0000H
 5    0000    FB                  EI              ; Enable all interrupts
 6    0001    3E 18               MVI A,18H       ; Load interrupt bit pattern
 7    0003    30                  SIM             ; Enable RST 6.5 & RST
 8                                                ; 5.5 ,reset RST7.5
 9    0004    C3 00 01            JMP BEGIN       ; Go to BEGIN
10    0007    00                  NOP             ; Delay subroutine 1
11    0008    F5          DLY1:   PUSH PSW        ; Save processor status
12    0009    D5                  PUSH D          ; save contents of D,E on stack
13    000A    11 50 03            LXI D,0350H     ; Load 50 in E, 03 in D
14    000D    1B          A:      DCX D
15    000E    7A                  MOV A,D
16    000F    B3                  ORA E
17    0010    C2 0D 00            JNZ A
18    0013    D1                  POP D           ; Transfers contents
19                                                ; of stack to DE
20    0014    F1                  POP PSW
21    0015    C9                  RET
22    0016    F5          DLY2:   PUSH PSW        ; Delay subroutine 2
23    0017    D5                  PUSH D
24    0018    11 FF FF            LXI D,FFFFH
25    001B    1B          B:      DCX D
26    001C    7A                  MOV A,D
27    001D    B3                  ORA E
28    001E    C2 1B 00            JNZ B
29    0021    D1                  POP D
30    0022    F1                  POP PSW
31    0023    C9                  RET             ; Command write subroutine-
32    0024    D3 C0       CWS:    OUT C0H         ; contents of Acc. are copied
33                                                ; to out port address
34                                                ; C0H. i.e., this subroutine
35                                                ; writes a command in
36                                                ; the LCD where port COH is
the
37                                                ; address of cammand in RAM
of
                                                  LCD.
38    0026    CD 08 00            CALL DLY1
39    0029    C9                  RET             ; Data write subroutine-
40    002A    F5          DWS:    PUSH PSW
41    002B    DB C0       C:      IN C0H          ; contents of port at CO
42                                                ; is read and loaded to acc.
43                                                ; i.e., this subroutine
44                                                ; write display data to DDRAM
45                                                ; of LCD having port add C1H
46    002D    E6 80               ANI 80H
47    002F    C2 2B 00            JNZ C           ; contents of Acc. are copied
48                                                ; to output port address C1
49    0032    F1                  POP PSW
50    0033    D3 C1               OUT C1H
51    0035    C9                  RET
52    0036    00                  NOP
53    0037    00                  NOP
54    0038    00                  NOP
55    0039    00                  NOP
56    003A    00                  NOP
57    003B    00                  NOP
58    003C    C3 88 00            JMP DISM2       ; Call location for RST 7.5.
59    003F    00                  NOP             ; This subroutine shows
60                                                ; present Temperature.
61    0040    CD 16 00    INADC1:                 CALL DLY2       ;   This
subroutine takes
62                                                ; INPUT from ADC
63    0043    CD 16 00            CALL DLY2
64    0046    C3 E8 01            JMP INADC2      ; To get input from ADC
65    0049    C9          SUB     RET             ; This subroutine
66    004A    3E 38       INTLCD:                 MVI A,38H       ;
initialises LCD
67    004C    CD 24 00            CALL CWS
68    004F    3E 0E               MVI A,0EH
69    0051    CD 24 00            CALL CWS
70    0054    3E 14               MVI A,14H
71    0056    CD 24 00            CALL CWS
72    0059    3E 01               MVI A,01H
73    005B    CD 24 00            CALL CWS
74    005E    C9                  RET
75    005F    00                  NOP
76    0060    7E          STRING:                 MOV A,M ; This subroutine dis-
plays
77                                                ; a string of data on the LCD
78    0061    FE A0               CPI A0H
79    0063    CA 6F 00            JZ AHEAD
80    0066    CD 2A 00            CALL DWS        ; To Data Write Subroutine
81    0069    23          BACK:   INX H
82    006A    0D                  DCR C
83    006B    C8                  RZ              ; Return on zero
84    006C    C3 60 00            JMP STRING
85    006F    23          AHEAD:  INX H
86    0070    7E                  MOV A,M
87    0071    CD 24 00            CALL CWS
88    0074    C3 69 00            JMP BACK
89    0077    00                  NOP
90    0078    CD 4A 00    DISM1:  CALL INTLCD ; This subroutine displays
91                                                ; max and min Temp. on LCD
92    007B    F5                  PUSH PSW
93    007C    E5                  PUSH H
94    007D    21 00 08            LXI H,0800H
95    0080    0E 22               MVI C,22H
96    0082    CD 60 00            CALL STRING
97    0085    E1                  POP H
98    0086    F1                  POP PSW
```

```
0087  C9              RET
0088  CD 4A 00  DISM2:  CALL INTLCD  ; This subroutine displays
                                     ; present Temp. on LCD, this
                                     ; is called when RST 7.5
                                     ; is activated.
008B  F5              PUSH PSW
008C  E5              PUSH H
008D  21 22 08        LXI H,0822H
0090  0E 22           MVI C,22H
0092  CD 60 00        CALL STRING
0095  E1              POP H
0096  F1              POP PSW
0097  CD 16 00        CALL DLY2
009A  FB              EI
009B  C3 5C 01        JMP INP3
009E  00              NOP
009F  00              NOP
00A0  F5        HDSTS:  PUSH PSW  ; This subroutine stores
                                     ; Max. Temp. in RAM
                                     ; after getting
                                     ; the display data from
                                     ; LOOK-UP table.
00A1  5C              MOV E,H
00A2  E5              PUSH H
00A3  26 02           MVI H,02H
00A5  16 00           MVI D,00H
00A7  6B              MOV L,E
00A8  19              DAD D       ; Add contents of DE to HL
00A9  19              DAD D
00AA  EB              XCHG        ; Exchange contents of
                                     ; H&L with D&E repectively
00AB  1A              LDAX D
00AC  32 01 08        STA 0801H
00AF  13              INX D
00B0  1A              LDAX D
00B1  32 02 08        STA 0802H
00B4  13              INX D
00B5  1A              LDAX D
00B6  32 04 08        STA 0804H
00B9  E1              POP H
00BA  F1              POP PSW
00BB  C9              RET
00BC  F5        LDSTS:  PUSH PSW  ; This subroutine stores
                                     ; min. temp. in RAM after
                                     ; getting the display.
00BD  5D              MOV E,L
00BE  E5              PUSH H
00BF  26 02           MVI H,02H   ; Data from LOOK-UP table
00C1  16 00           MVI D,00H
00C3  6B              MOV L,E
00C4  19              DAD D
00C5  19              DAD D
00C6  EB              XCHG
00C7  1A              LDAX D
00C8  32 13 08        STA 0813H
00CB  13              INX D
00CC  1A              LDAX D
00CD  32 14 08        STA 0814H
00D0  13              INX D
00D1  1A              LDAX D
00D2  32 16 08        STA 0816H
00D5  E1              POP H
00D6  F1              POP PSW
00D7  C9              RET
00D8  F5        CDSTS:  PUSH PSW  ; This subroutine stores present
                                     ; temp. in RAM after getting the
                                     ; display from LOOK-UP table.
00D9  E5              PUSH H
00DA  6F              MOV L,A
00DB  5F              MOV E,A
00DC  26 02           MVI H,02H
00DE  16 00           MVI D,00H
00E0  19              DAD D
00E1  19              DAD D
00E2  EB              XCHG
00E3  1A              LDAX D
00E4  32 27 08        STA 0827H
00E7  13              INX D
00E8  1A              LDAX D
00E9  32 28 08        STA 0828H
00EC  13              INX D
00ED  1A              LDAX D
00EE  32 2A 08        STA 082AH
00F1  E1              POP H
00F2  F1              POP PSW
00F3  C9              RET
00F4  00              NOP
00F5  00              NOP
00F6  00              NOP
00F7  00              NOP
00F8  00              NOP
00F9  00              NOP
00FA  00              NOP
00FB  00              NOP
00FC  00              NOP
00FD  00              NOP
00FE  00              NOP
00FF  00              NOP
0100  01 44 00  BEGIN:  LXI B,0044H ; Loads the display data
                                     ; from monitor ROM to RAM
                                     ; when the system is switched on.
0103  31 FF 09        LXI SP,09FFH
0106  21 A0 01        LXI H,01A0H
0109  11 00 08        LXI D,0800H
010C  C5              PUSH B
010D  E5              PUSH H
010E  D5              PUSH D
010F  7E        D:    MOV A,M
0110  12              STAX D
0111  0B              DCX B
0112  79              MOV A,C
0113  B0              ORA B
0114  CA 1C 01        JZ CHECK
0117  23              INX H
0118  13              INX D
0119  C3 0F 01        JMP D
011C  D1        CHECK:  POP D       ; Check whether all the
                                     ; data are loaded correctly
011D  E1              POP H
011E  C1              POP B
011F  1A        E:    LDAX D
0120  BE              CMP M
0121  C2 00 01        JNZ BEGIN   ; If incorrect loading,
                                     ; Jumps back to BEGIN
0124  0B              DCX B
0125  79              MOV A,C
0126  B0              ORA B
0127  CA 30 01        JZ MAIN     ; If correct loading program
                                     ; jumps to main routine
012A
012A  23              INX H
012B  13              INX D
012C  C3 1F 01        JMP E
012F  00              NOP
0130  CD 78 00  MAIN:   CALL DISM1      ; This routine is for calculation
                                     ; Max. & Min. Temp.and to take input
0133  CD 40 00 INP1: CALL INADC1 ; First reading after switching on
                                     ; the system.
0136  67              MOV H,A
0137  6F              MOV L,A
0138  CD A0 00        CALL HDSTS
013B  CD BC 00        CALL LDSTS
013E  7C              MOV A,H
013F  CD D8 00        CALL CDSTS
0142  CD 78 00        CALL DISM1
0145  00              NOP
0146  00              NOP
0147  00              NOP
0148  CD 40 00  INP2:   CALL INADC1 ; Second reading after
                                     ; switching on the system.
014B  47              MOV B,A
014C  7C              MOV A,H
014D  B8              CMP B
014E  FA 73 01        JM F
0151  68              MOV L,B
0152  CD BC 00        CALL LDSTS
0155
0155  78              MOV A,B
```

```
257  0156  CD D8 00          CALL CDSTS
258  0159  CD 78 00          CALL DISM1
259  015C  CD 40 00  INP3:   CALL INADC ; Third reading after
260                                     ; switching on the system
261  015F  4F                MOV C,A
262  0160  7C                MOV A,H
263  0161  B9                CMP C
264  0162  F2 88 01          JP G
265  0165  61                MOV H,C
266  0166  CD A0 00          CALL HDSTS
267  0169  79                MOV A,C
268  016A  CD D8 00          CALL CDSTS
269  016D  CD 78 00          CALL DISM1
270  0170  C3 5C 01          JMP INP3
271  0173  6C        F:      MOV L,H
272  0174  CD BC 00          CALL LDSTS
273  0177  00                NOP
274  0178  00                NOP
275  0179  00                NOP
276  017A  60                MOV H,B
277  017B  CD A0 00          CALL HDSTS
278  017E  78                MOV A,B
279  017F  CD D8 00          CALL CDSTS
280  0182  CD 78 00          CALL DISM1
281  0185  C3 5C 01          JMP INP3
282  0188  7D        G:      MOV A,L
283  0189  B9                CMP C
284  018A  FA 91 01          JM H
285  018D  69                MOV L,C
286  018E  CD BC 00          CALL LDSTS
287  0191  79        H:      MOV A,C
288  0192  CD D8 00          CALL CDSTS
289  0195  CD 78 00          CALL DISM1
290  0198  C3 5C 01          JMP INP3    ; Main routine ends here
291  019B  00                NOP
292  019C  00                NOP
293  019D  00                NOP
294  019E  00                NOP
295  019F  00                NOP
296  01A0  2A        DIT:    DB 2AH      ; * Display Information
297  01A1  3F                DB 3FH      ; ? Table
298                                      ; Alpha-numeric data
299  01A2  3F                DB 3FH      ; ? display command
300  01A3  2E                DB 2EH      ; . for 1st line
301  01A4  3F                DB 3FH      ; ?
302  01A5  DF                DB DFH      ; ‘
303  01A6  43                DB 43H      ; C
304  01A7  20                DB 20H      ; Space
305  01A8  4D                DB 4DH      ; M
306  01A9  61                DB 61H      ; a
307  01AA  78                DB 78H      ; x
308  01AB  69                DB 69H      ; i
309  01AC  6D                DB 6DH      ; m
310  01AD  75                DB 75H      ; u
311  01AE  6D                DB 6DH      ; m
312  01AF  2A                DB 2AH      ; *
313  01B0  A0                DB A0H
314  01B1  C0                DB C0H
315  01B2  2A                DB 2AH
316  01B3  3F                DB 3FH      ; ?  Alpha-numeric data
317  01B4  3F                DB 3FH      ; ?  data display
318  01B5  2E                DB 2EH      ; .  Command for 2nd
319  01B6  3F                DB 3FH      ; ?  line
320  01B7  DF                DB DFH      ; ‘
321  01B8  43                DB 43H      ; C
322  01B9  20                DB 20H
323  01BA  4D                DB 4DH      ; M
324  01BB  69                DB 69H      ; i
325  01BC  6E                DB 6EH      ; n
326  01BD  69                DB 69H      ; i
327  01BE  6D                DB 6DH      ; m
328  01BF  75                DB 75H      ; u
329  01C0  6D                DB 6DH      ; m
330  01C1  2A                DB 2AH      ; *
331  01C2  2A                DB 2AH      ; *
332  01C3  2A                DB 2AH      ; *
333  01C4  2A                DB 2AH      ; *
334  01C5  2A                DB 2AH      ; *
335  01C6  2A                DB 2AH      ; *
336  01C7  3F                DB 3FH      ; ?  Command for
337  01C8  3F                DB 3FH      ; ?  1st line
338  01C9  2E                DB 2EH      ; .  display
339  01CA  3F                DB 3FH      ; ?
340  01CB  DF                DB DFH      ; ‘
341  01CC  43                DB 43H      ; C
342  01CD  2A                DB 2AH      ; *
343  01CE  2A                DB 2AH      ; *
344  01CF  2A                DB 2AH      ; *
345  01D0  2A                DB 2AH      ; *
346  01D1  2A                DB 2AH      ; *
347  01D2  A0                DB A0H
348  01D3  C0                DB C0H
349  01D4  2A                DB 2AH      ; *
350  01D5  2A                DB 2AH      ; * Command for
351  01D6  2A                DB 2AH      ; * 2nd line
352  01D7  41                DB 41H      ; A display
353  01D8  74                DB 74H      ; t
354  01D9  20                DB 20H
355  01DA  50                DB 50H      ; P
356  01DB  72                DB 72H      ; r
357  01DC  65                DB 65H      ; e
358  01DD  73                DB 73H      ; s
359  01DE  65                DB 65H      ; e
360  01DF  6E                DB 6EH      ; n
361  01E0  74                DB 74H      ; t
362  01E1  2A                DB 2AH      ; *
363  01E2  2A                DB 2AH      ; *
364  01E3  2A                DB 2AH      ; *
365  01E4  2A                DB 2AH      ; *
366  01E5  2A                DB 2AH      ; *
367  01E6  2A                DB 2AH      ; *
368  01E7  00                NOP
369  01E8  CD 16 00  INADC2: CALL DLY2   ;
Subroutine for taking
370                                     ; input from ADC
371  01EB  3E FF             MVI A,FFH   ; INADC1 jumps to this
                                         subroutine
372  01ED  D3 04             OUT 04H     ; FFH is written in ADC,
373                                     ; to initialise conversion
374  01EF  CD 08 00          CALL DLY1   ; Port 04H is the address of ADC
375  01F2  DB 04             IN 04H      ; Data is taken from the ADC,
                                         port 04H
376  01F4  C3 49 00          JMP SUB
377  01F7  00                NOP
378  01F8  00                NOP
379  01F9  76                HLT
380  01FA  END                           ; Program ends here.
```

CROSS REFERENCE TABLE

```
A         000D :  17
AHEAD     006F :  79
B         001B :  28
BACK      0069 :  88
BEGIN     0100 :  9    221
C         002B :  47
CDSTS     00D8 :  242  257  268  279  288
CHECK     011C :  211
CWS       0024 :  67   69   71   73   87
D         010F :  214
DISM1     0078 :  233  243  258  269  280  289
DISM2     0088 :  58
DIT       01A0 :
DLY1      0008 :  38   374
DLY2      0016 :  61   63   111  369
DWS       002A :  80
E         011F :  231
F         0173 :  252
G         0188 :  264
H         0191 :  284
HDSTS     00A0 :  239  266  277
INADC1    0040 :  235  247  259
INADC2    01E8 :  64
INP1      0133 :
INP2      0148 :
INP3      015C :  113  270  281  290
INTLCD    004A :  90   100  376
LDSTS     00BC :  240  254  272  286
MAIN      0130 :  226
STRING    0060 :  84   96   108
LINES ASSEMBLED :  380   ASSEMBLY ERRORS :  0
```

❑

DTMF 8-CHANNEL SWITCHING VIA POWERLINE

O.C. FRANCIS

Using this simple circuit you can switch on/off up to eight appliances remotely via the mains line. 4-bit dual-tone multi-frequency (DTMF) data is sent through the mains line to switch on/off the desired appliances via eight relays.

Push-to-on/off toggle switches S1 through S8 are used to control the appliances. If any switch is in Up position, the encoder sends 'off' signal data and the respective output goes low in the decoder. If the switch is in Down position, the encoder sends 'on' signal data and the respective output goes high to become latched.

Eight 4-bit data words (0000 to 0111) are used to switch off eight appliances. Another eight 4-bit words (1000 to 1111) are used to switch on the appliances. If D bit (MSB) is high it is 'on' signal, and if D bit is low it is 'off' signal. Once switches have been set in on or off position, data words are automatically sent continuously.

In case the power fails, power-on-reset works and all the outputs of IC CD4099 (IC8) go low to switch off all the relays. However, when the power resumes, data is sent automatically again and the respective relays energise. So even after power failure correct devices are switched on/off automatically once the power resumes. The block diagram of powerline DTMF 8-channel switching is shown in Fig. 1.

The circuit

The circuit comprises two units, namely, a DTMF encoder at the controlling end and a DTMF decoder at the other end where the appliances are located. The two units are connected via phase (L), neutral (N), and earth (E) wires of the AC mains line, with the AC phase being the same.

The encoder. The encoder circuit (see Fig. 2) comprises DTMF tone generator IC UM95089 (IC2), dual binary counter IC CD4520 (IC3), and two 16-channel multiplexer ICs CD4067 (IC4 and IC5).

The mains frequency of 50 Hz is fed to pin 2 of dual binary counter IC3(A) via transformer X1 secondary. The D output (MSB) of IC3(A) is connected to strobe pin 10 of IC3(B). The 4-bit binary output of IC3(B) is fed to the two 16-channel multiplexers (IC4 and IC5) as address input. The 16 decoded outputs of IC4 are connected to the column pins of the DTMF tone generator (IC2), while the outputs of IC5 are connected via switches S1 through S8 to the row pins of IC2. Inputs Y0 to Y15 of IC5 are connected to switches S1 through S8 as shown in Fig. 2.

DTMF tone generator IC UM95089 (IC2) has four row pins and four column pins, each of which is associated with a specific frequency. When any row pin is shorted to any column pin, the associated dual-frequency tone is generated. At any given instant, the shorting

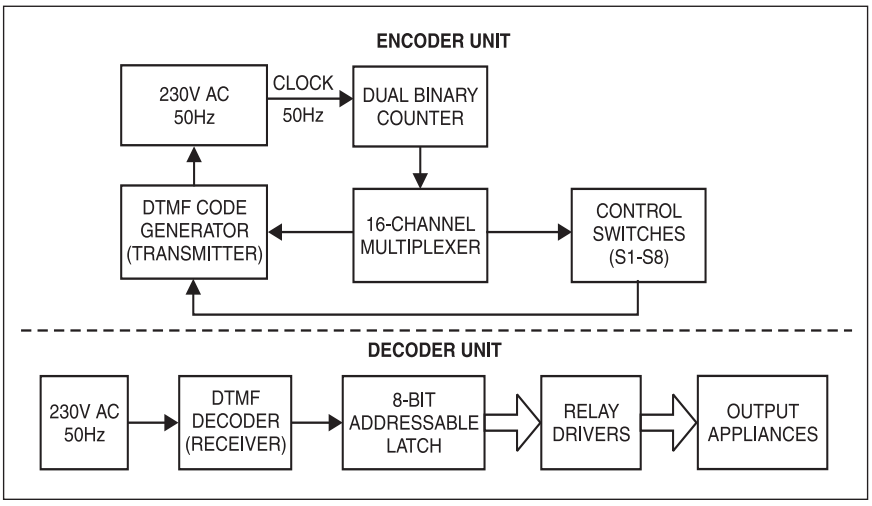


Fig. 1: Block diagram of powerline DTMF 8-channel switching

PARTS LIST

Semiconductors:

IC1, IC6	- +5V 7805 regulator
IC2	- UM95089 DTMF tone generator
IC3	- CD4520 dual-binary counter
IC4, IC5	- CD4067 16-channel multiplexer/demultiplexer
IC7	- MT8870 DTMF receiver
IC8	- CD4099 8-bit addressable latch
D1-D12	- 1N4007 rectifier diodes
LED	- 5mm red LED
T1-T9	- BC548 npn transistor

Resistors (all ¼-watt, ±5% carbon, unless stated otherwise):

R1, R2, R9-R16	- 1-kilo-ohm
R3	- 68-kilo-ohm
R4, R6	- 100-kilo-ohm
R5	- 120-kilo-ohm
R7, R8	- 10-kilo-ohm

Capacitors:

C1, C3	- 0.1μF polyester
C2, C4	- 1000μF, 25V electrolytic
C7	- 10μF, 25V electrolytic
C5, C6	- 0.1μF ceramic disk
C8, C9	- 100μF, 25V electrolytic

Miscellaneous:

RL1-RL8	- 9V, 150-ohm, 1C/O relays
S1-S8	- Push-to-on/off switch
X1-X2	- 230V AC primary to 9V-0-9V, 500mA secondary transformers
X_{tal1}-X_{tal2}	- 3.5795MHz crystal

of a row and column takes place as per the address present at pins 10 through 14 of IC4 and IC5 and position of switches S1 through S8. The selected column is connected to selected row via pins 1 of IC4 and IC5. A total of 16 dual-frequency tones, one for each combination, are thus possible. The tone output from IC2 is indicated by glowing of a red LED. For each count, one row and one column is connected together via IC4 and IC5 and a DTMF tone is generated and superimposed on to the neutral mains.

The decoder. The decoder circuit (see Fig. 3) comprises MT8870 DTMF receiver (IC7), CD4099 8-bit addressable latch (IC8), and relay drivers.

DTMF tone generated by the coder and transmitted through the mains wires is received by DTMF-to-binary decoder IC MT8870 (IC7) for con-

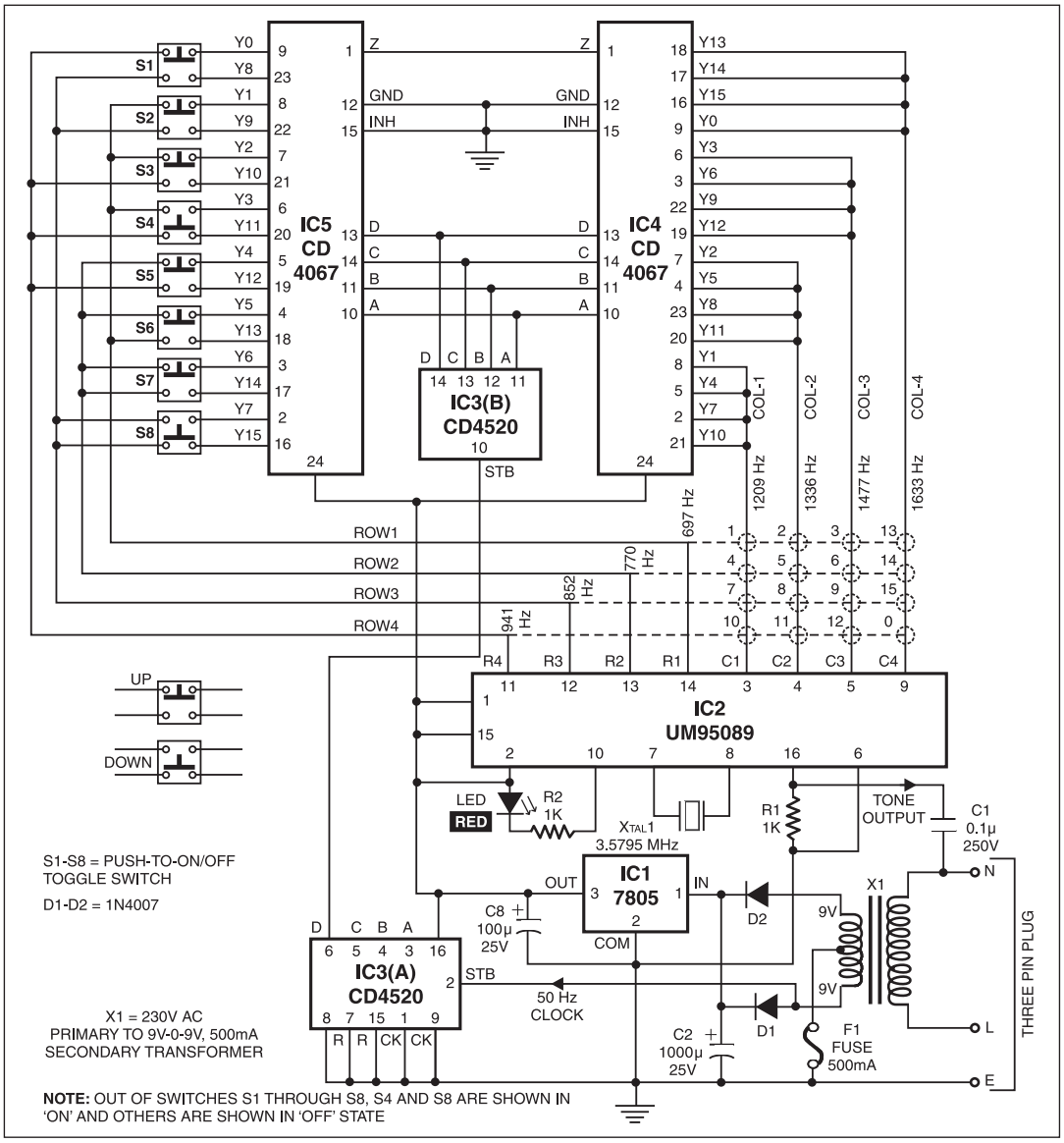


Fig. 2: Encoder circuit

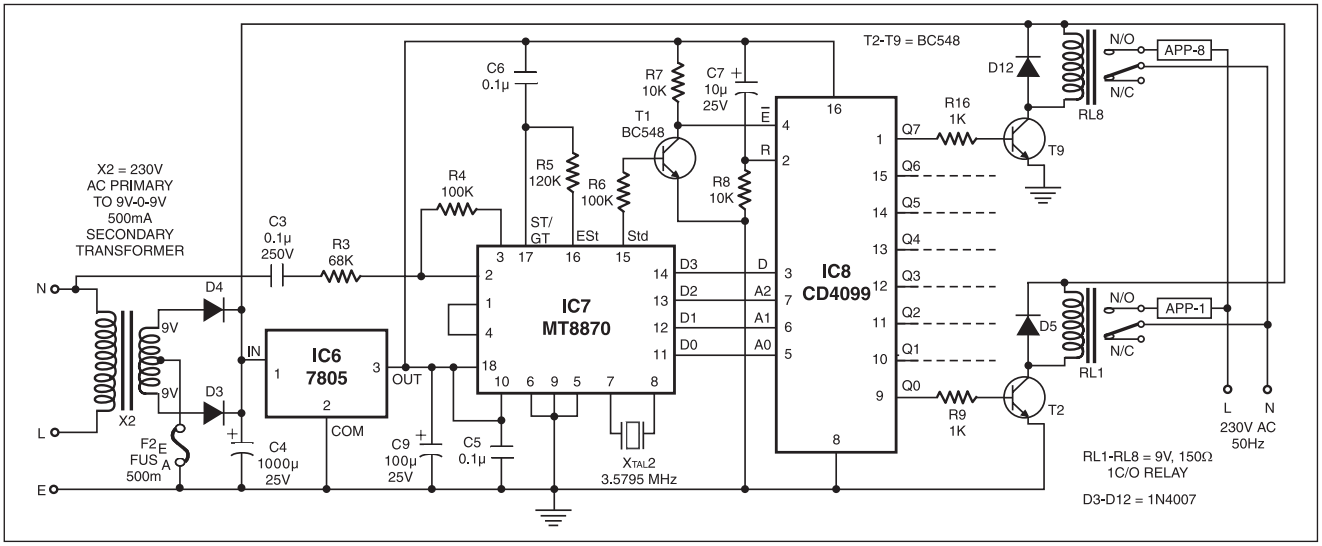


Fig. 3: Decoder circuit

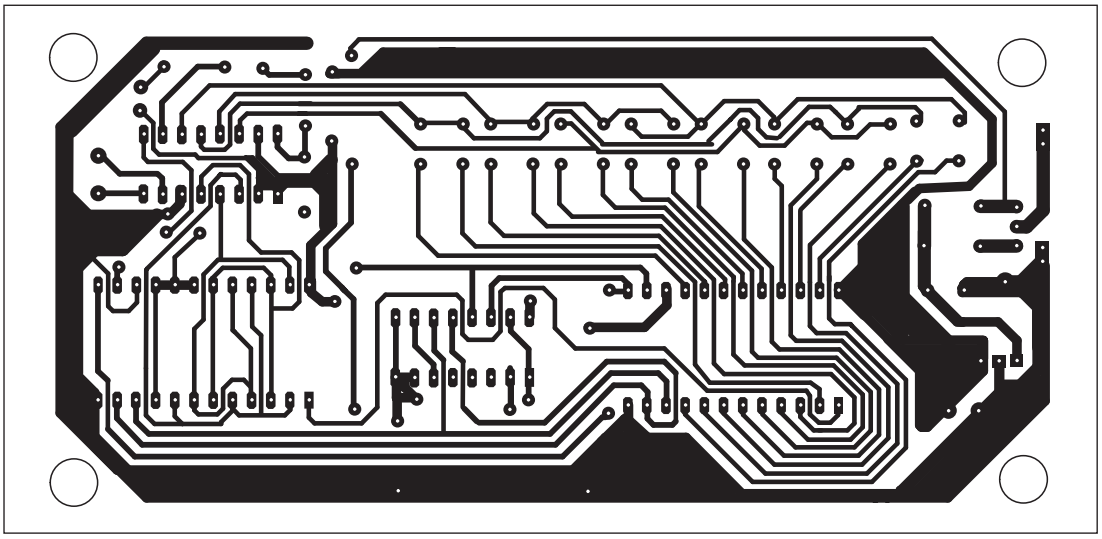


Fig. 4: Actual-size, single-side PCB layout for the encoder circuit

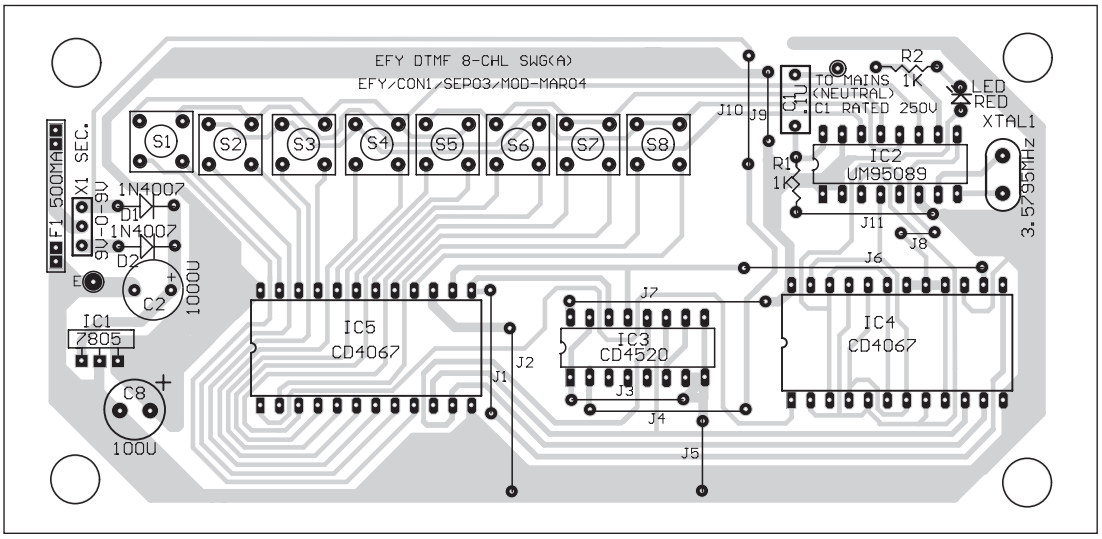


Fig. 6: Component layout for the PCB in Fig. 4

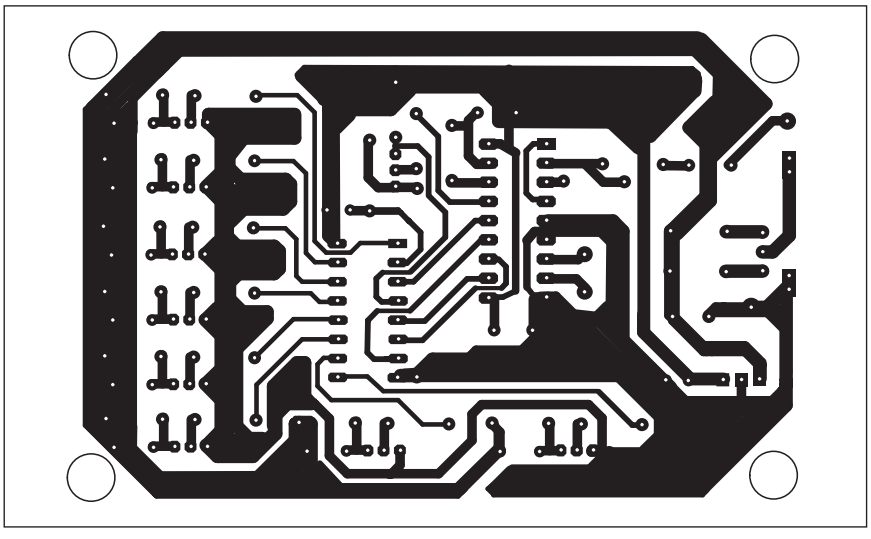


Fig. 5: Actual-size, single-side PCB layout for the decoder circuit

version into the corresponding 4-bit binary output at pins 11 through 14. This output is fed to 8-bit addressable latch CD4099 (IC8).

If D bit is high, one of the outputs of IC8 goes high and latches. If D bit is low, one of the outputs of IC8 will be low. The Q0 through Q7 outputs of IC8 are fed to, via 1k resistors, transistors T2 through T9 for driving relays RL1 through RL8 (9V, 150-ohm, single-changeover) to switch on/off the selected devices.

Working

Using the keyboard mounted on the coder's panel, you can easily switch on/off the desired appliances. The on/off

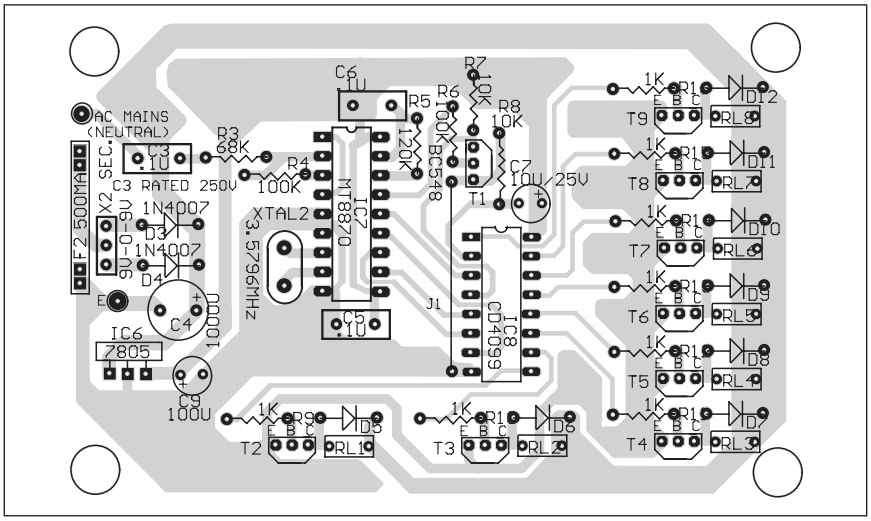


Fig. 7: Component layout for the PCB in Fig. 5

position of switches on the keyboard indicates whether the remote appliance is on or off.

Let's assume that you wish to switch on appliance Nos 4 and 8 connected across relays RL4 and RL8, respectively.

When you push switches S4 and S8 to Down position, these produce codes 1011 and 1111, respectively. This encoded data is superimposed to the mains neutral via capacitor C1. After receiving this data, IC7 places the corresponding binary numbers 1011 and 1111 at its output as well as the input of IC8. Since the D input (pin 3) of IC8 is high, its Q3 and Q7 outputs go high and latch. As a result, relays RL4 and RL8 energise to turn on appliance Nos 4 and 8. All other outputs will be low, keeping the remaining six appliances off.

When IC7 receives data, its delayed steering output goes high and transistor T1 conducts to make pin 4 of IC8 low. This enables IC8 to receive the data.

Power supply

The circuit, except relay driver, works on 5V. The relay driver works on 9V. Transformers X1 and X2, along with 3-pin voltage regulators IC1 and IC6 (IC 7805) and rectifier diodes D1 through D4, provide a regulated power supply of 5V. The output voltage from rectifier diodes D3 and D4 is used to drive the relays. To improve the current-handling capacity and to prevent thermal runway, regulators are provided with heat-sinks.

Construction

The entire circuit can be easily assembled on a general-purpose PCB board. However, actual-size, single-side PCBs for the encoder (Fig. 2) and the decoder (Fig. 3) are shown in Figs 4 and 5, respectively. The component layouts for the PCBs in Figs 4 and 5 are shown in Figs 6 and 7, respectively.

Cautions. From the maintenance point of view, it is advisable to use IC bases. While soldering the crystal, don't heat it for too long as it may get damaged. Make sure that live (L), neutral (N), and earth (E) wires of the mains line are not interchanged in any of the two units. The encoder circuit can be housed in a box with switches S1 through S8 mounted on the front panel. Similarly, the coder circuit can be housed in a box, with relays to control appliances mounted inside the box.

❑

Readers' comments:

Q1. I have the following doubts:
Is it safe to connect the tone out to the mains through a capacitor? Is it possible to connect it to the secondary winding of the transformer (in the transmitter and the receiver)?

Q2. The fuse in the diagram is connected between the centre tap of the secondary winding and ground. Is it not safer to insert a fuse on the live wire (same in the receiver side)?

Q3. The use of IC ULN2803 at the output of 4099 could avoid the use of eight transistors, resistors, and back-emf diodes.

Q4. Tone generator ICs normally require about 1/10-second gap between key depressions. Does IC3 (CD4520) reduce the mains frequency below this count when multiplexing? Even if it does, IC3(b) (4520) sends a BCD count to 16-way switches without any time delay between consecutive DTMF pulses. In this case the tone generator gets a row and column shorted without a gap between consecutive key depressions. If so, how will tone generator IC 5089 emit a tone?

K.S. Sankar
Chief Executive, Mostek Electronics
Chennai

The author, O.C Francis, replies:

A1. In normal condition it is safe to connect the tone to neutral through a capacitor. C1 and C3 may be 1kV type. It is not possible to connect the tone out to secondary winding, since the secondary winding is connected to Stb input of counter IC3(a).

A2. The fuse on the earth may be fast blow type. If necessary, an additional fuse can be inserted on the live wire.

A3. For relay driver IC ULN2803, refer to construction project 'Multiple Device Switching Using PC's Parallel Port' on page 52 in EFY's October issue. Each output of ULN2803 has an average current of 150 mA and an internal diode.

A5. For a reliable tone reception, 100ms tone duration is necessary. IC3 divides 50 Hz by 16 to produce 320ms tone. To have a gap between two tones, disconnect the INH inputs of IC4 and IC5 from GND and connect both INH inputs to D out of IC3(a).

TWO-IN-ONE STEREO AMPLIFIER

SANI THEO

Here is a circuit of mains-cum-battery operated stereo amplifier built around a readily available stereo tape deck mechanism. In addition, a simple built-in FM receiver circuit is incorporated. This FM circuit can be replaced with a readily available prewired FM plate through flick of a switch. This circuit can also be used as a car stereo, powered from a 12V car battery. Compactness and simplicity are its main features.

Block diagram

The block diagram of the two-in-one stereo amplifier system is shown in Fig. 1. The system can be subdivided into four parts:

1. Power supply
2. FM receiver
3. Audio preamplifier
4. Audio power amplifier

Power supply. The 230V AC mains power supply is stepped down using a step-down transformer and fed to the inputs of two regulated ICs (12V and 6V) after rectification by a bridge rectifier and smoothing by a capacitor. The 12V supply drives the audio power amplifier and FM circuits, while the 6V supply feeds the preamplifier circuit.

FM receiver. The FM receiver amplifies FM signals in the 88MHz-108MHz range. Provision has been made in the circuit to switch over to an FM kit through the slide switch.

Audio preamplifier. The preamplifier amplifies the signals received from the stereo heads or the FM receiver. It raises the amplitude of the input signal to a certain level required for the audio power amplifier. It gets power supply from regulator IC 7806. The preamplifier used here is based on IC LA3161 from Sanyo.

Power amplifier. The audio power amplifier further amplifies the signals coming out from the stereo preamplifier. The power amplifier used here is based on IC LA4440 from Sanyo. It gets 12V supply from the 7812 regulated IC.

Circuit description

The circuit uses ICs LA3161 and LA4440 (IC3 and IC4, respectively) popularly used in car stereos. The pin configurations of both the ICs are shown in Fig. 2, while their functions are given Table I and Table II, respectively.

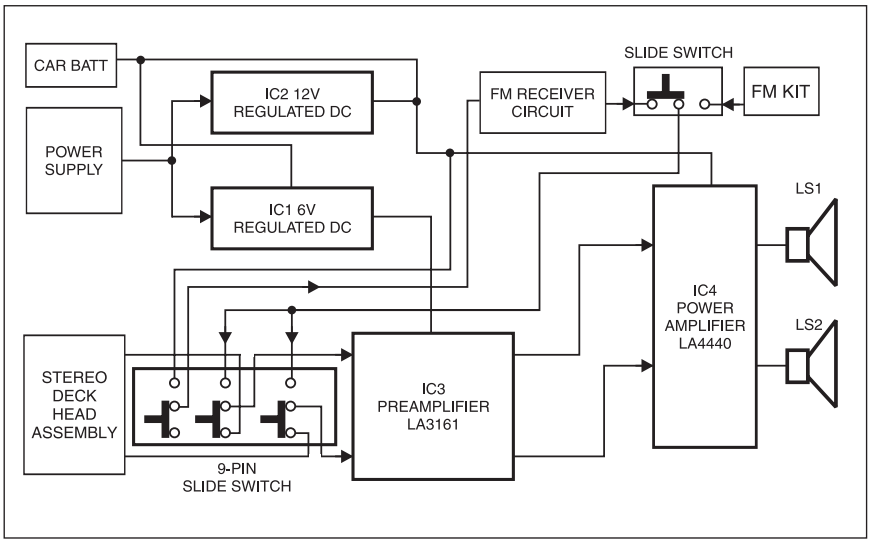


Fig. 1: Block diagram of two-in-one stereo amplifier

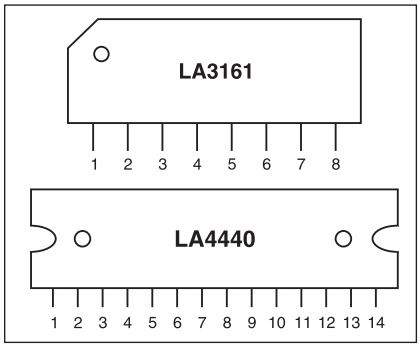


Fig. 2: Pin configurations of LA3161 and LA4440

As mentioned earlier, the power supply for the circuit is derived from AC mains. It can also be fed from the 12V car battery through a 2-pin connector (CON) as shown in Fig. 3. LED2 glows when the power is applied to the circuit. Preamplifier IC3 gets 6V supply at pin 4 through one-kilo-ohm resistor R2. Power amplifier IC4 gets 12V supply directly at its pin 11. The FM receiver gets 12V via switch S2.

Audio preamplifier. IC LA3161 (IC3) is a built-in 2-channel, low-noise preamplifier that requires few external components. It is an 8-pin IC with power dissipation of 200 mW. Typically, its current consumption is 6.5 mA, which can go up to a maximum of 8 mA.

One of the two outputs of the playback head is fed to pin 1 and the other output is fed to pin 8 of IC3 through selector switch S2. The FM receiver output is also fed through this switch to pins 1 and 8 of IC3. One can use a 3-pole, 2-way slide or push switch for S2.

The outputs of IC3 at its pins 3 and 6 are connected to volume control potentiometers VR1 and VR2 (each 470-kilo-ohm) via capacitors C12 and C13 (each 2.2μF, 25V), respectively. The terminals other than the common terminal of VR1 and VR2 are connected to power amplifier IC4

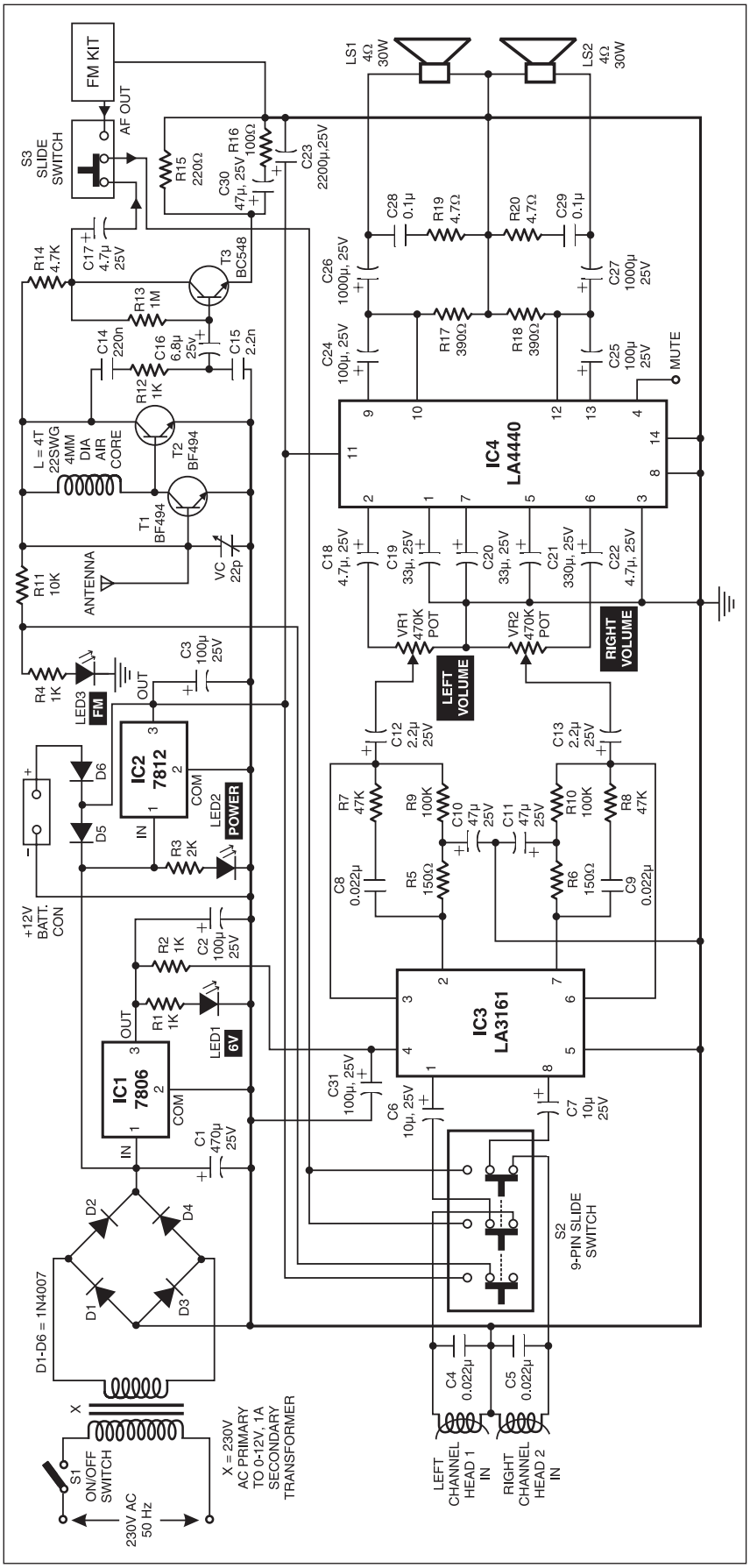


Fig. 3: The circuit of two-in-one stereo amplifier

PARTS LIST

Semiconductors:

IC1	- 7806 regulator
IC2	- 7812 regulator
IC3	- LA3161 preamplifier
IC4	- LA4440 audio power amplifier
T1, T2	- BF494 transistor
T3	- BC548 transistor
D1-D6	- 1N4007 rectifier diode
LED1	- Yellow, preamp indicator
LED2	- Red, power indicator
LED3	- Green, FM indicator

Resistors (all ¼-watt, ±5% carbon, unless stated otherwise):

R1, R2, R4, R12	- 1-kilo-ohm
R11	- 10-kilo-ohm
R3	- 2-kilo-ohm
R5, R6	- 150-ohm
R7, R8	- 47-kilo-ohm
R9, R10	- 100-kilo-ohm
R13	- 1-mega-ohm
R14	- 4.7-kilo-ohm
R15	- 220-ohm
R16	- 100-ohm
R17, R18	- 390-ohm
R19, R20	- 4.7-ohm
VR1, VR2	- 470-kilo-ohm potmeter

Capacitors:

C1	- 470µF, 25V electrolytic
C2, C3, C24, C25, C31	- 100µF, 25V electrolytic
C4, C5, C8, C9	- 0.022µF polyester
C6, C7	- 10µF, 25V electrolytic
C10, C11, C30	- 47µF, 25V electrolytic
C12, C13	- 2.2µF, 25V electrolytic
C14	- 220nF ceramic
C15	- 2.2nF ceramic
C16	- 6.8µF, 25V electrolytic
C17,C18,C22	- 4.7µF, 25V electrolytic
C19, C20	- 33µF, 25V electrolytic
C21	- 330µF, 25V electrolytic
C23	- 2200µF, 25V electrolytic
C26, C27	- 1000µF, 25V electrolytic
C28, C29	- 0.1µF polyester

Miscellaneous:

X	- 230 AC primary to 0-12V, 1A secondary transformer
S1	- SPST on/off switch
S2	- 9-pin slide switch
S3	- SPDT slide switch
LS1, LS2	- 4-ohm, 30W loudspeaker
Head1,Head2	- Stereo deck head
Antenna	- Whip antenna
CON	- 2-pin connector

at left and right input pins 2 and 6 through capacitors C18 and C22 (each 4.7µF, 25V), respectively. The glowing of LED1 indicates that the power supply to the preamplifier is on.

R5 and R6 are negative feedback resistors. Increasing their value decreases the voltage gain. R9 and R10 are DC feed-

TABLE I
Pin Details of Preamplifier IC LA3161

Pin	Function	Voltage
1	Input from audio head channel 1	0.6
2	Negative feedback for channel 1	0.6
3.	Output of channel 1	2.1
4	Positive supply voltage	6.0
5	Ground	0
6	Output of channel 2	2.1
7	Negative feedback for channel 2	0.6
8	Input from audio head channel 2	0.6

TABLE II
Pin Details of Power Amplifier IC LA4440

Pin	Function	Voltage
1	Negative feedback (NF1)	1.0
2	Input for channel 1	0.0
3	Preamplifier ground 1	0.0
4	Audio muting input (+ve)	—
5	Decoupling capacitor	12.0
6	Input for channel 2	0.0
7	Negative feedback (NF2)	1.3
8	Power amplifier GND 2	0.0
9	Bootstrap operation for channel 2	11.21
10	Audio output channel 2	6.8
11	Power supply (+ve)	12.0
12	Audio output channel 1	6.8
13	Bootstrap operation for channel 1	11.21
14	Power amplifier GND 1	0.0

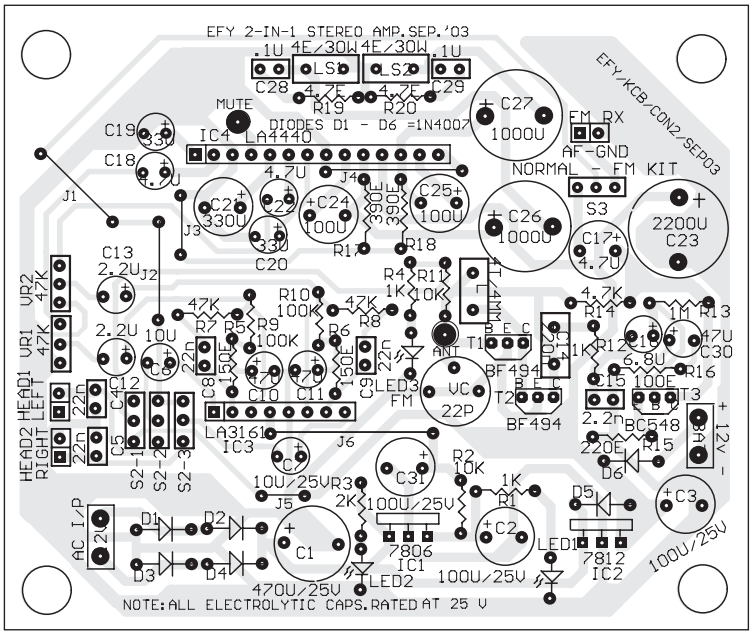


Fig. 5: Component layout for the PCB

back resistors. If the values of these resistors are very small or very high, the output decreases.

Audio power amplifier. Power amplifier IC LA4440 (IC4) has two built-in channels with built-in protection against thermal over voltage, surge voltage, and pin-to-pin short. Typical power dissipation and minimum quiescent current are 15W and 100 mA, respectively. It requires a heat-sink of 400 grams to prevent overheating and thermal runaway. The function and effect of change in values of components at various pins of LA4440 amplifier are detailed below.

Capacitors C19 and C20 (each 33μF, 25V) at pins 1 and 7, respectively, are feedback capacitors. The low cut-off frequency depends on these capacitors. If the capacitor values are increased, the starting time is delayed.

Capacitor C21 (330μF, 25V) is a decoupling capacitor used for ripple filtering. Capacitor C23 (2200μF, 25V) is a power source capacitor.

Capacitors C24 and C25 of 100μF, 25V are bootstrap capacitors. If their values are decreased, the output at low frequencies will be reduced.

Capacitors C26 and C27 of 1000μF, 25V are output capacitors. The low cut-off frequencies depend on these capacitors.

Output capacitors C28 and C29 (each 0.1 μF) and resistors R19 and R20 (each 4.7 ohms) are used to prevent oscillation. C28 and C29 are polyester capacitors, which offer stability against temperature variation and varying frequency response.

The mute facility can be added to the circuit by applying a positive voltage of greater than 6V to pin 4 of IC LA4440 through a integerator circuit comprising a resistor of 1 kilo-ohm and a capacitor of 47μF. Loudspeakers LS1 and LS2 are rated for 4-ohm, 30W each.

FM receiver

The FM receiver operates off regulated 12V DC via switch S2. It comprises three transistors (two BF494 and one BC548) and other associated components.

The output of the FM receiver at the collector of BC548 is fed to the preamplifier via capacitor C17 (4.7μF, 25V) and selector switch S2. By tuning 22pF trimmer capacitor VC, the receiver can receive the transmitted frequency. With a good whip antenna, it can receive even weak FM signals. LED3 glows when FM mode is selected.

The FM circuit receives power only when FM mode is selected through slide switch S2. This prevents continuous drain of current from the power supply. The FM receiver works fine up to 1 km range with the 'Long Range FM transmitter' published in EFY Dec. 99 or Electronics Projects Vol. 20.

Alternatively, you can use a readily available FM receiver kit through slide switch S3. FM kits using Philips IC TEA5591 and Sony IC CXA1019/CXA 1692 are easily available in the market for around Rs 50 to Rs 100. These FM kits have a very high sensitivity and are capable of receiving programmes from transmitting stations located many kilometres away. Note that while connecting the FM kit, its ground terminal should be made common with ground of the amplifier circuit.

Fig. 4 shows the actual-size, solder-side PCB for the stereo amplifier with its component layout in Fig. 5.

Operation

1. After you're done with connections, switch on the power supply using switch S1. Put an audio cassette in the stereo deck and press Play button. Now adjust volume controls VR1 and VR2.

2. If the audio output is low, in spite of increasing the volume controls to the maximum, adjust the spring loaded screw (provided near the deck head) using a screwdriver to increase the volume.

3. For FM operation, select FM mode

using switch 2. Using a screwdriver, tune the trimmer (VC) until a clear audio signal is received.

4. If the inbuilt FM receiver cannot receive the FM broadcast station, connect an FM kit and tune the trimmer provided on the kit. For further operation of the FM kit, refer to the operation manual provided by the manufacturer.

Note. Keep the number of jumpers to a minimum in order to prevent hum and other unwanted noise. Using a good shielded wire from the stereo head to the input of the preamplifier greatly reduces the noise. ❑

Readers' comments:

Q. I have successfully constructed the 'Two-in-One Stereo Amplifier' circuit. Please tell me which is the audio input line of this circuit so that I can connect the audio output of my CD player to this circuit?

R. Saravanan
Thanjavur, T.N.

The author, Sani Theo, replies:

A. I thank Mr Saravanan for showing interest in my circuit. In the circuit, the audio input is fed through switch S2 (9-pin slide switch). He can connect the audio output line of his CD player through this switch either by disconnecting the deck head inputs (right and left) or by using a 12-way DIL switch in place of the 9-pin slide switch. The latter is a better option in case he wants to use the stereo deck, FM and CD player as well.

MULTI-FEATURE EMERGENCY LIGHT

VINOD C.M.

Here is an interesting circuit that allows fluorescent tubes to glow at a voltage as low as 80V. What's more, using this circuit the tubes can be operated off a 12V battery. This feature is not found even in electronic ballasts. The circuit also charges the battery (12V, 15AH) at a constant rate using 80V-260V mains supply. It costs less than electronic ballasts or inverters with charging units.

Block diagram

Fig. 1 shows the block diagram of the multifeature emergency light circuit for fluorescent tube. The mains AC powers the range-selection section and the emergency light and charging section. The AC input range for the range-selection section is selected with the help of a comparator. The emergency light circuit works off either AC mains or DC supply.

Circuit description

The AC mains voltage is sampled by transformer X2. Its output voltage is rectified and filtered by capacitor C2. The filtered DC output voltage is supplied to the non-inverting input of operational amplifier CA3140A (IC2) through a potential divider comprising resistor R1 and preset VR1. The reference voltage at the inverting input (pin 2) of IC2 is provided through regulator IC 7809 (IC4) and a potential divider comprising resistors R2 and R3.

Whenever the AC mains voltage is greater than 115V, the output of the voltage comparator (IC2) goes high and relay RL1 energises. As a result, the 230V terminal on primary of transformer X1 is connected to the AC mains. Transformer X1 acts as a 230V primary to 24V secondary stepdown transformer. Therefore the secondary voltage of transformer X1 is 24V AC.

If the mains

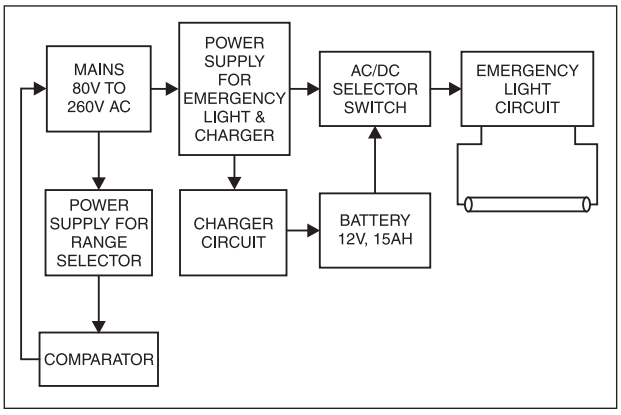


Fig. 1: Block diagram of multi-feature choke for fluorescent tube

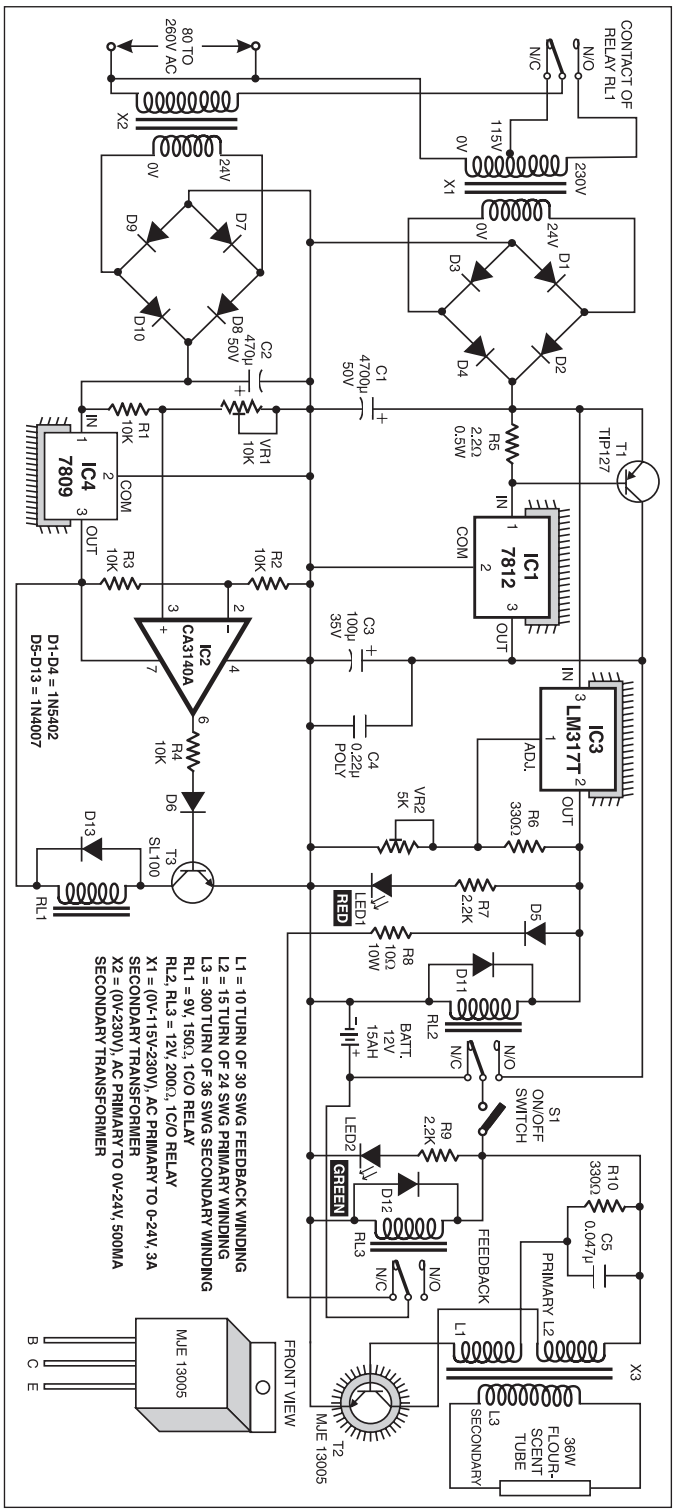


Fig. 2: Circuit diagram of multi-feature emergency light

PARTS LIST

Semiconductors:

IC1	- LM7812 12V regulator
IC2	- CA3140A operational amplifier
IC3	- LM317T adjustable voltage regulator
IC4	- LM7809 9V regulator
T1	- TIP127 pnp transistor
T2	- MJE13005 npn power transistor
T3	- SL100 npn transistor
D1-D4	- 1N5402 rectifier diode
D5-D13	- 1N4007 rectifier diode
LED1	- Red LED
LED2	- Green LED

Resistors (all ¼-watt, ±5% carbon, unless stated otherwise):

R1-R4	- 10-kilo-ohm
R5	- 2.2-ohm, 0.5W
R6, R10	- 330-ohm
R7, R9	- 2.2-kilo-ohm
R8	- 10-ohm, 10W

Capacitors:

C1	- 4700μF, 50V electrolytic
C2	- 470μF, 50V electrolytic
C3	- 100μF, 35V electrolytic
C4	- 0.22μF, 100V polyester
C5	- 0.047μF, 100V polyester

Miscellaneous:

X1	- 0V-115V-230V AC primary to 0-24V, 3A secondary transformer
X2	- 0V-230V AC primary to 0-24V, 500mA secondary transformer
X3	- 12V, 40W inverter transformer (refer Fig. 3)
RL1	- 9V, 150-ohm, 1C/O relay
RL2, RL3	- 12V, 200-ohm, 1C/O relay transformers

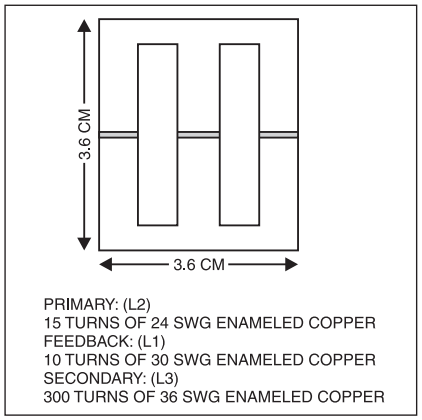


Fig. 3: Details of transformer X3

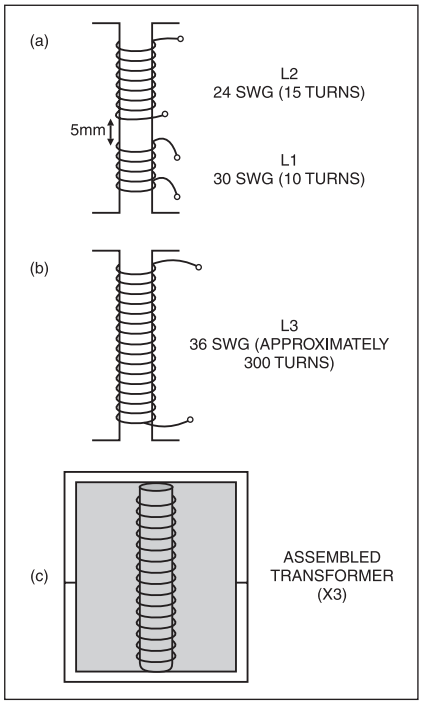


Fig. 4: Steps for fabrication of transformer X3

and voltage regulator IC 7812 (IC1) via resistor R5.

The regulator 7812 can deliver about 500mA current to the emergency light circuit. This current is insufficient for proper working of the emergency light circuit. Transistor T1 provides a bypass path for a current above 300 mA. The regulated voltage is supplied to the emergency light circuit.

Since the regulator ICs have internal over-voltage protection, they turn off when the input voltage is above 35V, i.e. when the mains voltage reaches about 260V in this circuit. Therefore the input voltage range of the regulator is 80V to 260V and the output voltage is constant at 12V DC.

Relay RL2 selects the source as the output voltage of regulator 7812 when AC power is present (indicated by LED1). Relay RL3 disconnects the supply to the charger when the emergency light circuit is working with the regulator circuit's output (indicated by LED2). Otherwise, it may lead to overloading of transformer X1 as the transformer will have to supply both the emergency light circuit and the charger section simultaneously. Switch S1 (off position) is used to charge the battery when emergency light circuit is not in use.

The charger section consists of adjustable voltage regulator LM317T (IC3) and current-limiting resistor R8. The output voltage is below 115V, the comparator output goes low and relay RL1 is de-energised. Therefore 115V tapping of primary transformer X1 gets connected to the AC mains supply. Transformer X1 acts as a 115V primary to 24V stepdown transformer. (The ratio of primary to secondary turns doubles when you use the centre tap on primary side of a transformer.)

Thus an AC voltage greater than 12V is obtained under AC mains voltage supply ranging from 80V to 230V. This voltage is rectified by a full-wave rectifier, filtered by C1, and given to the emitter of transistor T1

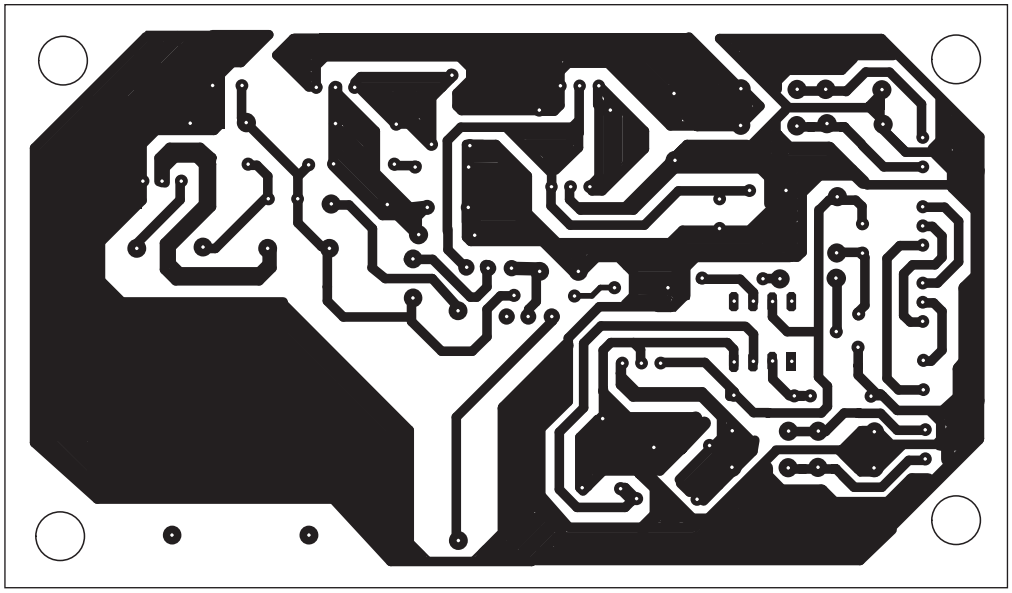


Fig. 5: Actual-size, single-side PCB layout of multi-feature choke for fluorescent tube

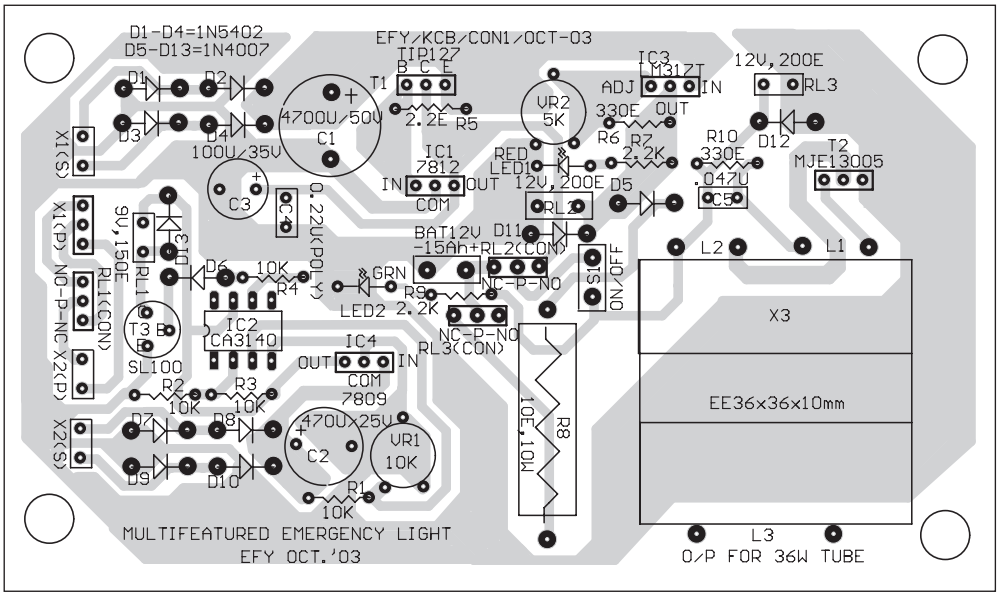


Fig. 6: Component layout for the PCB

voltage of the charger unit is adjusted with the help of preset VR2. Diode D5 prevents energisation of relay RL2 due to battery voltage.

The emergency light circuit is basically an inverter circuit. It comprises transistor T2, resistor R10, capacitor C5, and ferrite transformer X3. It operates at a very high frequency (above 20 kHz), which enables transformer X3 to step up the primary voltage from a 12V source to around 700V erratic pulses (peak-to-peak) in the secondary (due to ionisation inside the fluorescent tube). Here we've used a ferrite core transformer so that core losses are low. Winding details of transformer X3 are given in Fig. 3.

The steps for winding transformer X3 are as follows:

1. Take bobbin of the ferrite core transformer and wind coils L1 and L2 side by side as shown in Fig. 4(a).

2. Cover the coils with a paper insulator and wind coil L3 over it. After 150 turns are completed, cover this layer (Fig. 4(b)) with paper insulator.

3. Cover the final secondary winding having 300 turns with the paper insulator and insert the ferrite core in the bobbin as shown in Fig. 4(c). Finally, fix the core using insulation tape.

Construction and testing

The actual-size, single-side PCB for the multifeatured emergency light is shown in Fig. 5 and its component layout in Fig. 6.

The circuit can be constructed on a printed circuit board. A suitable heat-sink must be used for IC1, IC3, and IC4 and transistors T1 and T2.

A variac is used for initial setting of the unit. Set the output voltage of the variac to 115V and adjust preset VR1 carefully such that relay RL1 energises. Set the output voltage of the charger to about 13.5V using preset VR2. Now the unit is ready for use.

Note. For better results, use a 36W thin tube as it consumes less power.

❑

MULTIPLE DEVICE SWITCHING USING PC's PARALLEL PORT

R. KARTHICK

The PC parallel port is a powerful platform, though expensive, for implementing projects dealing with the control of real-world peripherals. It can be used to control the printer as well as household and other electrical appliances. The computer program included as part of the project controls the relays through the interface circuit, which, in turn, switch the appliances 'on' or 'off'.

The parallel port has 12 outputs including 8 data lines and 4 control lines. The circuit described here can be used to control up to 255 electrical appliances using only eight data output lines from the parallel port. Besides, the software program allows the users to know the current status of the devices.

Block diagram

The block diagram in Fig. 1 shows the main components of the system for switching multiple devices using PC. The control command to switch on/off the appliances is given through the keyboard. The software program scans the input and, as per the input command, the data is available at the parallel port.

Out of eight data output lines from the PC, bits D0 through D3 serve as address lines for decoders 2 and 3, which, in turn control the devices to be switched on/off. Bits D4 through D7 are used as address lines for decoder 1, whose outputs are used as enable signals for decoders 2 and 3 as shown in Fig. 1.

The output of decoders 2 and 3 drive D-type flip-flops, which, in turn, control the relays via relay drivers inside IC ULN2803.

The parallel port

The parallel port or line printer terminal (LPT) port terminates into a 25-pin D-type female connector available at the back of your PC. A basic IBM PC usually comes with one or two LPT ports. The original parallel port, called standard parallel port (SPP), is a bundle of three ports (or registers), namely, data port, status port, and control port. Pins 2 through 9 form the 8-bit data output port. This port is purely a write-only port. This means it can be used only to output some data through it. Pins 1, 14, 16, and 17 form the control port, which is capable of reading/writing. Pins 10 through 13 and pin 15 together form the status port. The status port is a read-only port.

The base address of the first parallel port (LPT1) is 0378 in hexadecimal (hex) notation (or 888 in decimal notation). The base address of the second parallel port (LPT2) is 0278 (hex). In this project, we've used only LPT1.

PARTS LIST

Semiconductors:

IC1, IC2, IC3 - 74LS154 1-of-16 decoder
IC4, IC5, IC6 - 74LS05 inverter
IC7-IC14 - 74LS74 D-type flip/flop
IC15, IC16 - ULN2803 octal Darlington array driver

Miscellaneous:

Power supply - 5V regulated DC, 12V regulated DC
Relay - 12V, 200-ohm, 1C/O SPDT

Circuit description

The circuit comprises decoder, inverter, latch, and relay driver sections. The circuit, excluding relay drivers and relays, is powered by a 5V DC regulated supply. Relay drivers and relays are driven by a 12V DC regulated supply. Each relay is rated 12V, 200-ohm.

The interface circuit for switching on/off 16 devices is shown in Fig. 2. For more than 16 loads, you can add more ICs in a similar way as shown in this circuit. IC 74LS154 is a 24-pin, 4-to-16-line decoder IC. It accepts four inputs and provides 16 outputs. Input address lines A1 through A4 (to pins 20 through 23) of IC1 and IC2 (IC 74LS154) each are connected to data bits D0 through D3 of the data lines of the computer parallel port.

In the circuit, only pins 2 through 9 of the parallel port 378 (hex) are used. D4 through D7 form the address-select. Pins

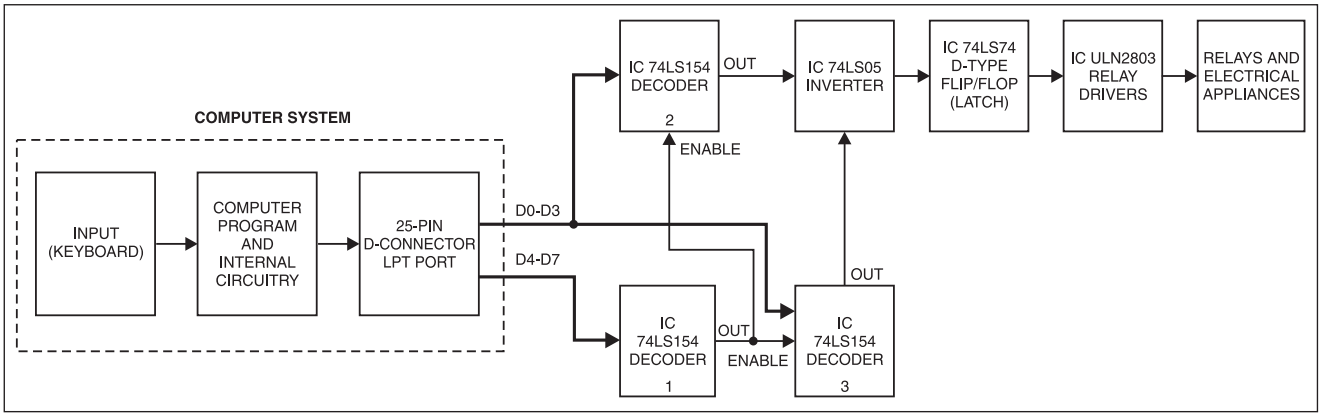


Fig. 1: Block diagram of the system for multiple device switching using PC's parallel port

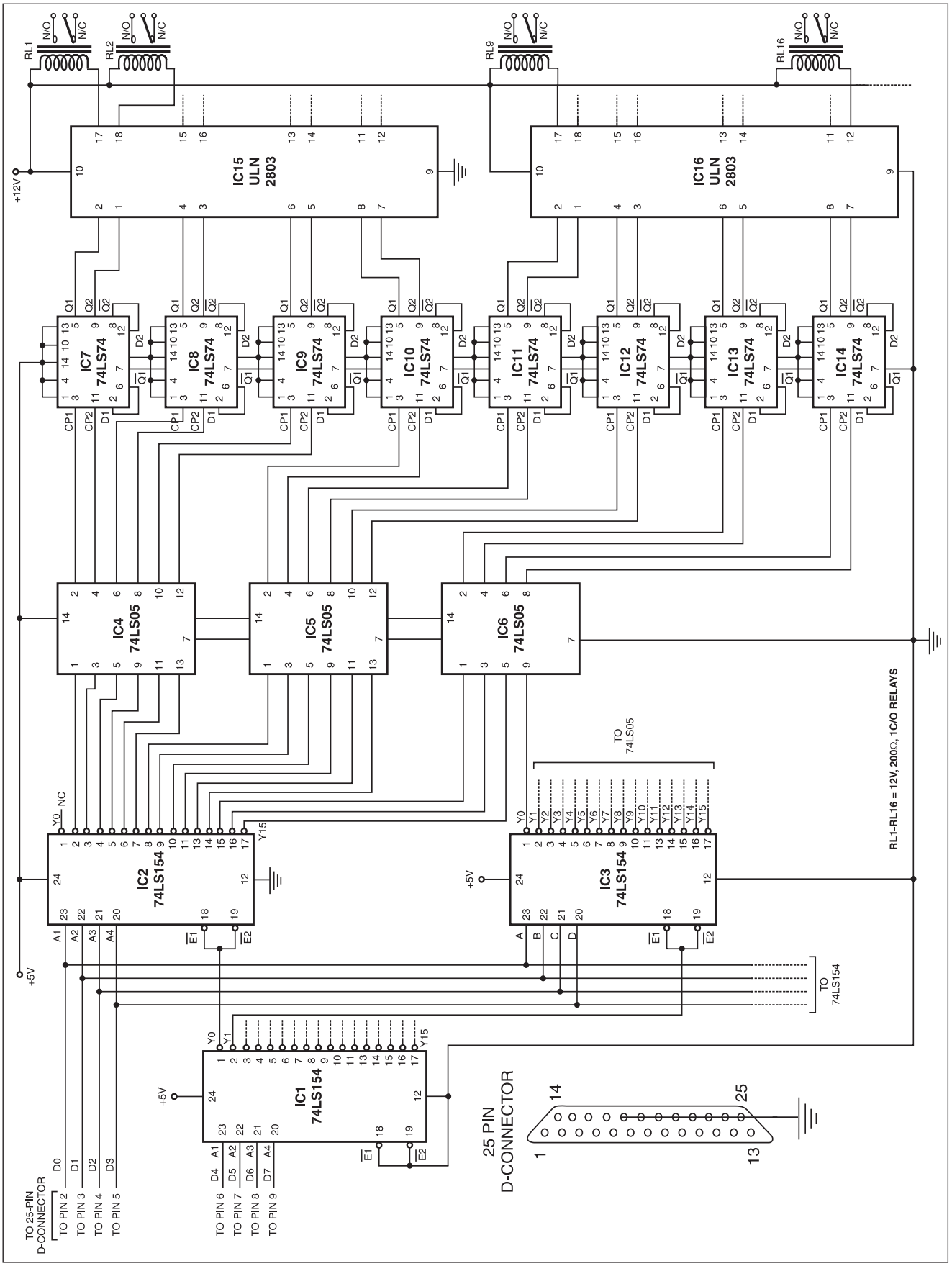


Fig. 2: The interface circuit for multiple device switching using PC's parallel port

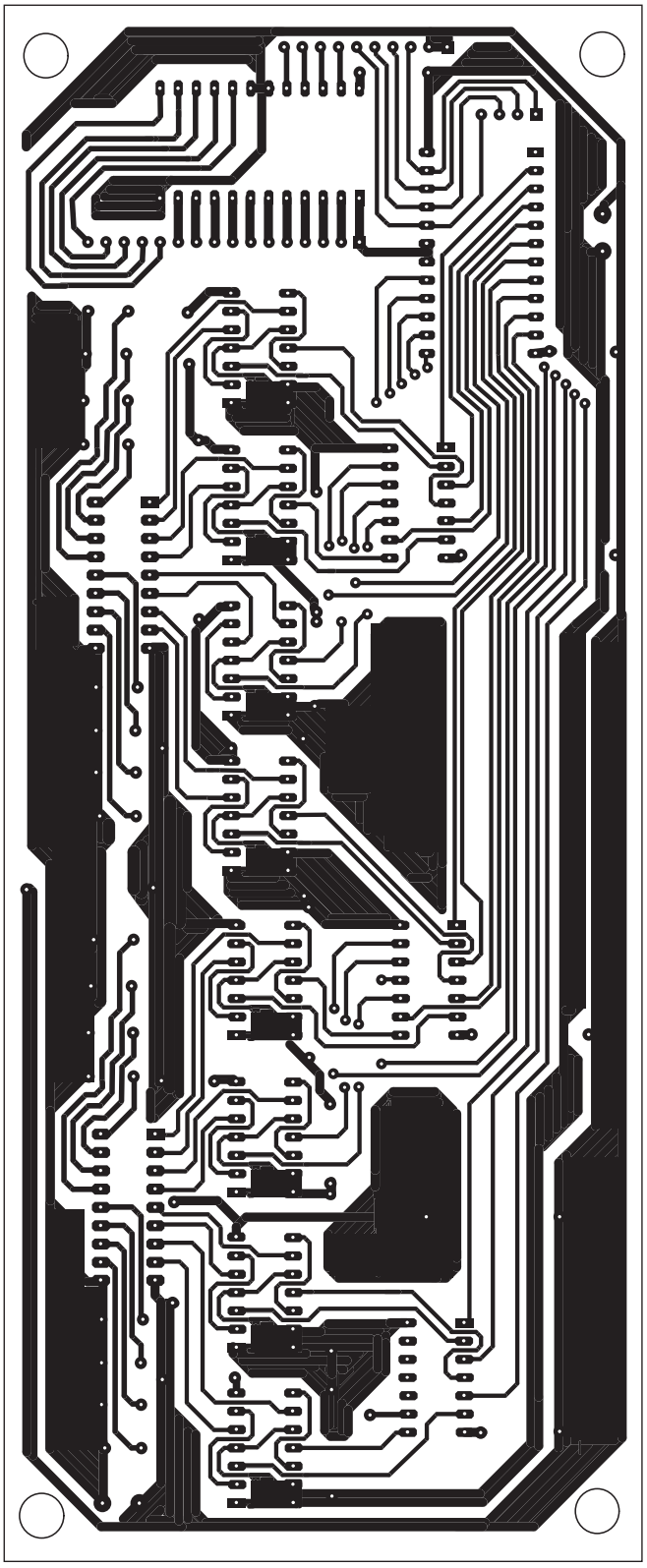


Fig. 3: Actual-size, single-side PCB for multiple device switching using PC's parallel port

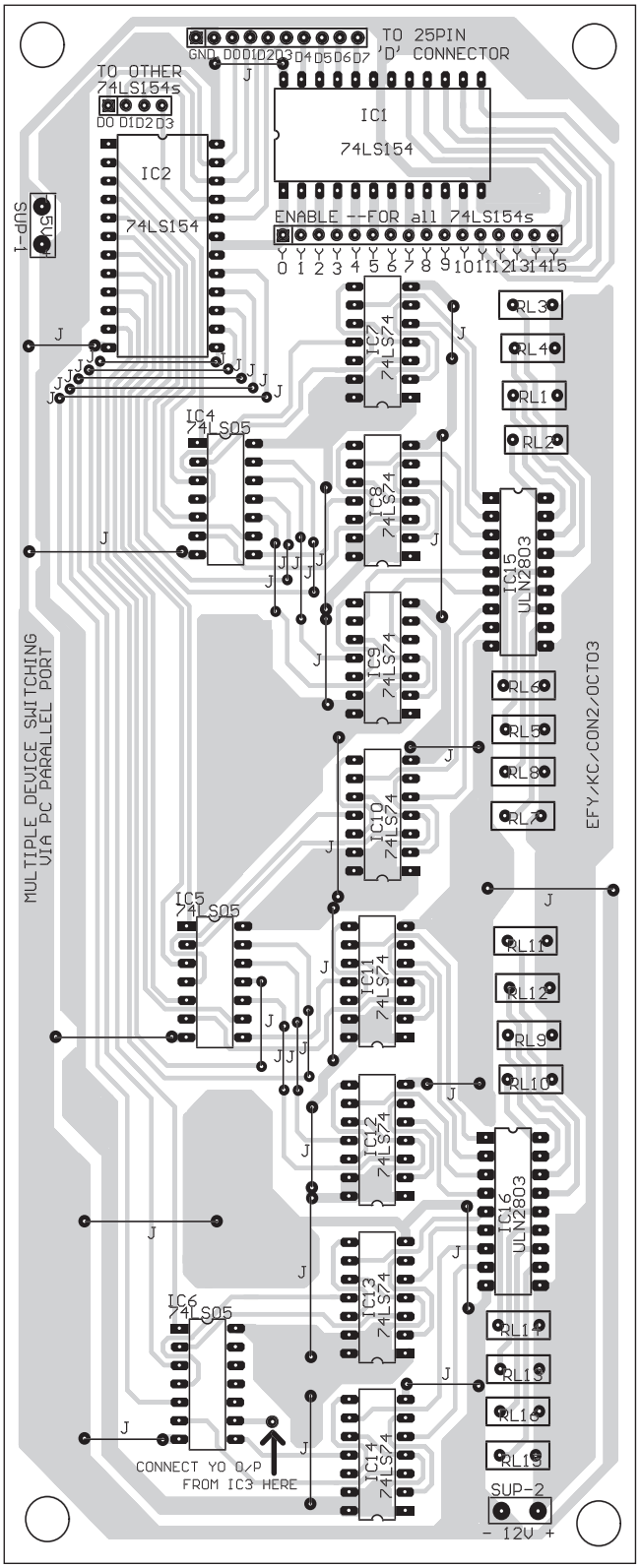
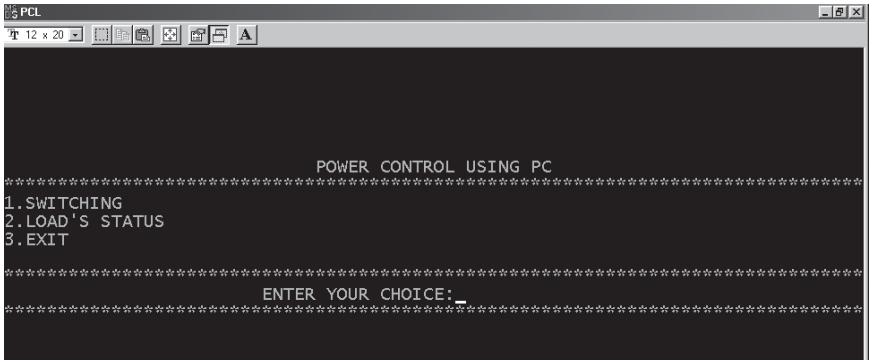


Fig. 4: Component layout for the PCB

18 through 25 are shorted to ground. Data lines D0 through D3 are input addresses for IC2 and IC3, and data lines D4 through D7 are input addresses for IC1.

When Enable pins $\overline{E1}$ and $\overline{E2}$ (active low) are high, all the outputs go high irespective of the address inputs (A1 through A4). Enable pins $\overline{E1}$ and $\overline{E2}$ of IC1 are grounded (permanently enabled) and its output pins Y0 through Y15 are connected to Enable pins of the respective decoder ICs 74LS154 of the next stage



Screenshot of the program output

(here IC2 and IC3).

Initially all the data input lines (D0 through D7) are low. Thus, except Y0, all the outputs of IC1 and IC2 are high. The output Y0 of IC2 is not used, for the reason that when all the input data lines are low, none of the outputs can be used for switching the loads. Suppose, out of eight input data lines, D0 is high. So, except Y1, all the outputs of IC2 will be high. Y1 is then inverted using IC4 (IC 74LS05). The output of IC4 at pin 2 is given to pin 3 of IC7 (IC 74LS74). IC 74LS74 is a dual D-type flip-flop used in toggle mode for latching the data.

With an active rising edge of the clock pulse (CP1 or CP2), the data input will be locked into IC7 through IC14. Thus, the output data will be latched until the next rising edge of the input clock pulse arrives. The outputs of ICs 74LS74 are given to relay driver ICs ULN2803 (IC15 and IC16), which, in turn, drive the relays. The relays switch on/off the appliances. It means that alternate rising edge of clock pulses will toggle the state of relay/device.

The actual-size, single-side PCB for the interface circuit for multiple device switching using a PC's parallel port is shown in Fig. 3 and its component layout in Fig. 4. Please note that IC3 is not shown on the PCB since only a single output (Y0) is used from this IC, which may be mounted on subsequent PCB if more than 15 devices are to be controlled.

Software program

The program to control the appliances is written in C. It is compiled using Turbo C compiler. The program can be written on any Windows text editor (notepad, wordpad, etc). The C programming language is more user-friendly and easy to understand than other programming languages. The source code (PCL.C) of the program is given at the end of this article.

On running the program, a menu-like screen appears, which prompts you to enter your choice, viz, 1 for Switching, 2 for Load's Status, and 3 for Exiting the program.

The Switching option 1 is used to switch on/off the appliance interactively. The Load's Status option 2 displays the status of all the loads that are already 'on'. The Exit option enables you to exit from the application program. A screenshot of the program output is shown above.

The advantage of this software is that the users would know the current device/load status. If a particular load is already 'on' and by mistake the user tries to turn it on again, the software automatically tells the user that the load is already 'on'. In the program, the 'outportb' command outputs the desired data from the parallel port.

This software program can control up to 255 electrical loads. However, due to limitation of space, we've presented here the circuit for switching only up to 16 loads.

Note. The source code and executable file of the program are included in the CD.

PCL.C

```
 /* R.KARTHICK,III ECE,K.L.N.COLLEGE OF ENGG,MADURAI.
 E                                                -
MAIL:KARTHICK_KLNCE@REDIFFMAIL.COM*/
 #include<stdio.h>
 #include<conio.h>
 #include<dos.h>
 #include<string.h>
 void load();
 void status();
 void ext();
 int nt,i,v[256],n1;
 int PORT=0x0378,z,nb=0;
 void main()
  {
  clrscr();
                for(i=0;i<256;i++)
                  v[i]=0;
                while(1)
                  {
                              clrscr();
                              gotoxy(30,7);
 printf("POWER CONTROL USING PC");
                              gotoxy(1,8);

 printf("****************************************");

p r i n t f ( " 1 . S W I T C H I N G \ n 2 . L O A D ' S
STATUS\n3.EXIT\n");
                              gotoxy(1,13);

 printf("****************************************");
                              gotoxy(25,14);
 printf("ENTER YOUR CHOICE:");
                              gotoxy(1,15);

printf("****************************************");
                              gotoxy(43,14);
                              scanf("%d",&nt);
                               switch(nt)
                               {
                               case 1:load();break;
                               case 2:status();break;
                               case 3:ext();break;
                               }
                    }
}


void load()
          {
                                   clrscr();
 printf("Enter the load number(1-255):");

scanf("%d",&nt);
                if(nt<=255 && nt>0)
                                                 {
                                                 clrscr();
                                                 gotoxy(1,12);

printf("****************************************");
                gotoxy(1,13);

printf("1.ON\n2.OFF\n");
                gotoxy(1,15);
printf("****************************************");
                gotoxy(1,16);
 printf("ENTER YOUR CHOICE:");

 scanf("%d",&n1);
                                                 if(n1==1  &&
v[nt]==0)
                                                 {

 outportb(PORT,nt);
 printf("Your load no %d is on",nt);
                v[nt]=1;
                delay(10);

 outportb(PORT,0);
                                                 }
 else  if(n1==1 && v[nt]!=0)
 printf("Your load no %d is already on",nt);
    if(n1==2 && v[nt]==1)
                                                 {

 outportb(PORT,nt);
  printf("Your load no %d is off",nt);
                v[nt]=0;
                delay(10);

 outportb(PORT,0);
                                          }
  else if(n1==2 && v[nt]!=1)
 printf("Your load no %d is already off",nt);
                                                 }
     getch();
     }
```

```
void status()
 {
 int x1;
  clrscr();
printf("The following loads are on:\n");
   for(x1=0;x1<=255;x1++)
     if(v[x1]==1)
 printf("%d\t",x1);
      getch();
  }
  void ext()
 {
 int x2;
    clrscr();
    for(x2=0;x2<=255;x2++)
    if(v[x2]==1)
      outportb(PORT,x2);
      exit(0);
```

❑

Readers' comments:

Q. The circuit is showing short-circuit at 5V supply. I've checked all the tracks of the PCB but found no short-circuit. There is no short circuit when IC 7405 is removed. What could be the problem?

Nilesh Todarmal
Through e-mail

The author, R. Karthick, replies:

A. The IC 7405 may be faulty, or check for any external short circuit. If the problem persists even after replacing IC 7405 with IC 7404 (both the ICs are used for inverting the signal), check your PCB again.

PROPORTIONAL LOAD CONTROL USING PC

ARVIND S.

Controlling household equip-ment from a PC has been a popular concept for many years now. The circuit described here provides a proportional control with respect to a digital input derived from the PC. The digital output from the computer's parallel port is used to control the power delivered to the load in a linear and proportional manner. The digital input range of the circuit is 0-255 (8-bit data) and the circuit can provide linear output control from 5 to 95 per cent of the mains input.

Circuit description

Fig. 1 shows the circuit for proportional load control using a PC. Triac BT139 used in the AC circuit controls the amount of power transferred to the load. The amount of power is controlled by varying the phase angle of triac triggering.

The mains AC waveform is shown in Fig. 2(a). The triggering of the triac is done with respect to the zero crossing of the AC signal. Every time the AC waveform crosses zero voltage line, the triac has to be triggered by the pulse at the gate. Once this is done, the triac latches and remains 'on' until the AC waveform reaches 0V. By delaying the triggering of the triac, the amount of power delivered to the load within an AC cycle (180 degrees) can be altered.

The digital output data at the parallel port (controlled by the program) is converted into analogue equivalent using latch IC 74HC573 (IC3) and an R-2R ladder network. The analogue voltage varies from 0 to 5V depending on the digital input from the parallel port. A digital value of 255 (maximum value) gets converted into an analogue equivalent of 5V, and 0 (minimum value) gets converted into 0V.

The analogue value is compared against the time base generated by a stable ramp generator and gets converted into corresponding time delay. In case the ramp and the analogue equivalent are the same, comparator N4 gives a pulse to opto-

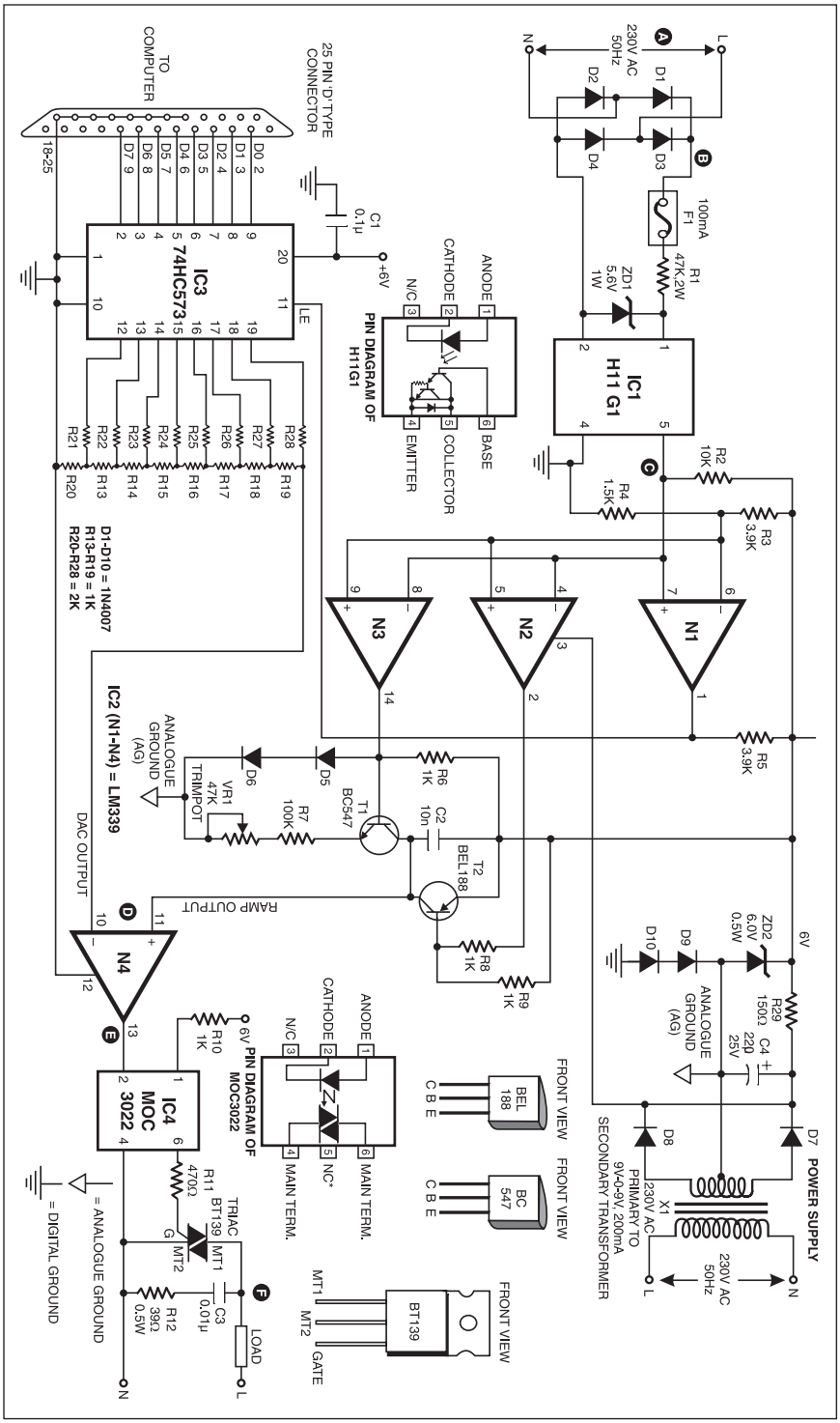


Fig. 1: The circuit for proportional load control using PC

PARTS LIST

Semiconductors:

IC1	- H11G1 optically coupled isolator
IC2	- LM339 comparator
IC3	- 74HC573 octal D-type latch
IC4	- MOC3022 optically coupled bilateral switch
T1	- BC547 npn transistor
T2	- BEL188 pnp transistor
TRIAC	- BT139 triac
D1-D10	- 1N4007 rectifier diode
ZD1	- 5.6V, 1W zener diode
ZD2	- 6.0V, 0.5W zener diode

Resistors (all ¼-watt, ±5% carbon, unless stated otherwise):

R1	- 47-kilo-ohm, 2W
R2,	- 10-kilo-ohm
R3,R5	- 3.9-kilo-ohm
R4	- 1.5-kilo-ohm
R6, R8-R10, R13-R19	- 1-kilo-ohm
R7	- 100-kilo-ohm
R11	- 470-ohm
R12	- 39-ohm, 0.5W
R20-R28	- 2-kilo-ohm
R29	- 150-ohm
VR1	- 47-kilo-ohm trimport

Capacitors:

C1	- 0.1μF ceramic disk
C2	- 10nF ceramic disk
C3	- 0.01μF ceramic disk
C4	- 220μF, 25V electrolytic

Miscellaneous:

X1	- 230V AC primary to 9V-0-9V, 200mA secondary transformer
F1	- 100mA fuse
	- 25-pin D-type connector

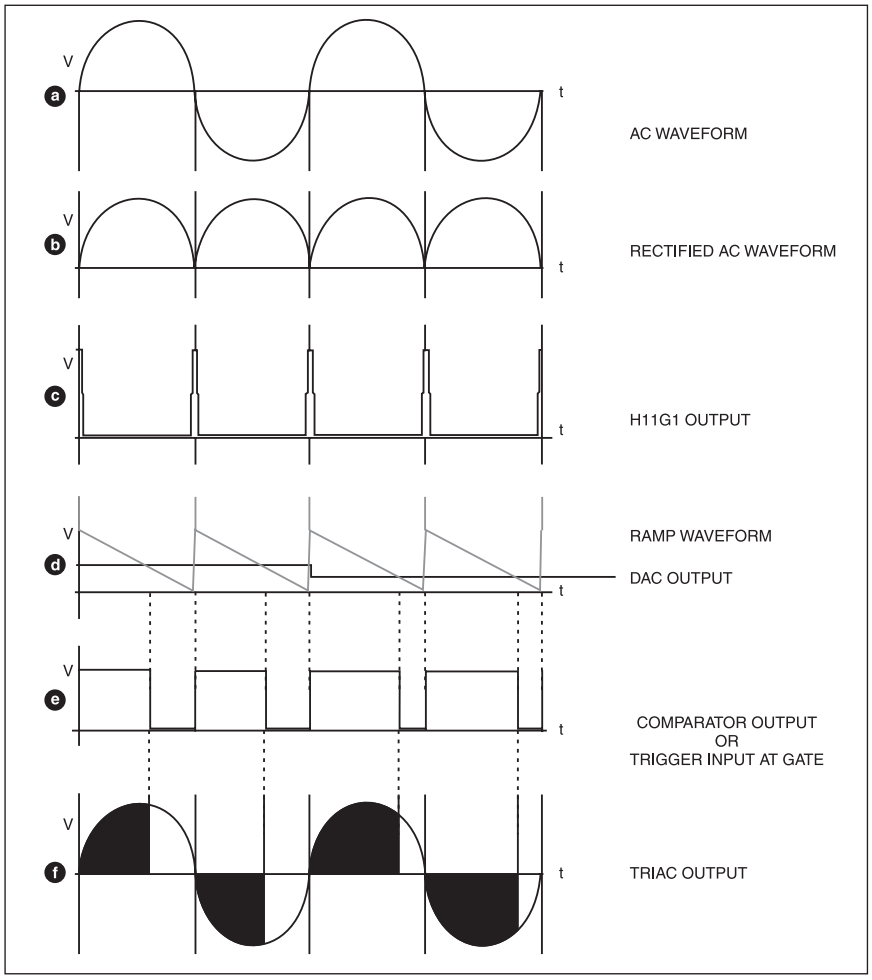


Fig. 2: Waveforms at different stages in the circuit

isolated diac MOC3022 (IC4), which, in turn, triggers the triac. This cycle is repeated for every zero-crossing in the AC line.

The MOC3022 is an optically-coupled isolator consisting of a GaAs infrared light-emitting diode (LED) and a light-activated silicon bilateral switch, mounted in a standard 6-pin dual-in-line package, that controls triggering of the triac.

A snubber circuit removes all EMI emissions from the AC line and also protects the triac. It comprises capacitor C3 and resistor R12.

Zero-crossing detection. Opto-isolator H11G1 (IC1) is used as the zero-crossing detector. It is an optically-coupled isolator consisting of an infrared LED and a high-voltage npn silicon Darlington phototransistor that has an internal base-emitter resistor to optimise switching speed.

The AC signal from mains is rectified by a full-wave bridge rectifier comprising diodes D1 through D4. The rectified output is pulsating DC (shown in Fig. 2(b)) having the same amplitude as the AC waveform, except that now it varies from 0 to 220V_{rms}, 100 times in a second. It is fed to the internal LED of opto-isolator IC1 through current-limiting resistor R1. The light emitted by the LED triggers the internal transistor of IC1 and it gives out a digital signal. The digital signal is used as a zero-crossing signal (shown in Fig. 2(c)).

Thus, when the AC waveform crosses zero, the inbuilt Darlington transistor pair turns off, setting the output to high (logic 1). And when the AC signal waveform picks up in amplitude either during positive half cycle or negative half cycle, the internal LED of IC1 conducts, switching on the internal transistor of IC1 and setting the output to low (logic 0). Zener diode ZD1 limits the signal to 5.6 volts.

Digital interface. The low-cost 74HC573 D-type octal latch (IC3) is used as the digital input interface for the circuit. Its input is connected to a 25-pin D-type connector that plugs into the computer's parallel port (LPT1) at the back of the PC. Pins 2 through 9 form the 8-bit data output port, that can be used to output data through it. Base address of the first parallel port (LPT1) is 0378 in hexadecimal (hex).

At every zero crossing of the signal, new data from the computer is latched on the output of IC3. Comparator N1 generates a latch-enable signal for IC3 at every zero-crossing of the AC signal waveform. The output of the latch feeds the R-2R ladder network. The ladder network comprises resistors R19 through R28. The digital-to-analogue converter (DAC) is the combination of latch IC3 and the R-2R network.

Ramp generator. A capacitor charged at a constant current (I) develops a linear voltage (V=I×t/C) across it. When a 10nF capacitor (C2) is charged at a constant current of 5 μA for a period of 10 ms, it develops 5V across it. This is the basis of the ramp generator operation.

Transistor T1 (BC547) is wired to deliver a constant current irrespective of the load resistance. The rate of constant cur-

rent is set by resistor R7 and trimpot VR1.

By synchronising the start of charging of the capacitor and the zero crossing of AC signal, a linear ramp in sync with the AC signal (refer Fig. 2(d)) is generated. (In the waveform shown in Fig. 2(d) the DAC output is superimposed over the ramp.) At the end of the cycle, comparator N2 triggers pnp transistor T2 (BEL188), which discharges capacitor C2 to prepare it for another cycle. During this brief period, N3 switches off constant-current generating transistor T1 (BC547).

The ramp voltage is fed to one of the inputs of comparator N4 and the other input of the comparator is connected to the output of the DAC (R-2R ladder). When the ramp voltage reaches the R-2R ladder voltage (analogue equivalent of the digital input from the computer's LPT1 port), comparator N4 goes low and the internal LED of IC4 conducts to trigger triac BT139 with the help of opto-isolated diac MOC3022 (IC4).

Figs 2(e) and (f) show the outputs of comparator N4 and the triac, respectively. In Fig. 2(f), the shaded portion of the waveform represents the output during the conduction state of the triac.

Power supply. The AC mains supply is stepped down by transformer X1 to deliver a secondary output of 9V-0-9V AC at 200 mA. The output of the transformer is rectified by a full-wave rectifier. Capacitor C4 acts as a filter to eliminate ripples. Zener ZD2 provides regulated 6V power supply for the circuit excluding comparator. A 12V DC supply is applied to the comparator.

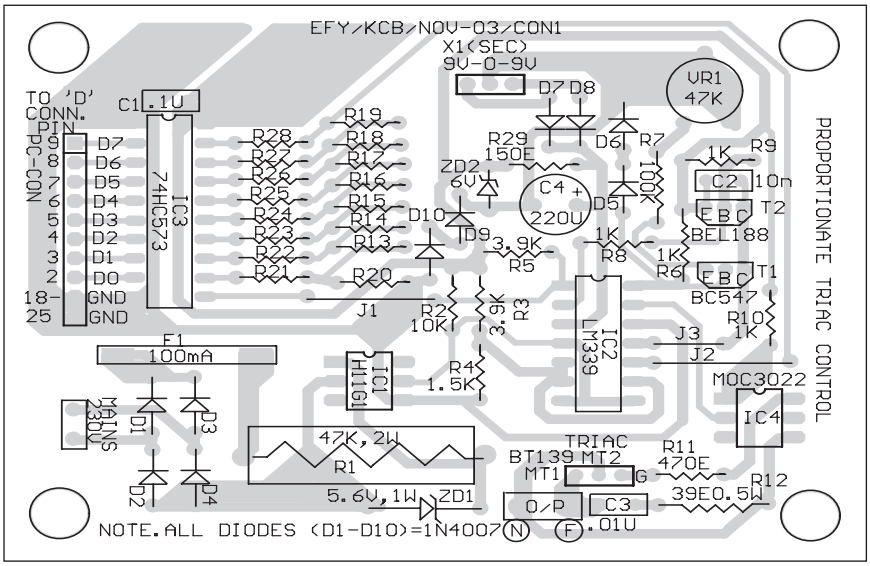


Fig. 4: Component layout for the PCB

Example

Let the digital input be 1000 0000 (128 in hex), which is half the maximum 8-bit value of 1111 1111 (255 in hex).

The analogue equivalent of the R-2R ladder network is 2.5V, for a 5V supply.

At zero crossing (t=0), the voltage across capacitor C2 is 0V with respect to 5V. (The ramp voltage has a negative slope, and, as time progresses, the voltage across the capacitor decreases with respect to ground.)

The 5μA charging current (I) is set by resistor R7 and trimpot VR1 that are connected in series with the emitter of transistor T1 (BC547). Thus the time taken by the ramp voltage to reach 2.5V is:

$t = CxV/I$

$= 5$ ms

This is half the 10ms time duration of one rectified AC cycle.

After 5 ms, comparator N4 triggers triac BT139 and only half the power is delivered to the load. Thus the triac is triggered proportional to the digital input. This is particularly useful for closed-loop applications.

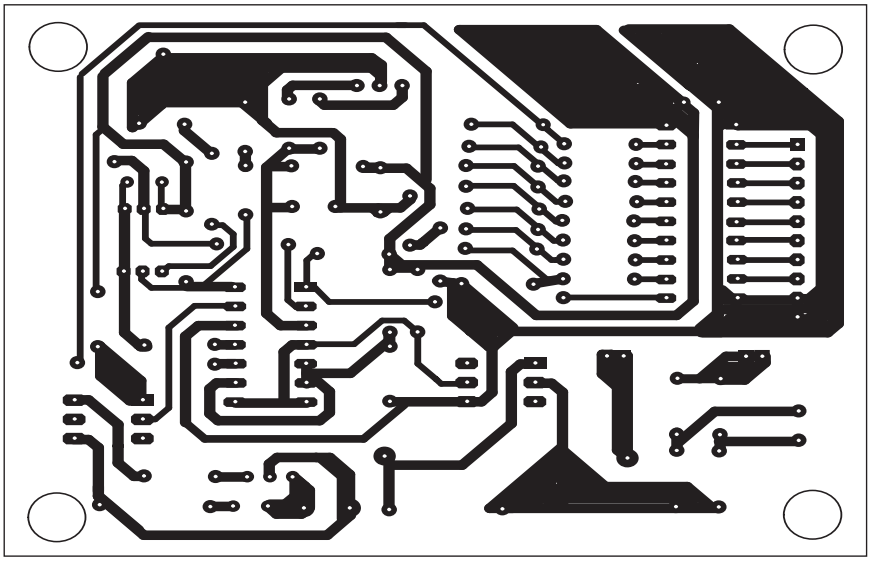


Fig. 3: Actual-size, single side PCB layout for proportional load control using PC

C program

A simple C-language program (given at the end of this article) is used to calibrate and test the circuit after it is connected to the parallel port (LPT1) of the computer. The program has two modes of operation, namely, automatic and manual.

In the automatic mode, a series of digital values that start from 0 and increase until the value reaches 255 and then reduce once again to reach 0, is sent to the port. This produces a dual-slope ramp that changes at the rate of 1 bit every 50 ms. The load power increases linearly from 0% to a maximum of 100% and then reduces back to 0%.

In the manual mode, any value from 0 to 255 can be sent to the port. Sending a value of 255 results in maximum power (100%) to the load and sending a value of 0 results in minimum power (0%) to the load. Sending any other value results in a load power that is proportional to the input.

Calibration

The slope of the ramp generated is critical for proper operation of the circuit.

The slope is dependent on the capacitor's charging current amplitude. This amplitude is fixed by resistor R7 and trimpot VR1. Because of component tolerances, the charging current cannot be set at the time of design. Improper setting of the current amplitude can result in unsatisfactory performance at the lower end of the input range.

After assembling the circuit, connect the D-type connector to the parallel port (LPT1). Set trimpot VR1 such that the resistance across VR1 is 0. Switch on the power to the circuit and run the C program in manual mode. In this mode, a value of 0 has to be sent to the parallel port. Now increase VR1 until the voltage across the load crosses 0V. After calibration, the circuit is ready to be used.

An actual-size, single-side PCB for the circuit is shown in Fig. 3 with its component layout in Fig. 4.

Cautions. 1. Use a low-tolerance polypropylene/polyester capacitor as C2.

2. Though the 100mA fuse (F1) is optional, connecting it will protect H11G1 in case the 47k resistor fails.

3. ICs must be handled carefully.

4. Using resistors with 1% tolerance for the R-2R ladder network improves linearity.

5. Any 'sensitive gate' triac can be used in place of BT139. The tab in BT139 is connected to MT2 and care must be taken to isolate this from the rest of the circuit.

6. Properly isolate the high side (the side connected to mains) of optocouplers H11G1 and MOC3022.

7. Don't connect Analogue Ground (AG) and Digital Ground (GND) together.

TRIAC.C

```
/* Proportional Load Control */

#include <dos.h>
main()
{
 int a,b,c,d;
 int i=9;
 clrscr();
 printf("\n\n\n\n\n\n\n");
 printf("****** TEST PROGRAM  *******\n");
 printf("******  Written by Arvind S *******\n");
 printf("**********************\n");
 printf("******  Proportional Load  *******\n");
 printf("*******  Control using PC  *******\n");
 printf("\n press any key to cotinued");
 getch();
 while(i!='0')
 {
 d=10;
  clrscr();
  printf("Input Choice\n");
   printf("Enter '1' for Automatic testing, '2' for manual testing\n");
  a=getch();
  if (a=='1')
    {
     b=0;
     while(b!=255)
     {
      outportb(0x378,b);
      b=b+1;
      delay(50);
      printf("Sending Value %d to the port\n",b);
     }
     while(b!=0)
     {
      outportb(0x378,b);
      b=b-1;
      delay(50);
      printf("Sending Value %d to the port\n",b);
     }
    }
  if (a=='2')
    {
      clrscr();
      printf("Type in Brightness value (0-255) and press Enter key\n");
      scanf("%d",&c);
     printf("\n Sending value %d to the port\n",c);
      outportb(0x378,c);
    }
  printf("Type '0' to quit, '9' to continue\n");
  i=getch();
  clrscr();
  }
 }
```

❑

BINARY-TO-HEXADECIMAL DECODER

SUNIL P.B.

Here is a binary-to-hexadecimal decoder that can decode any binary data (up to 24-bits) into its corresponding hexadecimal data (up to 6 hexadecimal digits). The binary data to be encoded is supplied via inputs of IC1 through IC3, and the output in hexadecimal format is displayed on six 7-segment displays as hexadecimal digits.

A number system with base 2 is known as the binary number system. Only two digits, namely, '0' and '1', are used to represent any number in the binary number system, and these are known as bits. The system has minimal base and it is a positional system, i.e. every position is assigned a specific weight.

Most computers use the hexadecimal number system with base 16. There are, infact, 16 combinations of 4-bit binary numbers. Any number is represented by 16 distinct symbols, which include numerals '0' through '9' and alphabets 'A' through 'F'. Since numbers and alphabets

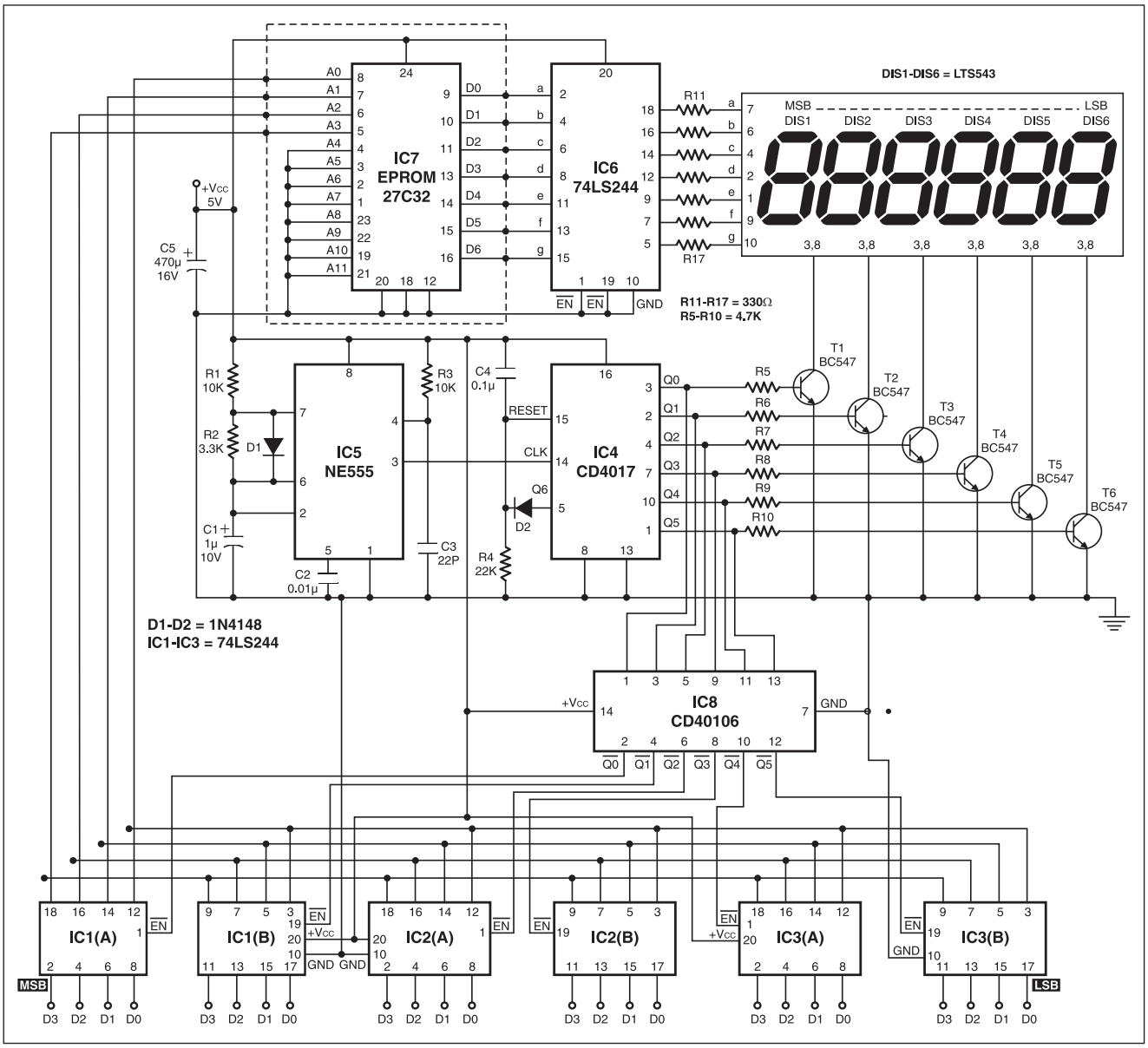


Fig. 1: Circuit diagram of binary-to-hexadecimal decoder

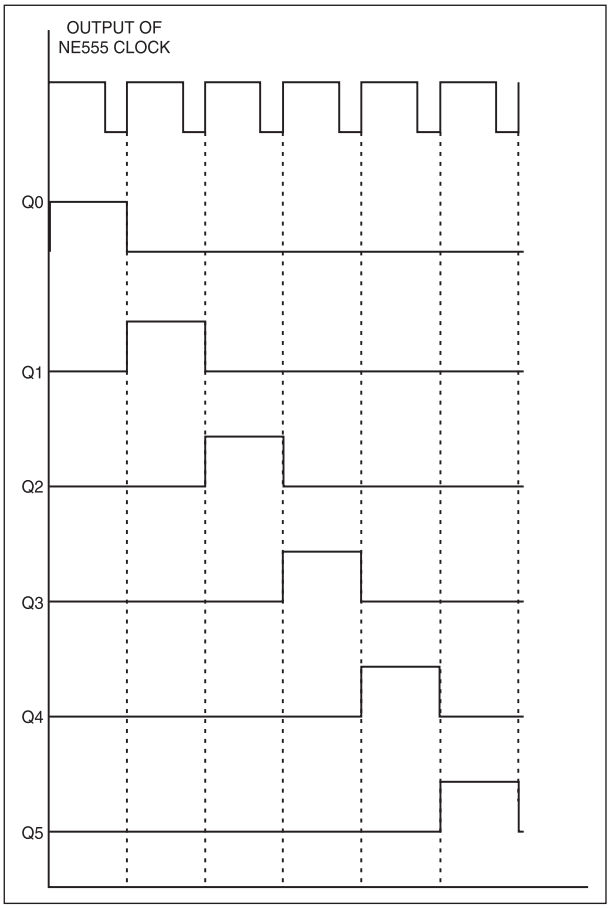


Fig. 2: Timing diagram of CD4017

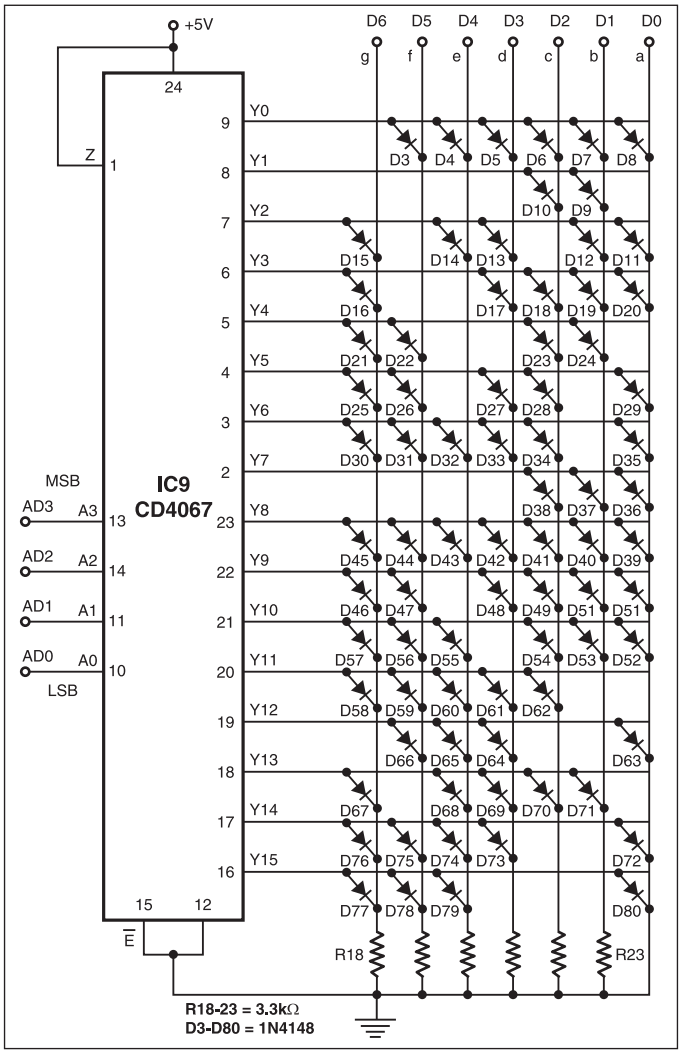


Fig. 3: Alternative circuit for EPROM

are used to represent the digits, this is an alphanumeric number system. Binary and equivalent hexadecimal numbers are shown in the table; since capital B and D cannot be displayed using 7-segment displays, hence lower-case characters 'b' and 'd' are used instead.

Circuit description

The circuit of the binary-to-hexadecimal decoder is shown in Fig.1. It works in scanning mode. Nibbles to be decoded are scan-ned sequentially at regular interval of time.

The scanning circuit comprises three octal buffers (IC1 through IC3), a Johnson ring counter (IC4), and an astable multivibrator (IC5). The scanning clock pulse is generated by NE555 (IC5), which is configured for astable operation in the circuit.

Timing capacitor C1 normally charges through resistors R1 and R2, but diode D1 bypasses resistor R2 during charging of C1. It discharges towards gro-

Number system			Segment data									Hex
Decimal	Binary	Hexadecimal	D7	D6	D5	D4	D3	D2	D1	D0	EPROM address	Equivalent EPROM data
0	0000	0	0	0	1	1	1	1	1	1	000	3F
1	0001	1	0	0	0	0	0	1	1	0	001	06
2	0010	2	0	1	0	1	1	0	1	1	002	5B
3	0011	3	0	1	0	0	1	1	1	1	003	4F
4	0100	4	0	1	1	0	0	1	1	0	004	66
5	0101	5	0	1	1	0	1	1	0	1	005	6D
6	0110	6	0	1	1	1	1	1	0	1	006	7D
7	0111	7	0	0	0	0	0	1	1	1	007	07
8	1000	8	0	1	1	1	1	1	1	1	008	7F
9	1001	9	0	1	1	0	1	1	1	1	009	6F
10	1010	A	0	1	1	1	0	1	1	1	00A	77
11	1011	b	0	1	1	1	1	1	0	0	00B	7C
12	1100	C	0	0	1	1	1	0	0	1	00C	39
13	1101	d	0	1	0	1	1	1	1	0	00D	5E
14	1110	E	0	1	1	1	1	0	0	1	00E	79
15	1111	F	0	1	1	1	0	0	0	1	00F	71

PARTS LIST

Semiconductors:

IC1- IC3, IC6 - 74LS244 octal buffer
IC4 - CD4017 Johnson ring counter
IC5 - NE555 timer
IC7 - 27C32 EPROM
IC8 - CD40106 hex inverter
IC9 - CD4067, 16 channel multiplexer/demultiplexer
T1-T6 - BC547 npn transistor
D1-D80 - 1N4148 switching diode

Resistors (all ¼-watt, ±5% carbon, unless stated otherwise):

R1, R3 - 10-kilo-ohm
R2, R18-R23 - 3.3-kilo-ohm
R4 - 22-kilo-ohm
R5-R10 - 4.7-kilo-ohm
R11-R17 - 330-ohm

Capacitors:

C1 - 1μF, 10V electrolytic
C2 - 0.01 μF ceramic disk
C3 - 22pF ceramic disk
C4 - 0.1μF ceramic disk
C5 - 470μF, 16V electrolytic

Miscellaneous:

DIS1-DIS6 - LTS543 common-cathode 7-segment display

und through R2 only. Capacitor C2 bypasses any noise to ground, preventing change in the calculated frequency. Consequently, the maximum duty cycle is increased beyond the normal 50 per cent.

The clock circuit provides 156Hz clock frequency to Johnson decade counter CD4017 (IC4). The complement of the serial output ($\overline{Q0}$) of shift register is connected back to serial input. The resulting circuit is referred to as Johnson counter.

The combination of capacitor C4 and resistor R4 gives power-on-reset pulse to pin 15 (reset) of IC4. When the circuit is switched on, IC4 starts counting from QO to Q5 depending on the timing of clock pulses generated from IC5 and resets at the next output (Q6). Output pin 5 (Q6) of IC4 is connected to reset pin 15 through diode D2. The timing diagram of CD4017 (IC4) is shown in Fig. 2.

The outputs of IC4 are fed to hex inverter CD40106 (IC8). The inverted outputs $\overline{Q0}$ through $\overline{Q5}$ enable buffers IC1 through IC3, respectively. The binary inputs to be decoded are applied to the inputs of buffers. Four-bit outputs of all buffers are connected together and given to address pins A0 through A3 of EPROM (IC7).

At a time, only one buffer outputs the binary input. When the enable input (EN) goes low, the corresponding buffer outputs the binary input and applies the same to the address pins (A0 through A3) of EPROM (IC7). At the same time, the correct hex digit is activated by the same output of IC4 (Q0, Q1,...or Q5) using a transistor (T1, T2,... or T5). Thus the correct hexadecimal equivalent of the applied binary digit is displayed in the correct place on 7-segment display DIS1 through DIS6. When the high output from IC4 is applied to the base of the transistor via a current-limiting resistor, the collector of the transistor goes low and enables the corresponding 7-segment display.

The data to display the hexadecimal equivalent of the binary nibble is preloaded in an EPROM. The EPROM is wired to use only 16 memory locations, i.e. it is addressed by address lines A0 through A3. The rest of address lines, i.e. A4 through A11, are grounded. You can use a smaller EPROM in place of EPROM 27C32, if desired. The EPROM 2732 can be programmed and erased. (For EPROM programming, refer to the article 'Manual EPROM Programmer Cum Verifier' on page 16 of *Electronics Projects Vol. 18*. For EPROM erasing, refer to the article 'Make Your Own EPROM Eraser with Electronic Timer' on page 56 of *EFY March 2002 or Electronics Projects Vol-23*). The EPROM is erased by exposing the EPROM window to UV light for about 30 minutes.

When a binary input is received at the address input of the EPROM, it generates the correct hex data for displaying the corresponding hexadecimal digit. The data outputs from the EPROM are buffered and available on the segment inputs of the 7-segment display through current-limiting resistors of 330 ohms.

The data to be programmed into the EPROM is given in Table I with address location. When the common pin (3 or 8) of the 7-segment display goes low via transistor, the data available at the output of IC6 activates the respective segment to display the corresponding hexadecimal digit. To avoid flickering, the scanning frequency is se-

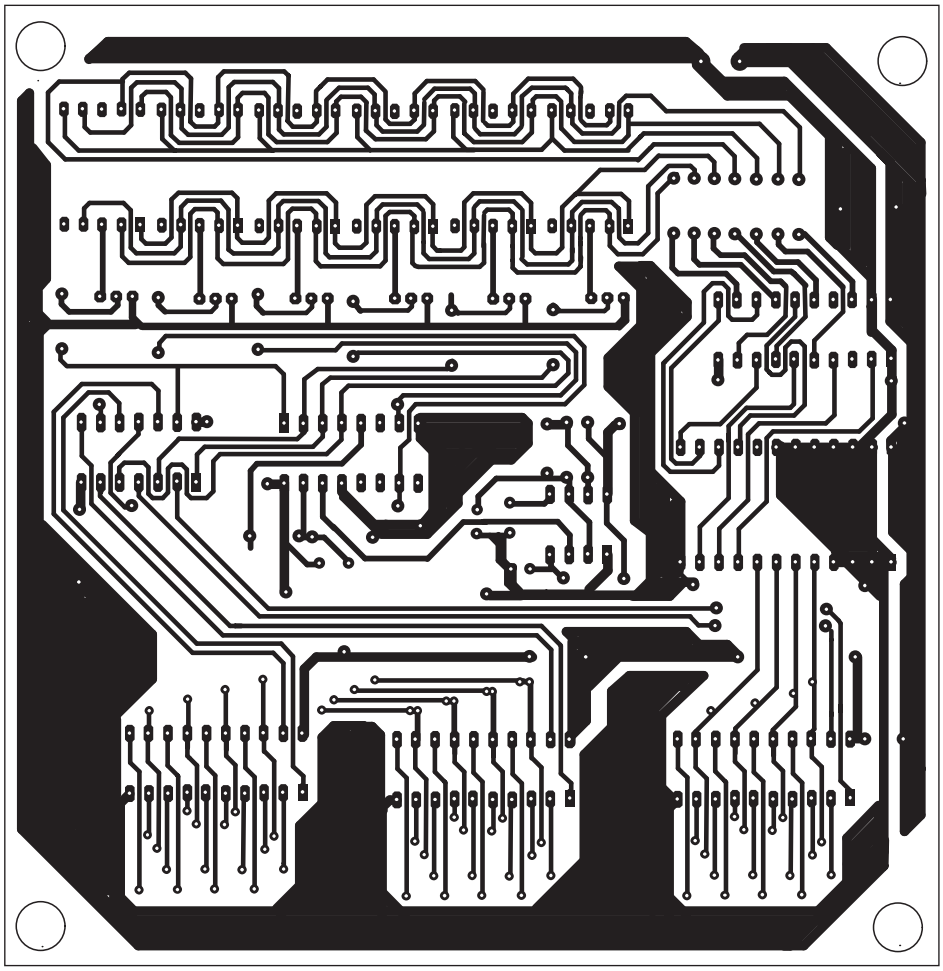


Fig. 4: Actual-size, single-side PCB for binary-to-hexadecimal decoder

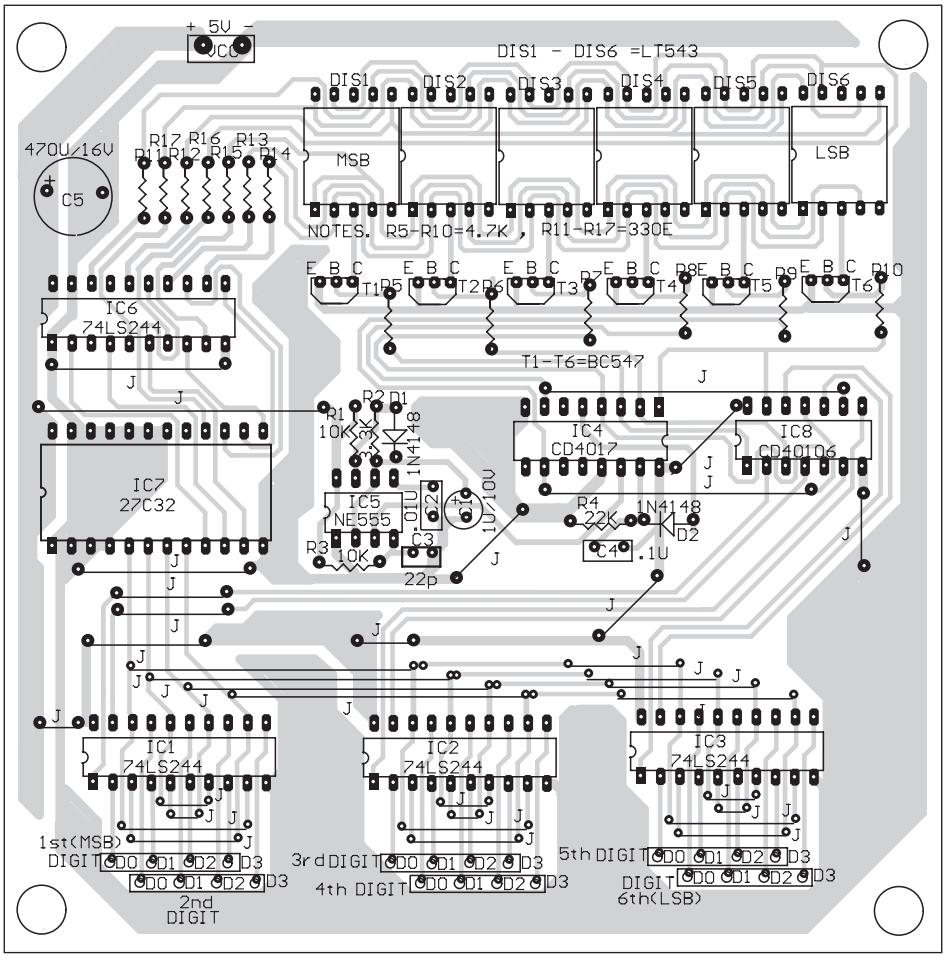


Fig. 5: Component layout for the PCB

lected larger than the persistence of vision of human eyes. The 7-segment display comprises seven light-emitting diodes with their cathodes connected together. This configuration is known as common-cathode 7-segment display.

For those who can't program the EPROM, an alternative circuit is shown in Fig. 3. The circuit comprises demultiplexer CD4067 (IC9), an array of diodes D3 through D80, and resistors R18 through R23. One just needs to replace the circuit within the dotted lines in Fig. 1 with the circuit in Fig. 3.

IC CD4067 contains 16 bidirectional analogue switches, which have their one side connected to the array of diodes through independent outputs Y0-Y16, respectively, and the other side connected to common input (Z) for +5V supply. With enable pin 15 (E) low, one of 16 switches is selected by A0-A3 and connected to data lines D0 through D6 via the array of diodes. All unselected switches are in the high-impedance state (off state).

The demultiplexer works like a binary-to-decimal decoder. Its decoded outputs are connected to the array of diodes to generate the corresponding hexadecimal data for the 7-segment display. Outputs D0 through D6 are connected to the input of buffer IC6.

The actual-size, PCB pattern for the binary-to-hexadecimal decoder is shown in Fig. 4 and its component layout in Fig. 5. The circuit operates off a 5V regulated supply.

❑

CONTROLLING A 7-SEGMENT DISPLAY USING PC'S PARALLEL PORT

K.S. SANKAR

Controlling appliances from your PC is quite interesting and convenient. Here is a simple circuit along with software to interface a 7-segment display to your PC's parallel port. You can display any 4-digit number on four 7-segment displays. This is achieved by means of time-division multiplexing.

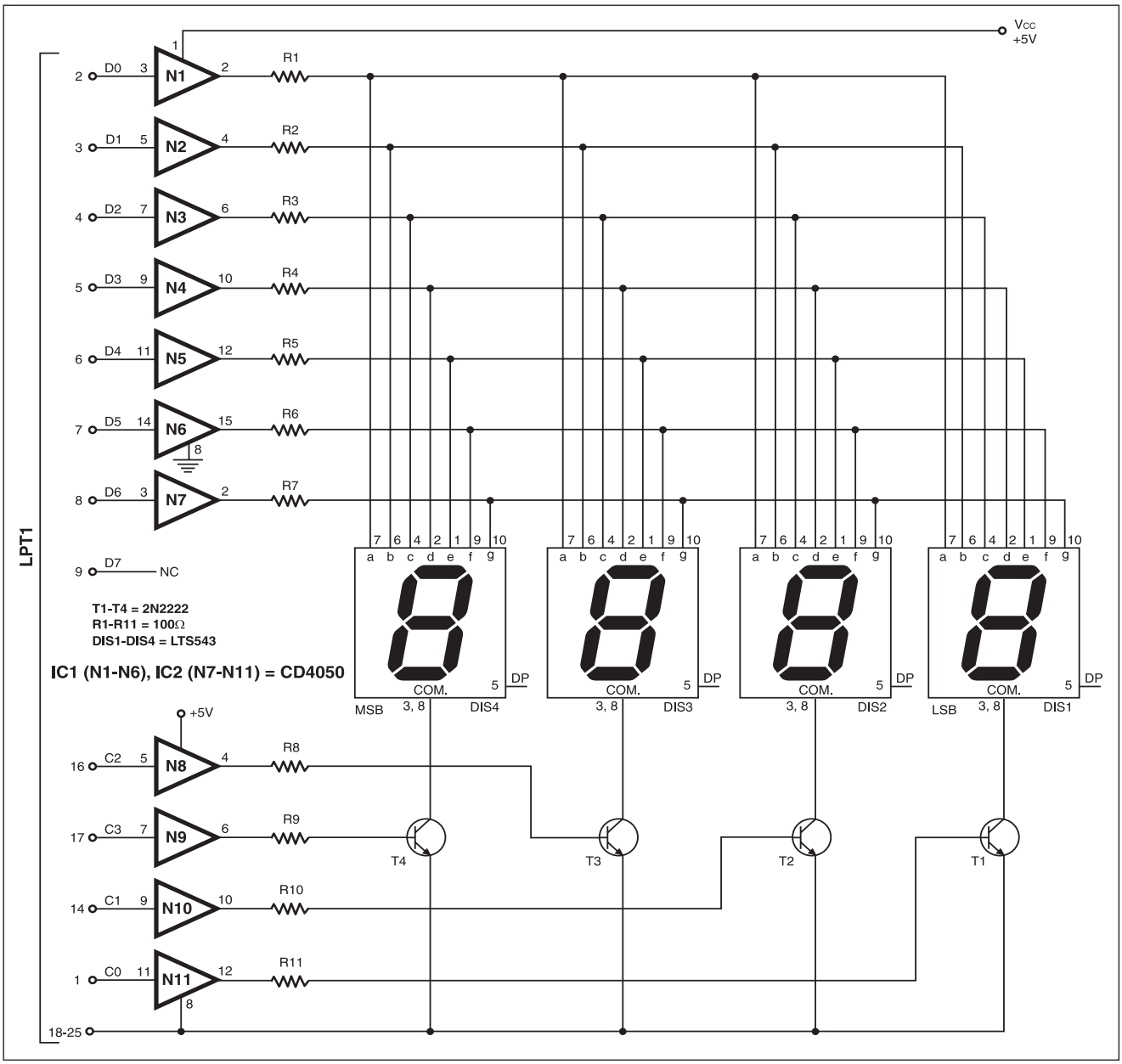


Fig. 1: Circuit for interfacing a 7-segment display with the PC

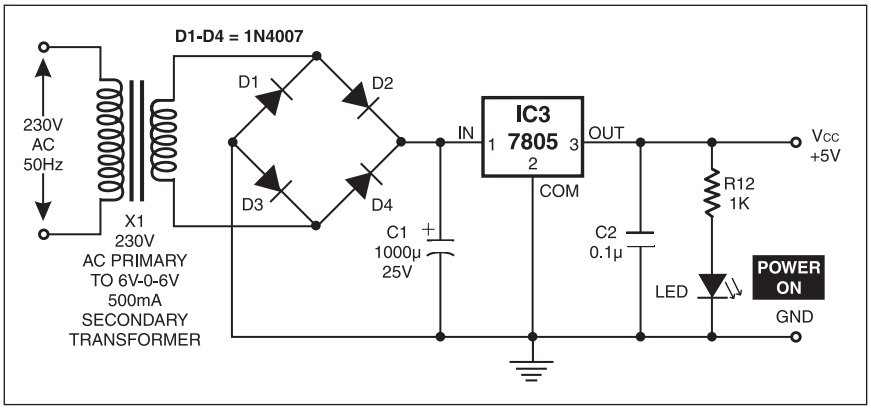


Fig. 2: Circuit of power supply

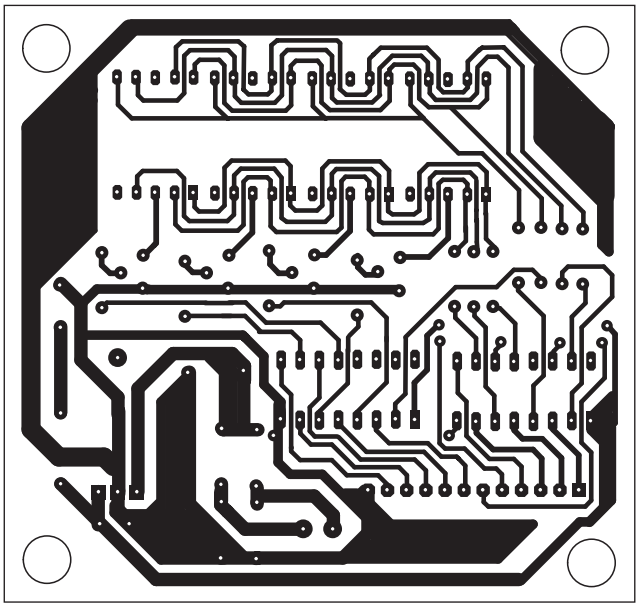


Fig. 3: Actual-size, single-side PCB layout for interfacing a 7-segment display with the PC

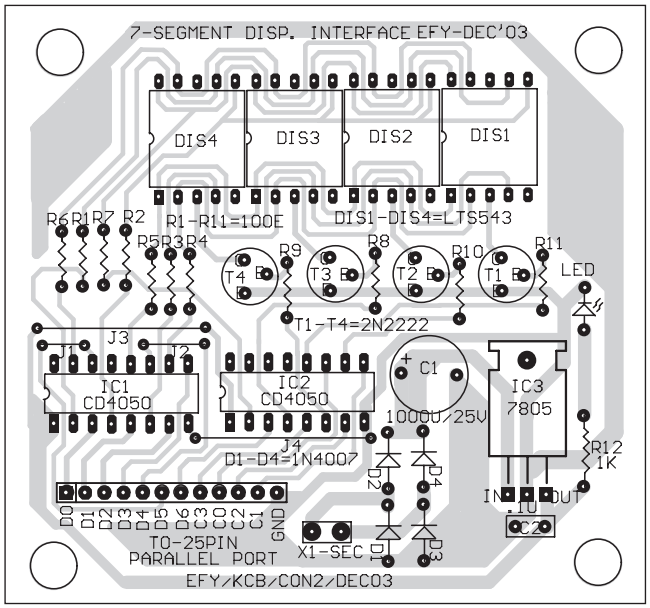


Fig. 4: Component layout for the PCB

Ordinarily the printer port of the PC is meant for outputting the data to the printer. It provides a set of ports, with some lines as output and some lines as input. In addition, there are some open collector lines that can be used as input.

The 25-pin parallel port connector at the back of the PC is a combination of three ports. The port address varies from 378H to 37AH for LPT1. The seven lines of port 378H (pins 2 through 8) are used in this circuit to output the code for 7-segment display via buffers N1 through N7. The remaining one line of port 378H (pin 9) is not used. The four lines (C0 through C3) of control port 37AH (terminating on pins 1, 14, 16, and 17, respectively) are also used as output lines to activate the common cathode pins of displays (DIS1 through DIS4) via buffers N11, N10, N8, and N9 respectively. Outputs on control lines C0, C1, and C3 are inverted inside the PC while output C2 is not inverted. Accordingly, to get a logic high at output pins 1, 14, and 17, one must send a logic 0 to these pins via the program.

All the four 7-segment displays share a common data bus. The PC places the 7-segment code for the first digit on the data bus and enables only the first 7-segment display. After a few milliseconds, the 7-segment code for the first digit is replaced by that of the next digit, but this time only second display is enabled. After all the digits are displayed in this way, the cycle repeats. The cycle continues over and over again. Because of this repetition at a fairly high rate, there is an illusion that all the digits are being continuously displayed.

The seven segments ('a' through 'g') of the four displays are connected to each other as well as to data port pins D0 through D6 via non-inverting hex buffers of CD4050 and current-limiting resistors R1 through R7, respectively, as shown in Fig. 1. Displays DIS1 through DIS4 are enabled by control port pins C0 through C4 with the help of transistors T1 through T4, respectively.

When a high output from buffers N8 through N11 is applied to the base of a transistor via a limiting resistor, the collector of the respective transistor goes low and enables the corresponding 7-segment display.

The 7-segment display comprises seven light-emitting diodes with their cathodes connected together. This configuration is known as common-cathode 7-segment display. All the input pins of the buffer are connected to 25-pin 'D' connector and plug on to the PC's parallel port through ribbon cable.

The power supply circuit is shown in Fig. 2. The AC mains supply is stepped down by transformer X1 to deliver a secondary output of 6V-0-6V AC, 500 mA. The output of the transformer is rectified by a full-wave rectifier comprising diodes D1 through D4. Capacitor C1 acts as a filter to eliminate ripples. Regulator IC 7805 (IC3) provides regulated 5V power supply to the circuit. The LED indicates the power-on state.

Examples

If you want to display digit '1,' segments b and c should glow, which means data

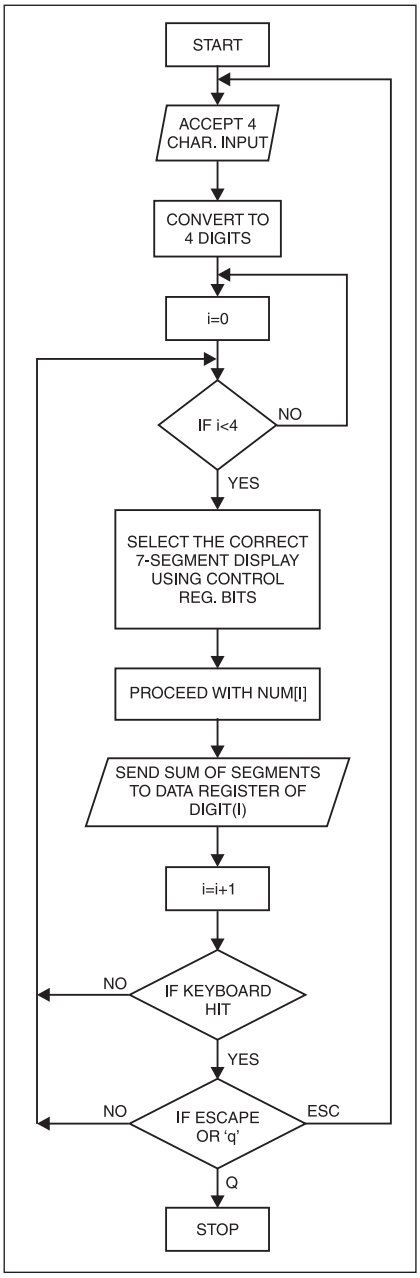


Fig. 5: Flow-chart of the program

port 378H outputs D1 and D2 and corresponding control port pin of the selected display should be high. If all control outputs are high, '1' will be displayed on all the four displays.

Now suppose you want to display the 4-digit number '1234' on the four 7-segment displays. For display of digit '4' on DIS1, segments b, c, f, and g should glow. This is possible by outputing decimal 102 (since b=2, c=4, f=32, and g=64—refer program). Simultaneously, display DIS1 is enabled when control bit C0 is high, while remaining control bits (C1, C2, and C3) are in low state to keep all other displays 'off.' Thus you get '4' on display DIS1. The next digit (3) is displayed on DIS2 when the relevant segments (a, b, c, d, and g) and control bit C1 is high. The remaining two digits (1 and 2) are displayed in similar way.

PARTS LIST

Semiconductors:

IC1, IC2	- CD4050 hex non-inverting buffer
IC3	- 7805 5V regulator
T1-T4	- 2N2222 npn transistor
D1-D4	- 1N4007 rectifier diode
LED	- Red LED
DIS1-DIS4	- LTS543 common-cathode 7-segment display

Resistors (all ¼-watt, ±5% carbon, unless stated otherwise):

R1-R11	- 100-ohm
R12	- 1-kilo-ohm

Capacitors:

C1	- 1000µF, 25V electrolytic
C2	- 0.1µF ceramic disk

Miscellaneous:

X1	- 230V AC primary to 6V-0-6V, 500mA secondary transformer
	- 25-pin D connector

An actual-size, single-side PCB for interfacing the 7-segment display with the PC is shown in Fig. 3 and its component layout in Fig. 4. To make the circuit simpler, you may omit resistors R1 through R11 since 4000B series CMOS have a channel resistance of about 200 ohms, when on.

The software

The software (given at the end of this article) displays the 4-digit number in a loop by displaying one digit at a time for a brief moment. The remaining displays are kept switched off during this time. The process repeats for all the remaining digits. Fig. 5 shows a flow-chart that explains overall working of the software.

When you run the program, the message "Enter a 4-digit number 0000-9999 and press Enter" appears on the PC screen. The accepted 4-character input is converted into four individual digits using mode function. Next, the correct 7-segment display for displaying a particular digit is selected by the control signal bit. Each segment of the display device is assigned with a value. The sum of the segment values for the digit is placed on the data bus. All the four digits are displayed one after another in time-division multiplexing mode. If you want to display another 4-digit number, press Esc key to start the program from beginning. Press Q key to quit the program.

47EFY.C

```
/* 47efy.c */
/* TO DISPLAY 4 DIGIT NUMBER IN  FOUR 7-SEGMENT DISPLAYS */
/* ————THROUGH 'C'LANGUAGE———— */
/* ——— by K.S.Sankar www.mostek.biz ——— */

#include<stdio.h>
#include<conio.h>
#include<math.h>
#define dy 10
void display(int num);
void main()
{
/*  ——a——     a=1
   |     |    b=2
   f     b    c=4
   |     |    d=8
    ——g——     e=16
   |     |    f=32
   e     c    g=64
   |     |
    ——d——
*/
int in;
int n[5],i,j;
int kb;
while(1)
{
outport(0x378, 0); /*Data port D0 to D7= 0 */
clrscr();
printf("ENTER A 4 DIGIT NUMBER  0000-9999 AND PRESS ENTER ");
scanf("%d",&in);
printf("\ndisplayed .. press Esc.to break");
printf("\n            press q or Q to quit ");
/*SEPARATE THE SINGLE 4 DIGIT NUMBER INTO FOUR 1 DIGITS
   USING MOD FUNCTION*/
for(i=1;i<=4;i++,in/=10)
 {
   n[i]=fmod(in,10);
 }
   /*DISPLAY ALL THE FOUR SEGMENTS SIMULTANEOUSLY*/
while(1)    /*Loop for ever*/
 {
    if(kbhit()) /*Look for a key press*/
    {
     kb=getch(); /*Check the character*/
              if(kb==27) /* Is it Esc ?*/
{
 printf("\n program break");/*Esc.so break to start*/
                printf("\npress enter to continue");
                getch();
                break;
                }
   if(kb==113 | kb==81) /*IS it 'q'or'Q'key press*/
                {
                 printf("\n program quit");/* Quit from program*/
                exit(0);
                }
    }
 /*SELECT 1ST SEGMENT AND DISPLAY 1ST NUMBER*/
   outport(0x378,0);/*Data port all low*/
   outport(0x37a,3);/*Select 1st display*/
   display(n[4]); /*Display No.on 1st display*/
   delay(dy); /*Delay*/
 /*SELECT 2ND SEGMENT AND DISPLAY 2ND NUMBER*/
   outport(0x378,0);
   outport(0x37a,15);
   display(n[3]);
   delay(dy);
```

```
 /*SELECT 3RD SEGMENT AND DISPLAY 3RD
NUMBER*/
   outport(0x378,0);
   outport(0x37a,9);
   display(n[2]);
   delay(dy);
 /*SELECT 4TH SEGMENT AND DISPLAY 4TH
NUMBER*/
   outport(0x378,0);
   outport(0x37a,10);
   display(n[1]);
   delay(dy);
  }
}
outport(0x378,0);
getch();
}
void display(int num)
{
/*ASSIGN VALUES TO ALL SEGMENTS*/
int a=1;  /*D0*/
int b=2;  /*D1*/
int c=4;  /*D2*/
int d=8;  /*D3*/
int e=16; /*D4*/
int f=32; /*D5*/
int g=64; /*D6*/
int n;
if(num== 0)
 n=a+b+c+d+e+f;/*Segment on for digit"0"*/
else
if (num == 1)
 n=b+c; /*Segment on for digit"1"*/
else
if (num == 2)
 n=a+b+d+e+g; /*Segment on for digit"2"*/
else
if ( num == 3)
 n=a+b+c+d+g; /*/*Segment on for digit"3"*/
else
if( num == 4)
 n=b+c+f+g;  /*Segment on for digit"4"*/
else
if ( num == 5)
 n=a+c+d+f+g;/*Segment on for digit"5"*/
else
if (num == 6)
 n=a+c+d+e+f+g;/*Segment on for digit"6"*/
else
if( num == 7)
 n=a+b+c; /*Segment on for digit"7"*/
else
if( num == 8)
 n=a+b+c+d+e+f+g;/*Segment on for digit"8"*/
else
if ( num == 9)
 n=a+b+c+d+f+g;/*Segment on for digit"9"*/
else
 n=0;
outport(0x378,n);
}
/* END */
```

❑

ECONOMICAL UPS FOR CORDLESS PHONES

T.K. HAREENDRAN

Cordless telephone base sets are pow-ered from AC mains using small AC adaptors. In the event of mains failure, a low-capacity UPS will do the job satisfactorily. Here's an efficient, economical, and easy-to-construct UPS for cordless telephones.

When mains is available, a relay forming part of the circuit energises and the mains input voltage is connected to the primary of transformer via normally-open (N/O) RL(a) contacts. Simultaneously, the secondary output of the transformer is available across bridge rectifier comprising diodes D2 to D5 and LED1 glows. The battery starts charging if switch S2 is on.

Fig. 1 shows the block diagram of the UPS for cordless telephones. Normally, cordless phones require small-capacity adaptors (9V/12V, 500mA) to enable the operation of the circuit and to charge the battery in the handset of the cordless. The output power of this UPS circuit is limited to around 2 watts, which is sufficient to operate most cordless telephones. The functional block diagram of IC CD4047B used in the oscillator section of the UPS is shown in Fig. 2.

The circuit of the UPS for cordless telephone is shown in Fig. 3. Pin configurations of MOSFET IRF Z44N, SCR BT169, transistor BD139, and optocoupler IC PC817 are shown in Fig. 4.

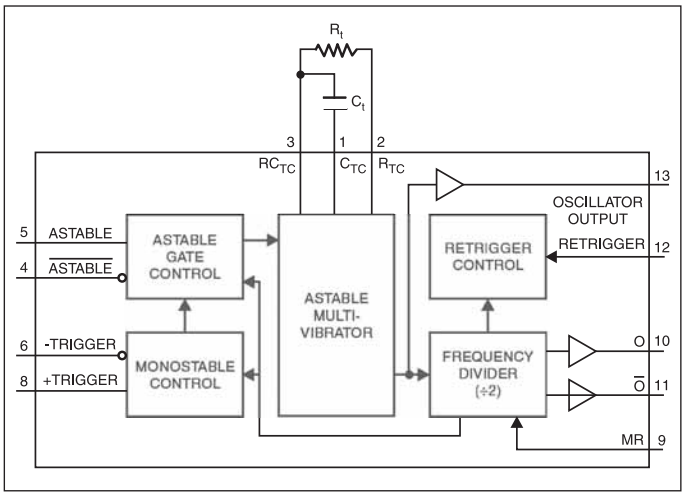


Fig. 2: Functional block diagram of IC CD4047B

Circuit description

Optocoupler IC PC817 (IC1) senses the availability of input mains supply (230V AC). When mains (230V AC) is available, IC1 conducts and the current passes through R2, C2, diode D1, inbuilt LED of IC1, and R1. Output pin 4 of IC1 is connected to the inputs of gate N1 of IC2 as well as a 12V SMF battery. When

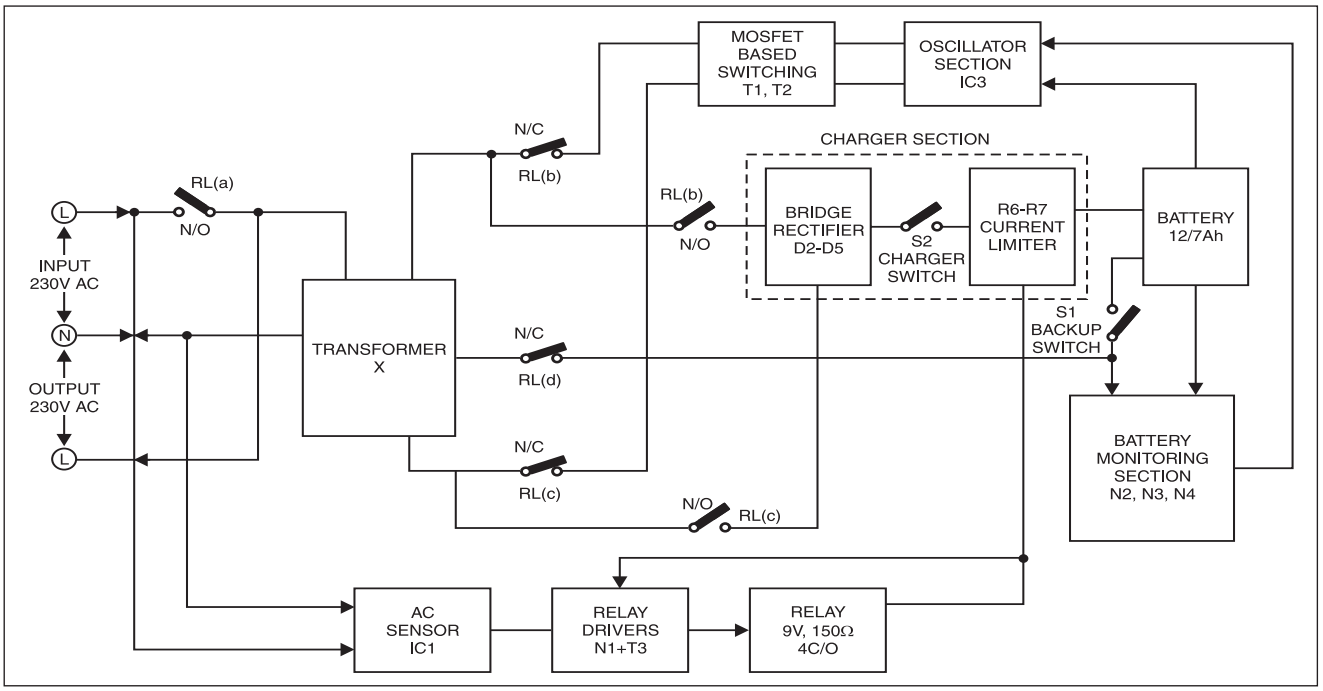


Fig. 1: Block diagram of the UPS for cordless telephones

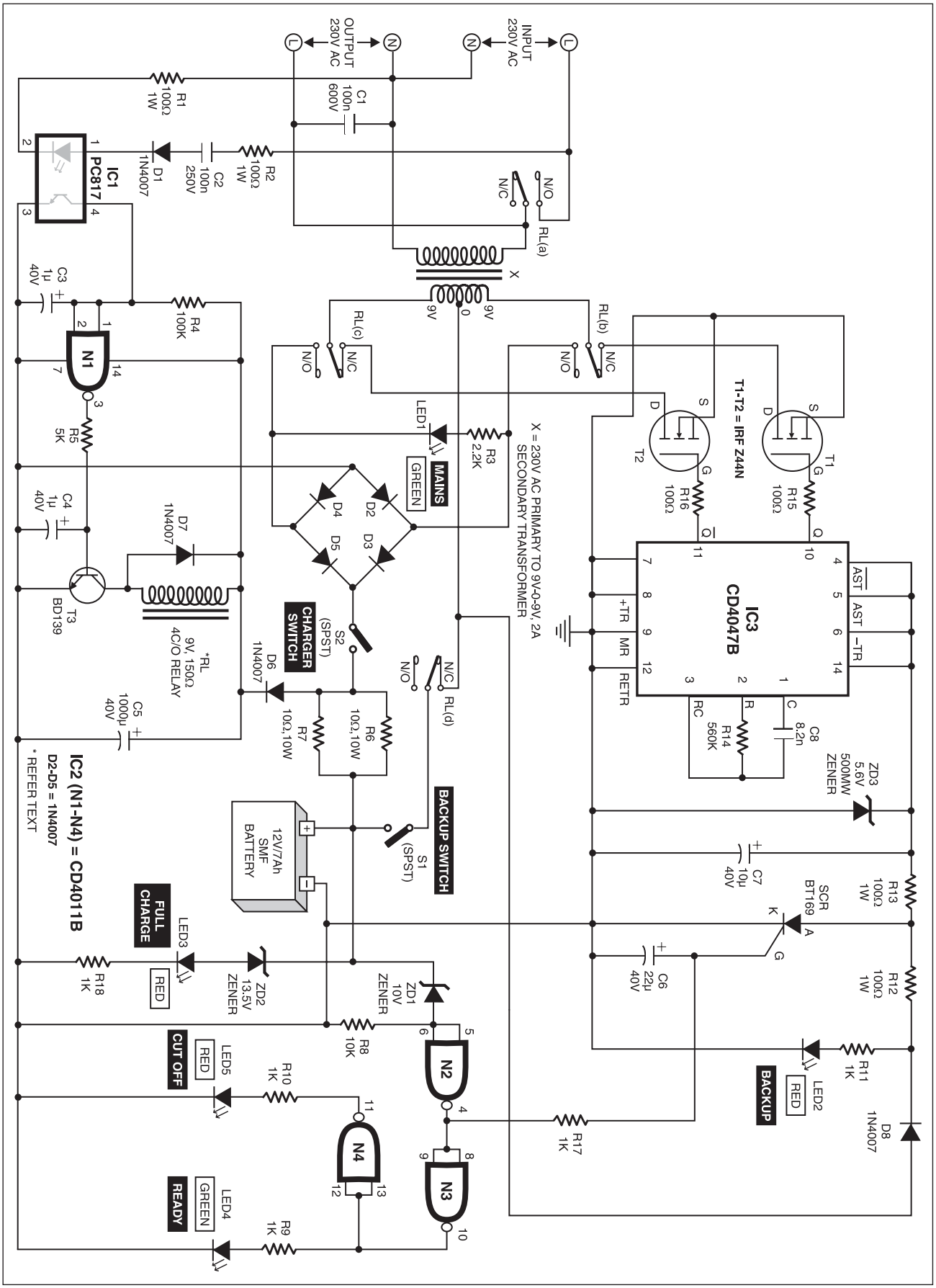


Fig. 3: Circuit of the UPS for cordless telephone

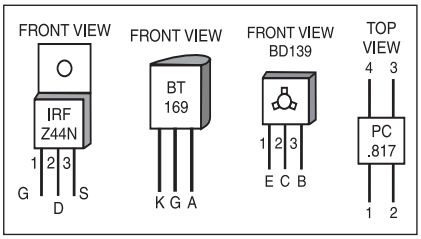


Fig. 4: Pin configurations of MOSFET IRF Z44N, SCR BT169, transistor BD139, and optocoupler IC PC817

IC1 conducts, pin 3 of gate N1 goes high and relay RL energises via driver transistor T3.

As a result:

1. The mains input AC supply connected to the primary of the transformer is transferred to the output of the UPS.

2. The step-down secondary windings of transformer X is connected to the bridge rectifier circuit (comprising diodes D2 through D5) through relay contacts RL(b) and RL(c). LED1 glows to indicate that mains is available. The battery charger circuit starts charging the battery, provided charging switch S2 is 'on' (closed).

3. The power supply to IC3 through relay contacts RL(d) is disconnected. As a result, the transformer works as a charger transformer and the battery starts charging through current-limiting resistors R6 and R7. LED1 glows to indicate the presence of AC supply.

Assume that backup switch S1 is in 'on' position. Now, in case the mains supply fails, relay RL instantly de-energises and the battery supply is connected to the inverter section, which inverts the DC voltage into AC voltage at the transformer X primary.

The inverter circuit wired around IC CD4047B (IC3) is an astable multivi-

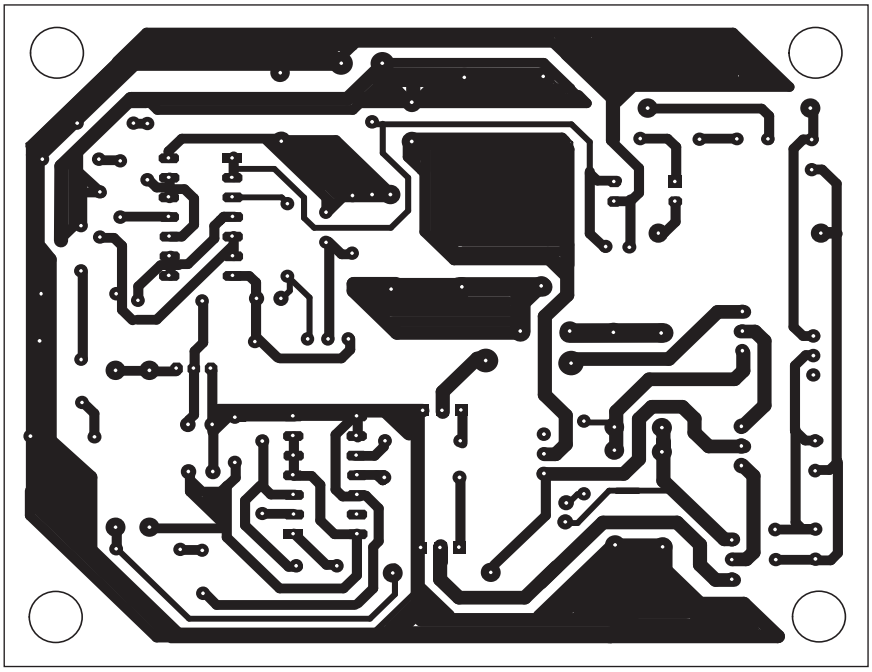


Fig. 5: Actual-size, single-side PCB layout for the UPS for cordless phones

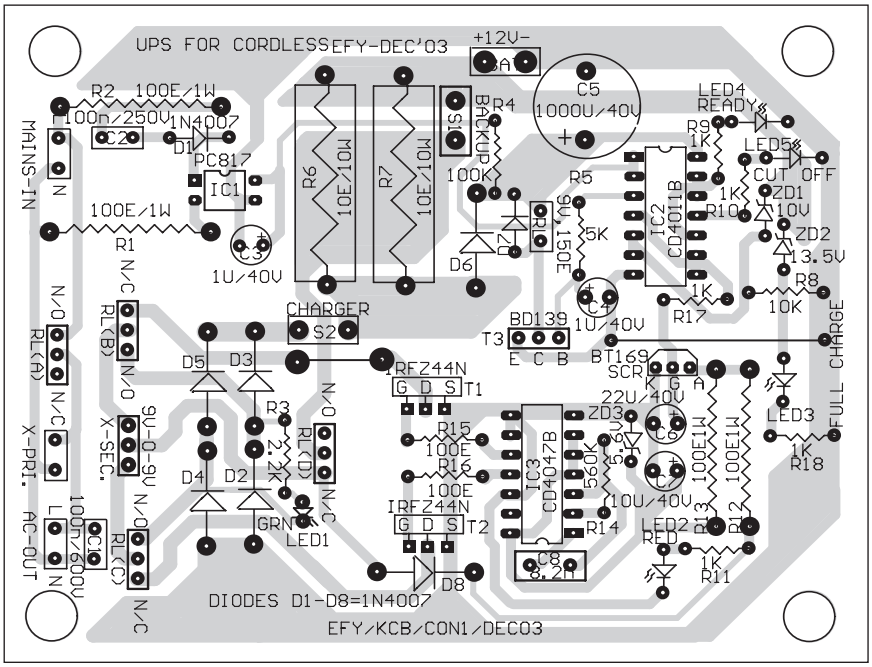


Fig. 6: Component layout for the PCB

PARTS LIST

Semiconductors:

IC1	- PC817 opto-coupler
IC2	- CD4011B quadruple 2-input NAND gate
IC3	- CD4047 mono/astable multivibrator
SCR	- BT169
T1-T2	- IRFZ44N n-channel MOSFET
T3	- BD139 npn transistor
D1-D8	- 1N4007 rectifier diodes
ZD1	- 10V zener diode
ZD2	- 13.5V zener diode
ZD3	- 5.6V zener diode
LED1, LED4	- Green LED
LED2, LED3	
LED5	- Red LED

Resistors (all ¼-watt, ±5% carbon, unless stated otherwise):

R1, R2, R12, R13	- 100-ohm, 1-watt
R3	- 2.2-kilo-ohm
R4	- 100-kilo-ohm
R5	- 5-kilo-ohm
R6, R7	- 10-ohm, 10-watt
R8	- 10-kilo-ohm
R9, R10, R11, R17, R18	- 1-kilo-ohm
R14	- 560-kilo-ohm
R15, R16	- 100-ohm

Capacitors:

C1	- 100nF, 600V polyster
C2	- 100nF, 250V polyster
C3, C4	- 1μF, 40V electrolytic
C5	- 1000μF, 40V electrolytic
C6	- 22μF, 40V electrolytic
C7	- 10μF, 40V electrolytic
C8	- 8.2nF ceramic disk

Miscellaneous:

X	- 230V AC primary to 9V-0-9V, 2A secondary transformer
RL	- 9V, 150-ohm, 4C/O relay
S1, S2	- SPST switch
	- 12V, 7Ah SMS battery
	- IC bases
	- Metal box for front panel

brator operating at a frequency of 50 Hz. The Q and $\overline{Q}$ outputs of IC3 directly drive the power MOSFETs (T1 and T2). The two MOSFETs are used in push-pull type configuration. The inverter output is filtered by capacitor C1.

NAND Gates N2, N3, and N4 of IC2 are used for monitoring and displaying the UPS system conditions. Zener diode ZD1 checks low voltage condition of the battery.

Whenever the battery voltage falls below 10V DC, output pin 4 of N2 goes from low to high state and the SCR is fired via resistor R17. As a result the supply voltage is pulled down to ground via RL(d), diode D8, resistor R12, and SCR, and the oscillator is disabled due to the absence of power supply to IC3. LED5 glows to indicate cut-off condition, i.e. the battery is out of use.

When the battery voltage is higher than 10 volts but less than 13.5 volts, output pin 4 of N2 remains low and the SCR does not fire. The resulting supply voltage is available to IC3. LED4 glows to indicate that the battery is ready for use.

When mains is available, the relay energises. Switch off backup switch S1 and keep charger switch S2 in 'on' position. The battery is now in charging mode. When the battery voltage increases above 13.5 volts, LED3 glows to indicate that the battery is fully charged. Now shift switch S2 to 'off' position to prevent overcharging of the battery.

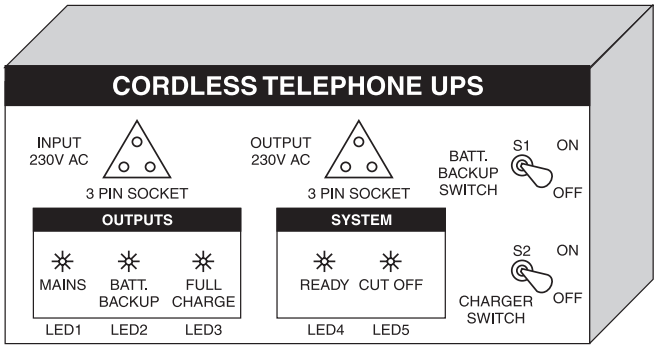


Fig. 7: Proposed front-panel layout including LEDs arrangement for the UPS for cordless telephones

The actual-size, single-side PCB layout for the circuit of UPS for cordless phones is shown in Fig. 5 and its components layout in Fig. 6.

The circuit can be easily assembled and placed inside a metal box. Mount the transformer on the chassis of the box. Also mount the battery in the box using supporting clamps. The proposed front-panel layout for the UPS, including LEDs arrangement, is shown in Fig. 7. Use suitable heat-sinks for MOSFETs. You may use two 9V, 300-ohm, 2c/o relays in parallel instead of a single 4c/o, 150-ohm relay.

The same circuit can be used for other applications as well to deliver higher power if you use a transformer with a higher current rating. ❑

Readers' comments:

Q. I have noted the following:

1. Optocoupler PC817 starts loading the circuit when it is in inverter mode, i.e. when the mains supply fails. Because of this, every time I use the circuit on battery, I have to remove the optocoupler. Please suggest some way out.
2. The positions of resistor R8 (10k) and zener diode ZD1 (10V) need to be interchanged. The reason: considering the 12V coming from the battery in the event of mains failure, and because of the current position of the zener diode, the input to NAND gate N2 (CD4011) will be about 2 volts. This voltage acts as a low logic and thereby fires the silicon-controlled register (SCR) every time the inverter section is 'on.' So the supply to the oscillator IC (CD4047) is pulled to ground and LED5 glows to indicate the cut-off condition. When I used the circuit before interchanging the positions, it didn't work. As soon as I changed the positions of the zener diode and the resistor, the circuit started working.

Chintan S. Pandya
chints45in@yahoo.com

EFY. Thank you very much for pointing out the mistakes. However, we don't agree with your first observation because the inverter circuit and the AC mains are separated through the relay. The positions of zener diode ZD1 and R8 need not be interchanged.

SECTION B :
CIRCUIT IDEAS

INTELLIGENT WATER PUMP CONTROLLER WITH WATER-LEVEL DISPLAY

JOHNSON MATHEW EASOW

Most of the circuits for multi-level indication/control of water in tanks employ a bunch of wires running between the circuit and the overhead tank, which accounts for almost half the cost of the entire project. Here is an intelligent scanned water-level indicator-cum-pump controller circuit (Fig. 1) that utilises just four wires to the overhead tank to indicate nine different levels. The connection arrangement for the overhead tank (OHT) and the underground tank (UGT) is shown in Fig. 2. Two wires from the circuit in Fig. 1 run to the underground/ground-level tank (to output line K and return line J, respectively) to check the availability of water in the tank before operating the pump, thereby guarding the pump against

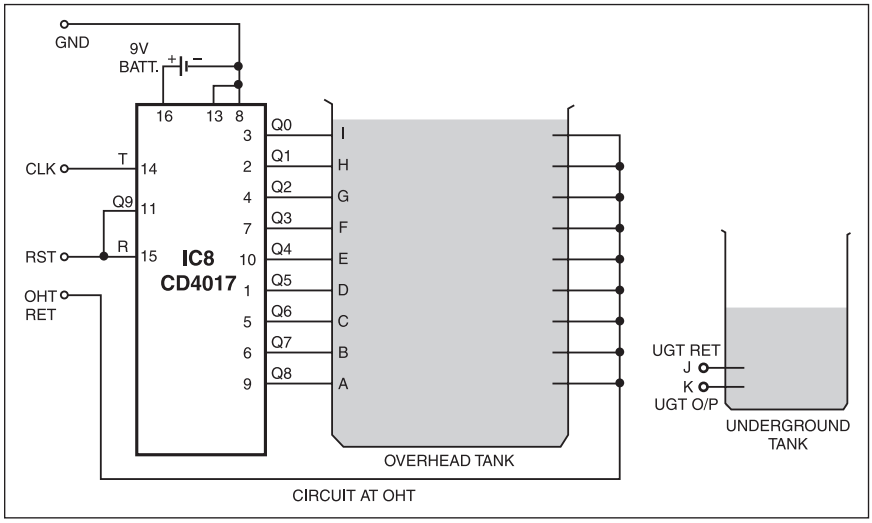


Fig. 2: Connection arrangement for overhead and underground tanks

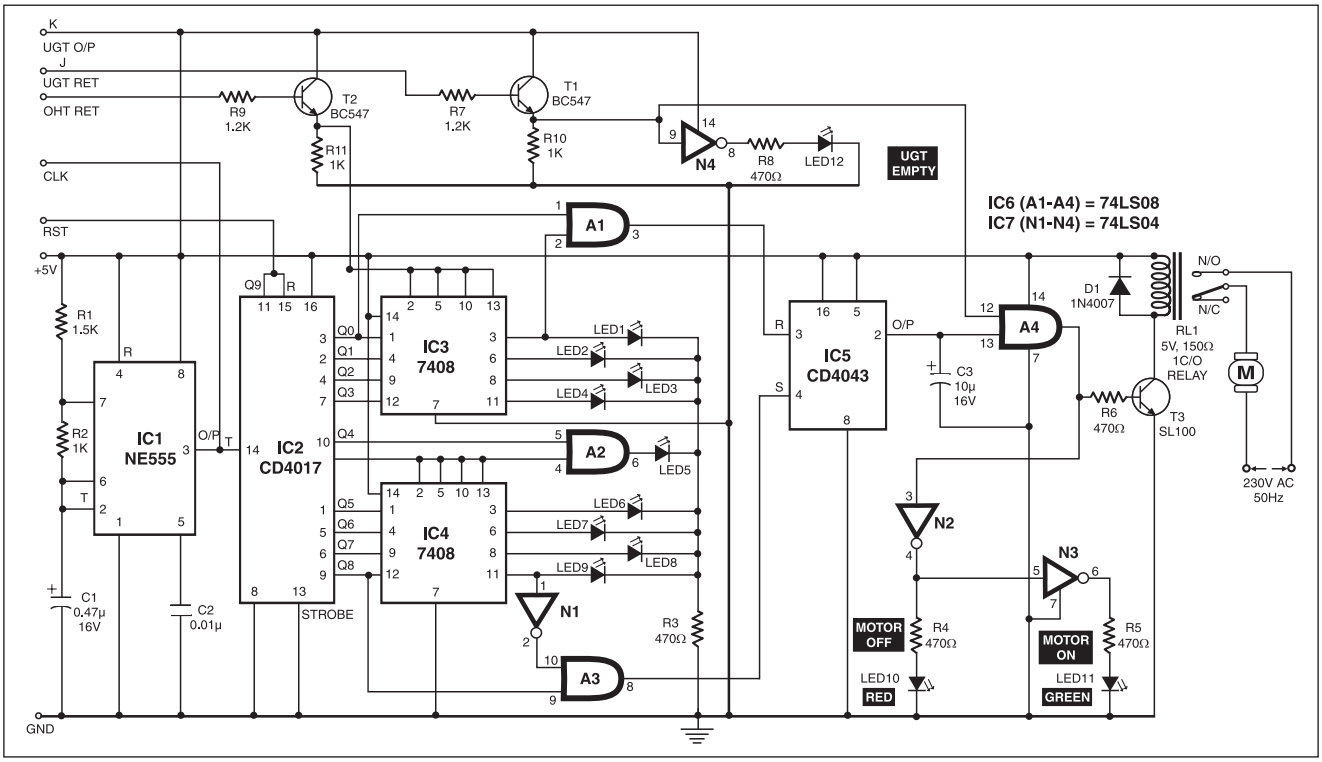


Fig. 1: Intelligent water-level indicator-cum-water pump controller

the damage due to dry running.

The scanning section employs an NE555 timer (IC1) wired as an astable multivibrator to oscillate at around 1 kHz. The output of NE555 is connected to CLK inputs of two CD4017 Johnson counters (IC2 and IC8). (IC8 is placed near the overhead tank in Fig. 2.)

Suppose at a given time, there is some specific water level in the OHT. The clock from NE555 keeps advancing the Q outputs of IC2 and IC8 starting from Q0. Only when the Q output of IC8 corresponding to the first (starting from top) water-submerged probe goes high, the OHT RET line goes high through water in the OHT. This causes pins 2, 5, 10, and 13 of quad AND gate ICs (IC3 and IC4) as well as one input of AND gate A2 to go high via emitter-follower transistor T2. The identical Q output of IC2 goes high simultaneously to light up the corresponding LED (LED1 through LED9) to indicate that particular level.

Similarly, upon reception of the next clock, the next lower level is indicated by the next LED, and so on. Scanning at a very high speed gives the illusion that all LEDs up to the one corresponding to the actual level in the OHT are continuously lit. This is due to the persistence of vision.

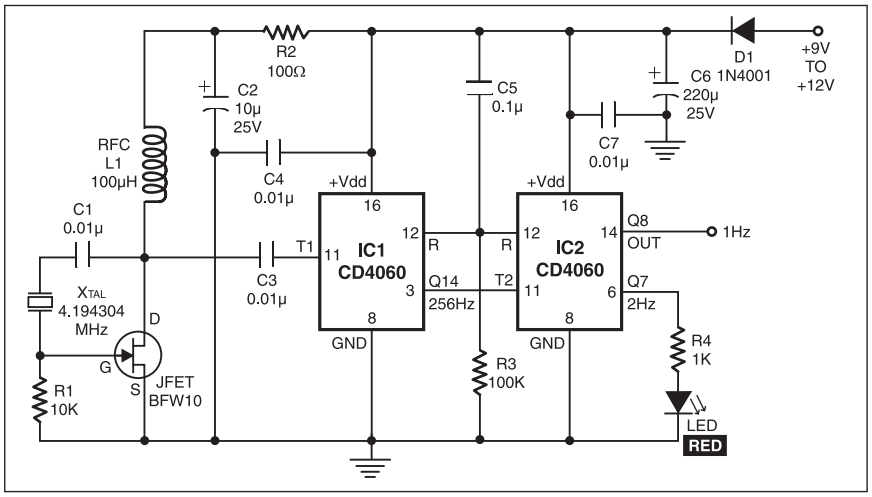
When the water level in the OHT is high enough to light up LED1, both the inputs of AND gate A1 also go high simultaneously. As a result, the output of AND gate A1 goes high to reset the flip-flop IC (IC5). The output pin 2 of IC5 goes low to de-energise relay RL1. Now when the water level in the OHT goes low such that LEDs 1 through 9 are off, the output of AND gate A3 goes high to set RS flip-flop (IC5), thereby making its output pin 2 high. Only when there is enough water in the UGT, pin 12 of AND gate A4 will be at logic 1 to provide forward bias to relay driver transistor SL100 (T3) to energise the relay to switch on the pump motor. The motor will switch off only when the water level reaches the uppermost level or when the UGT gets empty. LED10 through LED12 indicate 'motor off', 'motor on', and 'UGT empty', respectively. IC8 is powered separately, using a 9V battery that lasts long enough.

PRECISION 1HZ CLOCK

PRAVEEN SHANKER

This precision 1Hz clock uses just two ICs and one junction field-effect transistor (JFET) in conjunc-tion with a commonly available crystal and a handful of other passive components.

The Pierce-type crystal oscillator is formed around BFW10 n-channel JFET using the components as shown in the figure. A commonly available low-priced 4.194304MHz crystal is connected between its gate and drain via capacitor C1. A readymade 100μH radio frequency coil (RFC) forms the load for JFET at crystal frequency and also prevents the RF signal from entering the DC supply.



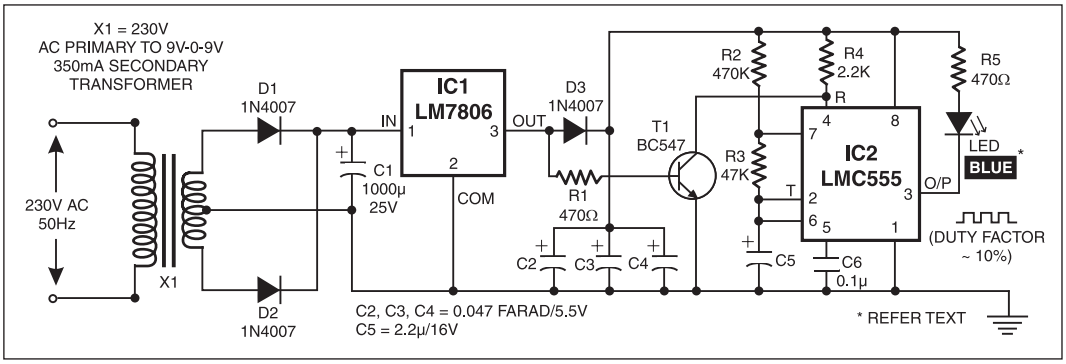
The 4.194304MHz output from the drain is fed to input pin 11 of the 14-stage binary counter-cum-oscillator CD4060 (IC1). The output from pin 3 of IC1 is equal to the crystal frequency divided by 16,384, i.e. 4.194304 MHz/16,384 = 256 Hz, which is connected to pin 11 of another IC CD4060 (IC2). The Q8 output from pin 14 of IC2 further divides the input frequency by 256 to give 1Hz pulses (with 50 per cent duty cycle). The LED connected to pin 6 (Q7) of IC2 flickers at the rate of 2 Hz ('on' time = 0.25 second and 'off' time = 0.25 second), indicating that the circuit is functioning properly.

BLUE LED NIGHT LAMP WITH BACK-UP

T.K. HAREENDRAN

Here's an attractive night lamp cir-cuit that can be easily built using readily available electronic components.

When 230V AC, 50Hz AC mains voltage is fed to the circuit, it is stepped down by transformer X1 to provide about 9V DC supply to three-pin regulator IC LM7806 (IC1) after rectification by diodes D1 and D2 and smoothing capacitor C1. The 6V regulated DC output voltage is then applied to the parallel combination of three 47000μF,



5.5V DC high-capacity capacitors (C2, C3, and C4) via diode D3 and the capacitors start charging.

The next part of the circuit comprises a conventional astable (free-running) multivibrator wired around IC LMC555 (IC2). Here IC LMC555, which is the CMOS version of the standard 555 series, is used deliberately to save power consumption. The IC consumes power of less than 1 mW at 5V supply.

When mains supply is present, transistor T1 gets forward biased via resistor R1 and its collector is thus pulled towards ground. This, in turn, grounds reset pin 4 of IC2. Hence the astable is disabled and the blue LED (LED1) at its output pin 3 glows steadily. Resistor R5 limits the LED current to a safe value. (***Note.*** One may use any LED in place of blue LED.)

When mains supply stops, the astable is powered for a short time by the reservoir capacitor combination of C2, C3, and C4 and the LED starts blinking to save power. The blinking continues for about one minute, to allow the user to switch on the emergency power or search for an alternative light source.

The back-up time of this circuit is dependent on the value of reservoir capacitors. You may replace C2-C3-C4 combination with a single 1F, 5.5V capacitor for better results. (***Note.*** Changing the LED blinking rate affects the back-up time duration.)

The major advantages of capacitors in Farad range are their small size, low weight, and minimum mounting space requirements. The high-value capacitors used in this circuit are widely used in memory circuits of cordless phones, timer circuits of cassette recorders, and non-volatile memories of certain TVs, hence you can procure the same from dealers of the mentioned equipment. Alternatively, you can procure a 0.22F, 5.5V capacitor from Matsushita Electric (Japan) or take one piece from an old VCR (preferably from National Panasonic or Matsushita).

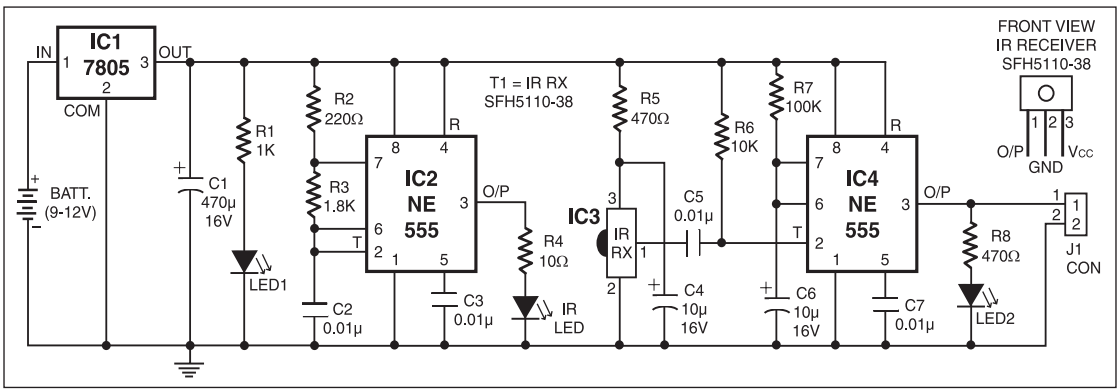
INFRARED PROXIMITY DETECTOR

K.S. SANKAR

This proximity detector using an infrared detector (Fig. 1) can be used in various equipment like automatic door openers and burglar alarms. The circuit primarily consists of an infrared transmitter and an infrared receiver.

The transmitter section consists of a 555 timer IC functioning in astable mode. It is wired as shown in the figure. The output from astable is fed to an infrared LED via resistor R4, which limits its operating current. This circuit provides a frequency output of 38 kHz at 50 per cent duty cycle, which is required for the infrared detector/receiver module. Siemens SFH5110-38 is a much better choice than SFH506-38. Siemens SFH5110-38 is turned on by a continuous frequency of 38 kHz with 50 per cent duty cycle, whereas SFH506 requires a burst frequency of 38k to sense. Hence, SFH5110-38 is used.

The receiver section comprises an infrared receiver module, a 555 monostable multivibrator, and an LED indicator. Upon reception of infrared signals, 555 timer (mono) turns on and remains on as long as infrared signals are received. When the signals are interrupted, the mono goes off after a few seconds (period = 1.1 R7xC6) depending upon the value of R7-C6 combination. Thus if R7 = 470 kilo-ohms and C6 = 4.7μF, the mono period will be around 2.5 seconds.



Both the transmitter and the receiver parts

can be mounted on a single breadboard or PCB. The infrared receiver must be placed behind the infrared LED to avoid false indication due to infrared leakage.

An object moving nearby actually reflects the infrared rays emitted by the infrared LED. The infrared receiver has sensitivity angle (lobe) of 0-60 degrees, hence when the reflected IR ray is sensed, the mono in the receiver part is triggered. The output from the mono may be used in any desired fashion. For example, it can be used to turn on a light when a person comes nearby by energising a relay. The light would automatically turn off after some time as the person moves away and the mono pulse period is over.

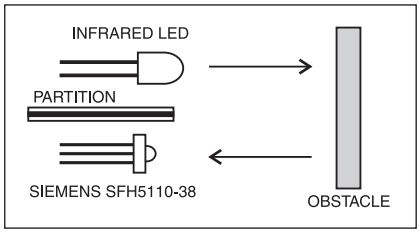


Fig. 2: Proposed arrangement for separation of IR LED and receiver module in the proximity detector

The sensitivity of the detector depends on current-limiting resistor R4 in series with the infrared LED. Range is approximately 40 cm. For 20-ohm value of R4 the object at 25 cm can be sensed, while for 30-ohm value of R4 the sensing range reduces by 22.5 cm.

(***Note.*** The author procured the samples of Siemens products from Arihant Electricals, New Delhi, the distributor of Siemens in India.)

Readers Comments:

❑ My problem is that it has a range of about 20cm only. I am using IR receiver TSOP1738 in place of SFH-5110-38. Also guide me to make an object counter for use in door-opening systems.

Basarat Ahmed Ustad
Through e-mail

The author K.S. Sankar replies:

It is important to use SFH-5110-38K Siemens IC in this project because the transmitter provides a continuous beam of 38k and not burst frequencies. If you want to use SFH506-38K, use another 555 timer at about 100Hz square wave and connect the output to pin 4 of the 38k oscillator—this will provide the burst frequency.

It is very important to shield the IR LED with a 6mm dia. aluminium tube of 2.54cm (1-inch) length to produce a narrow beam for better results. Cover the back of the IR LED with red wax so that there is no leakage of light. The 3-pin IR receivers are very sensitive.

For use as an object counter, the output can be connected to IC 4017 CMOS decade counter to indicate the number of pulses, or to a BCD counter and then to a BCD-to-7-segment display unit to see the output count as shown here in Fig. 1.

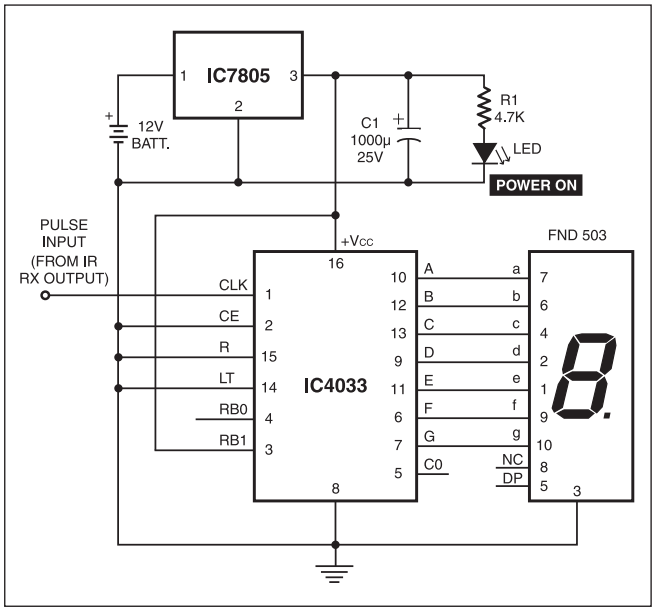


Fig. 1: Object counter

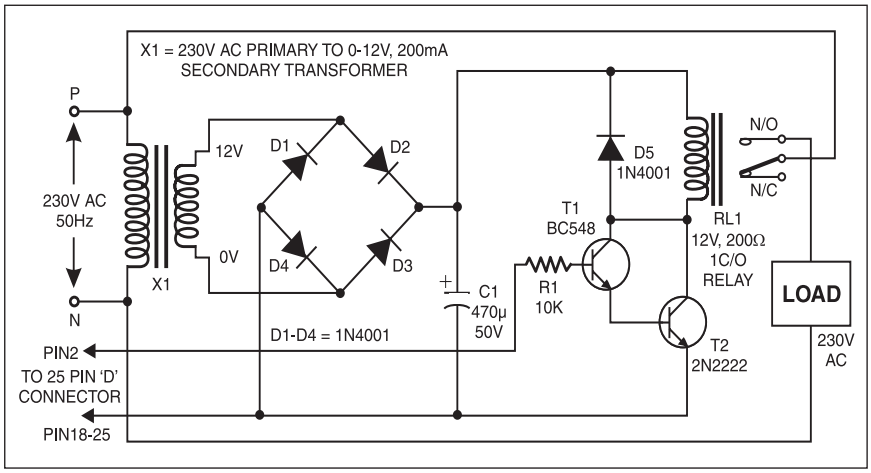
COMPUTERISED UNIVERSAL TIMER

D.K. KAUSHIK

This simple and flexible timer is more accurate than the real-time clock of the computer used for the purpose. It can be used in laboratories, dark rooms, kitchens, and for competitions in educational institutes. The program written in Q-Basic is self-explanatory.

Generally, a universal timer provides the facility for switching on an electrical/electronic device after elapse of a certain time period, say, 5 minutes. The software does the same job here.

When the program is executed, it displays 0:0:0 on the monitor, indicating 0 hour, 0 minute, and 0 second. The dis-



play time 0:0:0 is increased by 10 seconds each time function key F1 of the computer keyboard is depressed. So by depressing function key F1 the required time is set for which the electrical or electronic device is to be switched on. However, in debate competitions the time allowed for a candidate to speak is filled the way it is discussed above. The program may be changed as indicated by REM statements and the single quote (') in the beginning of a program line may be accordingly removed in the program.

Now, after setting the time in the manner as discussed above, function key F2 of the computer keyboard is depressed to switch on any device. Simultaneously, the countdown of the time in the display box starts. The device will remain on until the display box shows 0:0:0 and then it will get switched off. The figure shows the relay interface circuit connected between D0 line (pin 2) and ground line (pin 25) of 378H output port of LPT1 printer port of the computer.

COMPUTER PROGRAM IN Q-BASIC

```
CLS : SCREEN 1: COLOR 3, 10
N1 = 0: N2 = 0: N3 = 0
LOCATE 4, 10: PRINT " PROGRAM DEVELOPED BY DR.D.K.KAUSHIK"
LINE (90, 60)-(200, 60), 2: LINE -(200, 75), 2
LINE -(90, 75), 2: LINE -(90, 60), 2
LOCATE 9, 14: PRINT N3; ":"; N2; ":"; N1
LOCATE 13, 9: PRINT "PRESS F1 KEY FOR TIME ADJUSTMENT"
LOCATE 15, 9: PRINT "PRESS F2 KEY FOR START"
LOCATE 17, 9: PRINT "PRESS F3 KEY TO STOP"
KEY(2) ON: ON KEY(2) GOSUB START
KEY(3) ON: ON KEY(3) GOSUB LAST1
PORT% = &H378
DELAY:
KEY(1) ON: ON KEY(1) GOSUB SET
GOTO DELAY
SET:
SOUND 650, 3
N1 = N1 + 10
IF N1 < 60 THEN LOCATE 9, 14: PRINT N3; ":"; N2; ":"; N1: RETURN
N1 = 0
N2 = N2 + 1
IF N2 < 60 THEN LOCATE 9, 14: PRINT N3; ":"; N2; ":"; N1: RETURN
N2 = 0: N3 = N3 + 1
LOCATE 9, 14: PRINT N3; ":"; N2; ":"; N1
RETURN
START:
KEY(1) OFF: KEY(2) OFF
Z1 = VAL(RIGHT$(TIME$, 2))
OUT PORT%, 1
REM FOR DEBATE COMPETITION OUT PORT%,1 SHOULD BE REPLACED
REM BY OUT PORT%,0
START1:
Z2 = VAL(RIGHT$(TIME$, 2))
IF Z2 = Z1 THEN GOTO START1 ELSE GOTO START2
START2:
IF N1 = 0 AND N2 = 0 AND N3 = 0 THEN LOCATE 9, 14
PRINT N3; ":"; N2; ":"; N1: GOTO LAST
N1 = N1 - 1
IF N1 < 0 THEN N1 = 59: N2 = N2 - 1
IF N2 < 0 THEN N2 = 59: N3 = N3 - 1
IF N3 < 0 THEN N3 = 0
LOCATE 9, 14: PRINT N3; ":"; N2; ":"; N1
Z1 = Z1 + 1
IF Z1 = 60 THEN Z1 = 0
IF N1 = 0 AND N2 = 0 AND N3 = 0 THEN GOTO LAST ELSE GOTO
START1
LAST:
OUT PORT%, 0
REM FOR DEBATE COMPETITION OUT%,0 SHOULE BE REPLACED
BY OUT PORT%,1
FOR T = 1 TO 3
SOUND 550, 17
NEXT T
' FOR J=1 TO 10
'FOR J=1 TO 50000 : NEXT J : NEXT I
'OUT PORT%,0
END
LAST1:
OUT PORT%, 0
END
```

Readers Comments:

❑ I have constructed the circuit but found no output at the parallel port. The interface circuit is correct and the software seems to be the souce of problem.

When the program is executed, it displays 0:0:0 on the monitor. I increased it to 10 seconds using F1 key after pressing the function key F2 to start the count. However, even after 10 seconds, there was no output at the parallel port. I tried by setting different timings but it failed to respond. I checked the circuit by giving external input to the 25-pin D connector and found it working well. Please check the program and suggest me an alternative.

J. Sunil
Trichy

The author Dr D.K. Kaushik replies:

I thank Mr Sunil for showing interest in my article. As regards his queries, the program and the circuit diagram in my article are correct. The problem that the reader is facing is either due to the wrong circuit or due to the wrong connection to the printer port.

I advise Mr Sunil to first check whether the printer port is giving proper output voltage. To check its working, write a small program as given below and execute:

```
PORT%=& H378
OUT PORT%,1
END
```

After execution of this program, measure the voltage at pin 2 w.r.t. pin 25 of the printer port. If this voltage is between 4 and 5 volts, the printer port is working fine. If no voltage is received, the printer port is not working, so get it checked from some computer hardware expert. If the voltage is proper as mentioned above, connect the circuit to the printer port. The relay should now energise.

In case the measured voltage is less than 4 volts, reduce the value of resistor R1 from 10 kilo-ohms to 4.7 kilo-ohms. The relay will then energise.

In case the voltage is between 4 and 5 volts as mentioned above and even then the relay is not energised, check the relay circuit. The relay and/or transistors of the circuit may be damaged. Check these by applying 5 volts to resistor R1 w.r.t. ground pin of the circuit. The relay should energise.

After you've thoroughly checked the circuit as discussed above and made necessary corrections, the program will definitely work.

NUMBER GUESSING GAME

PRIYANK MUDGAL

This number guessing game is quite simple. In this game the player thinks of any number between 1 and 99. Then he scans the eight groups of numbers given in the eight boxes in the table. Each group corresponds to a specific switch (indicated on the top of each group) on an 8-way DIP switch. The person scans the numbers in each box and slides the switch corresponding to a box to 'on' position if he finds his number in that box. After having scanned all the eight boxes and switching on the relevant DIP switches, he is required to press switch S9 and the number thought of by the person is displayed on the 7-segment displays. After this, all switches on the 8-way DIP switch need to be turned off to try display of another number in a similar fashion.

The circuit (Fig. 1) comprises two BCD-to-7-segment decoder/driver CD45 11 ICs (IC1 and IC2). IC1 generates the number for tens position and IC2 generates the number for units position. Input pins 7, 1, 2, and 6 of both the ICs are connected to ground through 1-kilo-ohm resistors. The common-cathode terminals of both the displays are connected to push-to-on switch S9.

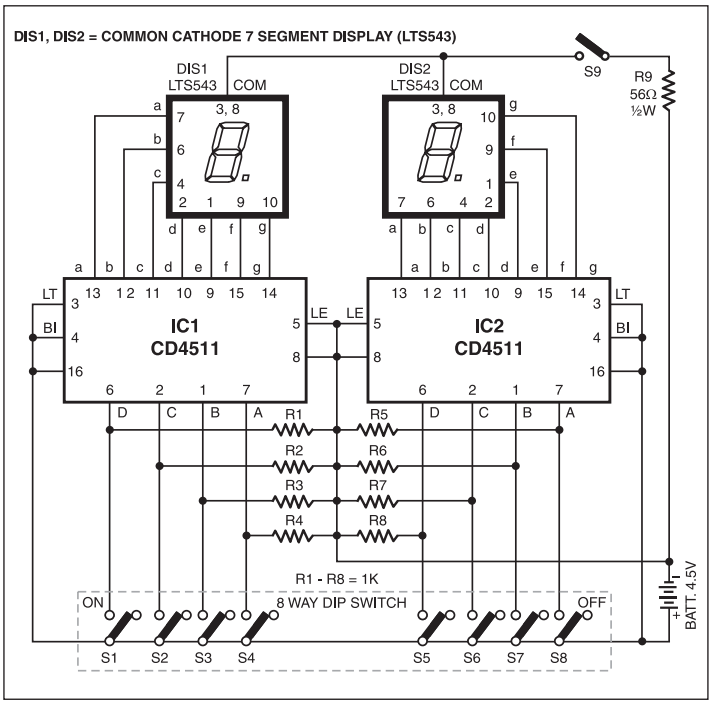


Fig. 1: Number guessing game circuit

Suppose you want to display 47. For this, 4 is to be displayed in tens position and 7 in units position. In order to generate 4 (binary 100) on the display (DIS1), switch S2 is to be turned on. To display 7 (binary 111) on the display (DIS2), switches S6, S7, and S8 are to be turned on. Thus to generate 47, switches S2, S6, S7, and S8 are to be turned on. The number 47 is placed in groups 6, 7, 8, and 2. So when you spot 47 in these groups, switch on the same combination of switches. On depressing switch S9, 47 appears on the display. Other numbers can be generated using the same procedure.

In order to make the circ-uit compact, a DIP switch has been used here. As it may be diff-icult to turn the small swi-tches on and off, you may use SPDT tog-gle switches in place of the DIP switch. The circuit can be placed inside a plastic case with appropriate cuts made for disp-lays and switc-hes (Fig. 2). A strip of paper containing groups of numbers can be stuck just under the 8-way DIP switch (or under the row of SPDT switches used in place of DIP switch). The proposed cabinet with front-panel layout is shown in the figure.

This circuit smoothly runs on two pen torch batteries. Thus current-limiting resistors are not necessary for displays.

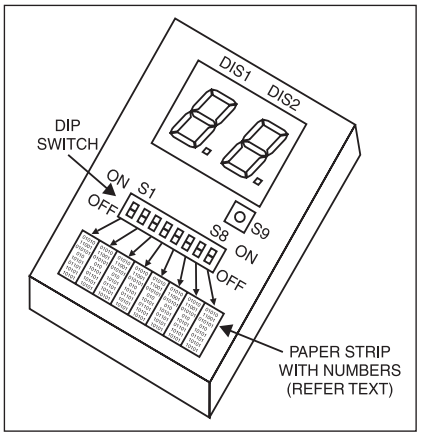


Fig. 2: Suggested case

Eight Groups of Numbers and Their Respective Switches

Switch 1	Switch 2	Switch 3	Switch 4
80, 81, 82, 83, 84, 85, 86, 87, 88, 89 90, 91, 92, 93, 94, 95, 96, 97, 98, 99	40, 41, 42, 43, 44, 45, 46, 47, 48, 49 50, 51, 52, 53, 54, 55, 56, 57, 58, 59 60, 61, 62, 63, 64, 65, 66, 67, 68, 69 70, 71, 72, 73, 74, 75, 76, 77, 78, 79	20, 21, 22, 23, 24, 25, 26, 27, 28, 29 30, 31, 32, 33, 34, 35, 36, 37, 38, 39 60, 61, 62, 63, 64, 65, 66, 67, 68, 69 70, 71, 72, 73, 74, 75, 76, 77, 78, 79	10, 11, 12, 13, 14, 15, 16, 17, 18, 19 30, 31, 32, 33, 34, 35, 36, 37, 38, 39 50, 51, 52, 53, 54, 55, 56, 57, 58, 59 70, 71, 72, 73, 74, 75, 76, 77, 78, 79 90, 91, 92, 93, 94, 95, 96, 97, 98, 99
Switch 5	**Switch 6**	**Switch 7**	**Switch 8**
8, 9, 18, 19, 28, 29, 38, 39 48, 49, 58, 59, 68, 69, 78, 79 88, 89, 98, 99	4, 5, 6, 7, 14, 15, 16, 17, 24, 25, 26, 27 34, 35, 36, 37, 44, 45, 46, 47, 54, 55 56, 57, 64, 65, 66, 67, 74, 75, 76, 77 84, 85, 86, 87, 94, 95, 96, 97	2, 3, 6, 7, 12, 13, 16, 17, 22, 23, 26, 27 32, 33, 36, 37, 42, 43, 46, 47, 52, 53, 56, 57, 62, 63, 66, 67, 72, 73, 76, 77, 82, 83, 86, 87, 92, 93, 96, 97	1, 3, 5, 7, 9, 11, 13, 15, 17, 19 21, 23, 25, 27, 29, 31, 33, 35, 37 39 41, 43, 45, 47, 49, 51, 53, 55, 57, 59 61, 63, 65, 67, 69, 71, 73, 75, 77, 79 81, 83, 85, 87, 89, 91, 93, 95, 97, 99

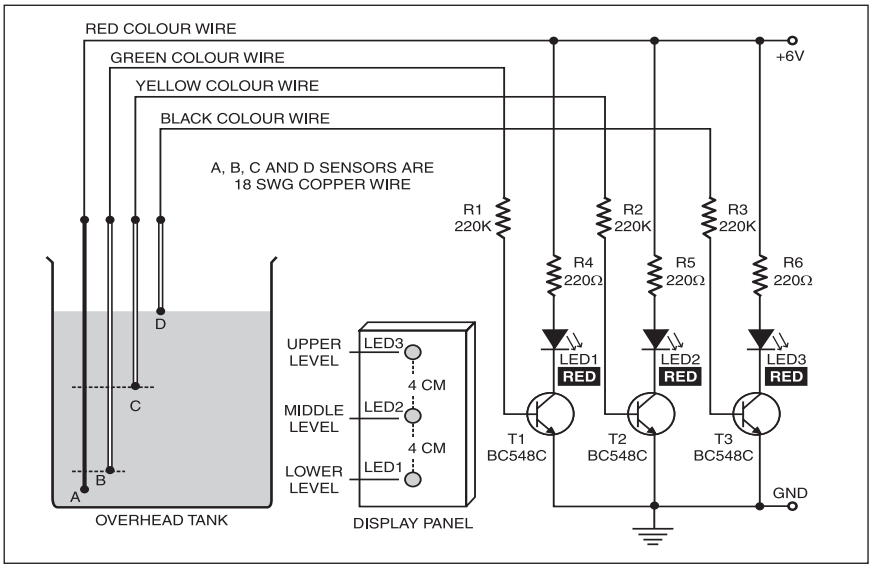
WATER-LEVEL INDICATOR

P. VENKATA RATNAM

Here's a simple water-level indicator for overhead tanks that uses three LEDs (LED1, LED2, and LED3) to indicate minimum, middle, and maximum water levels in the tank.

The sensor probes comprise A, B, C, and D, where A is the common probe and B, C, and D are meant for sensing the minimum, middle, and maximum levels, respectively. When water in the tank touches sensor wires A and B both, a small current passes from A to B through water and to the base of transistor T1 via resistor R1. As a result, transistor T1 conducts, causing LED1 to glow. Similarly, when water touches sensor C, LED2 glows to indicate that the water has reached the middle level. Finally, when water touches sensor D, LED3 glows to indicate the maximum level of water. Thus all the three LEDs glow when the tank is full. At this stage, the motor should be switched off manually.

The circuit can be easily assembled on a general-purpose PCB and enclosed in a wooden box. The three LEDs should be mounted on the front panel of the box with a spacing of about 4 cm between them. Short lengths of four 18 SWG copper wires may be used for sensor probes. For the common sensor A, a bare copper wire of 18 SWG should be used. For sensors B, C, and D three single-core PVC wires should be used, with their insulation removed to a length of one centimetre towards the ends. All the four wires may be tied around a 12.5mm dia. PVC tube with nylon thread at different heights, without touching each other (not shown in figure).



The sensor probes should be kept in the tank vertically and connected to the main circuit using four flexible PVC wires of different colours.

The circuit is powered by a battery eliminator or a 6V battery and kept near the motor switchboard. The current drawn by the circuit, when all the LEDs glow, is up to 50 mA, which is less than the current drawn by a 6V bed-lamp.

ULTRA-BRIGHT LED LAMP

N.S. HARISANKAR VU3NSH

This ultra-bright white LED lamp works on 230V AC with minimal power consumption. It can be used to illuminate VU meters, SWR meters, etc.

Ultra-bright LEDs available in the market cost Rs 8 to 15. These LEDs emit a 1000-6000mCd bright white light like welding arc and work on 3 volts, 10 mA. Their maximum voltage is 3.6 volts and the current is 25 mA. Anti-static precautions should be taken when handling the LEDs. The LEDs in water-clear plastic package emit spotlight, while diffused type LEDs have a wide-angle radiation pattern.

This circuit (Fig. 1) employs capacitive reactance for limiting the current flow through the LEDs on application of mains voltage to the circuit. If we use only a series resistor for limiting the current with mains operation, the limiting resistor itself will dissipate around 2 to 3 watts of power, whereas

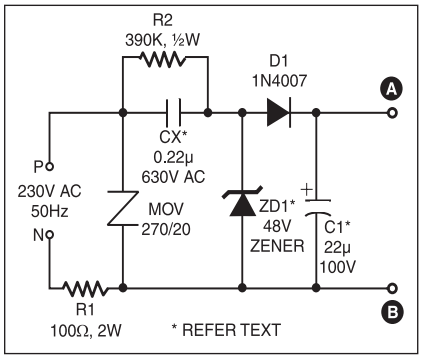


Fig. 1: The circuit of ultra-bright white LED lamp

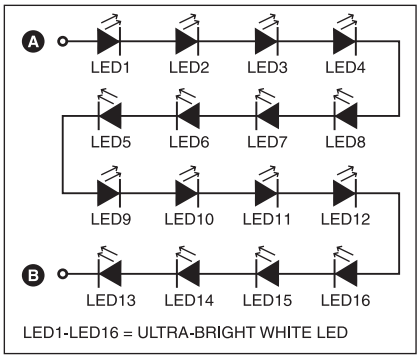


Fig. 2: 16-LED combination

no power is dissipated in a capacitor. The value of capacitor is calculated by using the following relationships:

$X_C = 1/(2\pi fC)$ ohms ————(a)

$X_C = V_{RMS}/I$ ohms ———— (b)

where X_C is capacitive reactance in ohms, C is capacitance in farads, I is the current through the LED in amperes, f is the mains frequency in Hz, and Vrms is the input mains voltage.

The 100-ohm, 2W series resistor avoids heavy 'inrush' current during transients. MOV at the input prevents surges or spikes, protecting the circuit. The 390-kilo-ohm, ½-watt resistor acts as a bleeder to provide discharge path for capacitor Cx when mains supply is disconnected. The zener diode at the output section prevents excess reverse voltage levels appearing across the LEDs during negative half cycles. During positive half cycle, the voltage across LEDs is limited to zener voltage.

Use AC capacitors for Cx. Filter capacitor C1 across the output provides flicker-free light. The circuit can be enclosed in a CFL round case, and thus it can be connected directly to AC bulb holder socket. A series combination of 16 LEDs (Fig. 2) gives a luminance (lux) equivalent of a 12W bulb. But if you have two series combinations of 23 LEDs in parallel (total 46 LEDs as shown in Fig. 3), it gives light equal to a 35W bulb. 15 LEDs are suitable for a table-lamp light.

Diode D1 (1N4007) and capacitor C1 act as rectifying and smoothing elements to provide DC voltage to the row of LEDs. For a 16-LED row, use Cx of 0.22 μF, 630V; C1 of 22 μF, 100V; and zener of 48V, 1W. Similarly, for 23+23 LED combination use Cx of 0.47 mF, 630V; C1 of 33 μF, 150V; and zener of 69V, 1W.

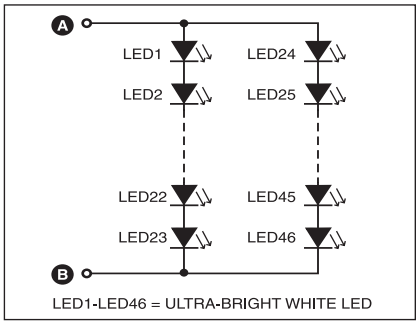


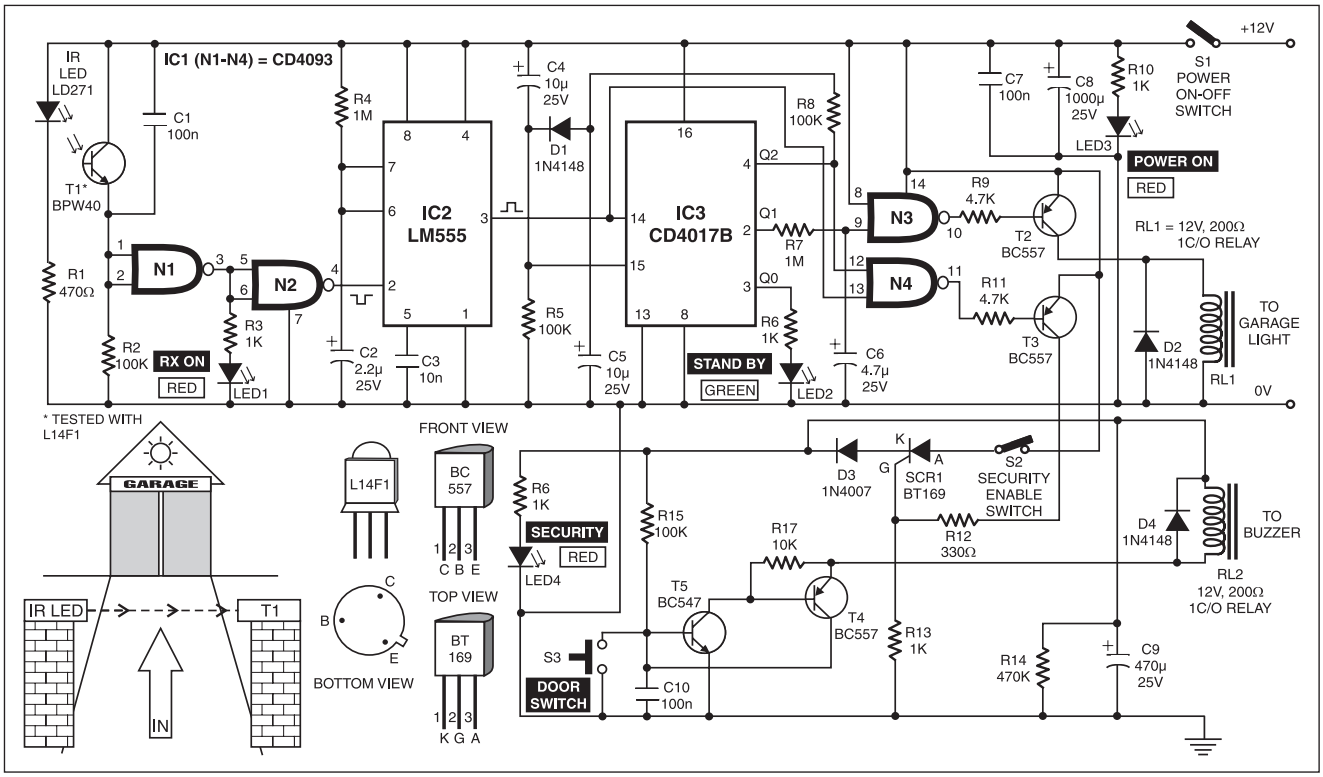
Fig. 3: 46-LED combination

GARAGE LIGHT AND SECURITY CONTROL

T.K. HAREENDRAN

Useful for vehicle owners, this gad-get automatically turns on indoor/outdoor garage lights and raises an alert when an automobile enters the garage.

Assume switch S2 is in 'on' (closed) state. When power switch S1 is turned on, the complete circuit is energised by the 12V DC supply. LED3 lights up to provide power-on indication. Simultaneously, IC3 (CD4017B) is instantly reset by the power-on-reset circuit formed by the combination of capacitor C4 and resistor R5, and green LED2 lights up as a standby indicator. As per the physical arrangement, IR rays from IR-LED fall on phototransistor T1 and it conducts to pull up the inputs of NAND



gate N1 (used here as an inverter) to logic 1. As a result, the output of gate N2 goes high to make the monostable built around IC2 inactive.

Now, when a vehicle moves through the door, the IR beam is interrupted and the output state of gate N2 changes from high to low state, which triggers the monostable and red LED1 (Rx on) lights up briefly. The output of monostable provides clock pulse to IC3, which changes its output state, with its pin 2 going high and pin 3 going low. As a result, standby indicator green LED2 goes off and relay driver pnp transistor T2 gets forward biased via gate N3 to energise relay RL1. The contacts of relay RL1 can be used to switch indoor and outdoor garage lights.

After parking the vehicle, when the owner moves through the passage to interrupt the light beam once again, the monostable (IC2) is retriggered and the output state of IC3 changes again. This time, the output at pin 2 of IC3 goes low, while the output at its pin 4 goes high. This output resets IC3, after a short delay determined by components R8, C5, and D3. Standby LED2 again lights up.

Before the resetting function, the security system drive circuit is activated via gate N4 as follows: During retriggering, both inputs (pins 12 and 13) of gate N4 are at high logic level, taking its output pin 11 low to forward bias pnp transistor T3 via resistor R11. As a result, the SCR (BT169) is triggered via R12 and latched. Now the DC supply is extended to the rest of the circuit via the SCR until it is reset by disabling switch S2.

Door switch S3 is N/O type and it opens only when the door is opened. This triggers the regenerative pair of transistors T4 and T5, and relay RL2 is energised (and latched). Contacts of relay RL2 may be connected to an emergency beeper, a high-power signalling device, or an automatic telephone dialer, as desired by the user.

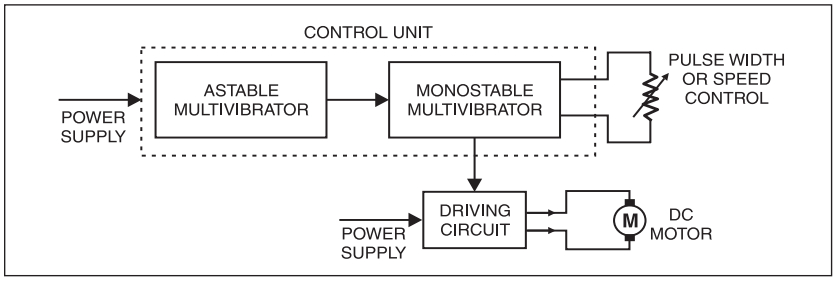
Resistor R7 and capacitor C6 have been deliberately added to delay the switching off of relay RL1. This extends the lamp's 'off' time (for a short duration), allowing the owner to move in while the light is on. The delay can be increased by increasing the value of capacitor C6. (***Note.*** A high-value capacitor will also increase the delay in turning the lights on, which is not desirable.)

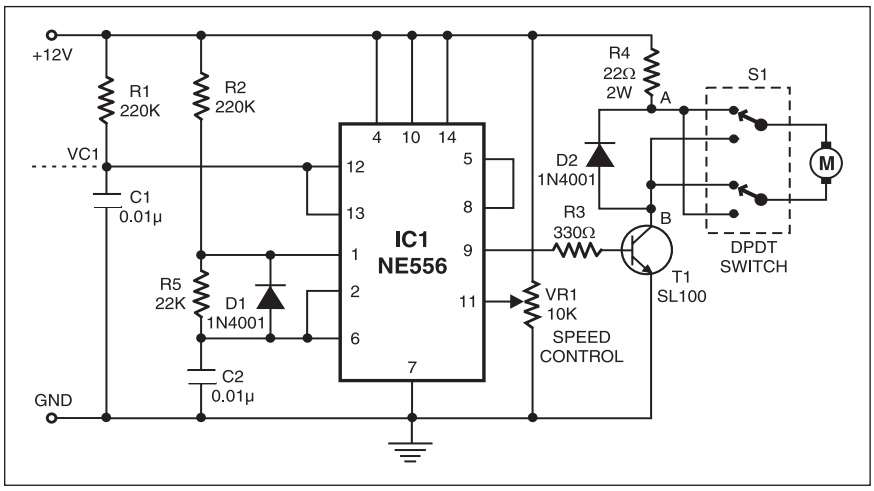
PWM-BASED SPEED CONTROL FOR DC MOTORS

ANAND TAMBOLI

There are several methods for controlling the speed of DC motors. One simple method is to add series resistance using a rheostat. As considerable power is consumed in the rheostat, this method is not economical. Another method is to use a series switch that can be closed/opened rapidly. This type of control is termed as chopper control. We've described here a PWM-based chopper circuit that smoothly controls the speed of general-purpose DC motors.

Fig. 1 shows the block diagram of a basic PWM-based chopper. The circuit shown in Fig. 2 is designed as per this diagram. A dual timer IC (NE556) is used to configure both the astable as well as the monostable multivibrator. Timing components for the astable are chosen to provide a frequency of 546 Hz, while the monostable components are selected to obtain a maximum pulsewidth of 2.42 ms. Diode D1 improves duty factor of the astable oscillator output, whereas D2 acts as a free-wheeling diode. Transistor SL100 drives the motor, while the 22-ohm, 2W resistor (R4) serves as a current limiter, avoiding overheating of the transistor. The DPDT switch enables direction reversal of the motor, as





desired.

The speed can be varied by adjusting VR1, which changes the threshold value to which capacitor C1 in the monostable circuit is charged. This, in turn, determines its output pulsewidth and hence the average voltage applied to the motor. Waveforms shown in Fig. 3 depict the average voltage for controlling various speeds.

For effective speed control, 'on' period (T_{ON}) of the astable should be equal to the maximum pulsewidth (T_{ON}) of the monostable.

For higher voltage and power requirements, SL100 can be replaced by an appropriate MOSFET or IGBT with relevant changes in the drive circuitry.

LED SAND-GLASS TIMER

ASHOK K. DOCTOR

This circuit (Fig. 1) simulates the old sand-glass timer. A total of 32 LEDs create the effect of sand grains passing from the upper half of sand-glass to its lower half.

When the power is switched 'on', shift registers IC3 and IC4 are reset by the power-on-reset circuit formed by resistor R34 and capacitor C7. After a few seconds, the sand-glass action starts. IC CD4060B (IC2) is a 14-stage ripple binary counter with built-in oscillator. It generates clock pulses, which are fed to both the shift registers (IC3 and IC4). The clock frequency can be adjusted by preset VR1, while rotary switch S2 helps in selecting time periods.

Time period options of 5, 10, and 20 minutes have been provided. Additional options are available on utilising outputs Q3 through Q9 of IC2.

Switch S1 is an SPDT switch, which selects odd- or even-numbered set of LEDs. In other words, it selects which side of the sand-glass is up.

Assuming S1 to be in position (a), every clock pulse causes shifting of a high logic level to IC3 via its inputs A and B, as long as pin 13 (Q7) of IC4 remains low. The MSB (Q7) of IC3 is shifted to inputs A and B of IC4. Controlled by the outputs of shift registers, transistors T1 through T16 switch off the odd-numbered LEDs and switch on the even-numbered LEDs in sequence, when pin 13 of IC4 goes high. Counter IC2 is reset via gates N1 and N2 of IC1 (CD4093, which is a quad 2-input Schmidt NAND gate), while

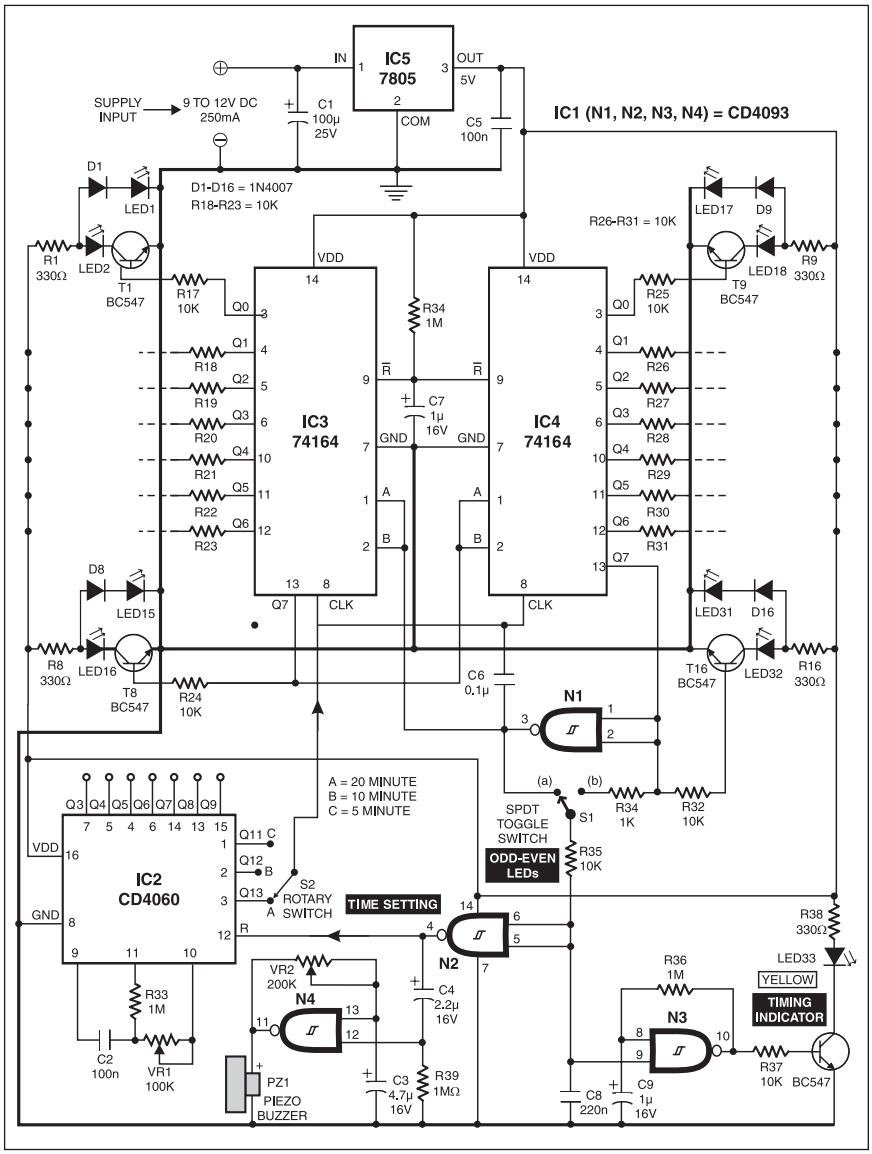


Fig. 1: LED sand-glass timer

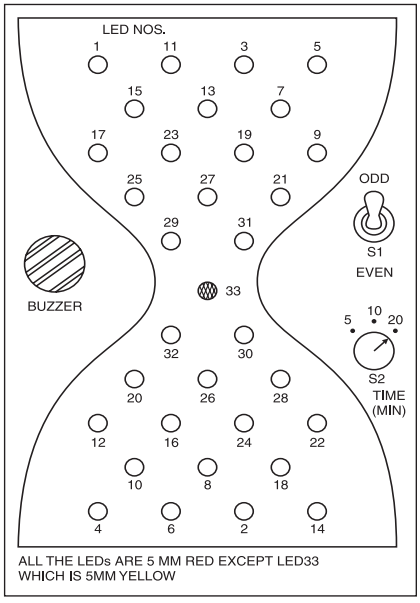


Fig. 2: Front panel of sand-glass timer

oscillator N4 sounds piezobuzzer PZ1. When one timing cycle is completed, the buzzer stops after giving two beeps and the timing indicator yellow LED also stops blinking. The tone of buzzer can be set by VR2 and C3.

Now reverse the sand-glass and flip SPDT switch S1 to (b). Logic low levels are shifted to IC3, with pin 13 of IC4 being at logic high. Now, even-numbered LEDs go off one by one, and odd-numbered LEDs light one by one until pin 13 of IC4 goes low and the buzzer beeps. LED33 indicates that the sand-glass is ready for timing.

Take a 15x5cm piece of dark-coloured acrylic or plastic sheet to mount LEDs. Mount switch S1 at the back of the cabinet. The circuit works off a 9V, 250mA supply provided by a battery eliminator. Take care while wiring LEDs. The circuit is symmetrical and it can be assembled on a veroboard or a general-purpose PCB.

THREE COLOUR DISPLAY USING BICOLOUR LEDs

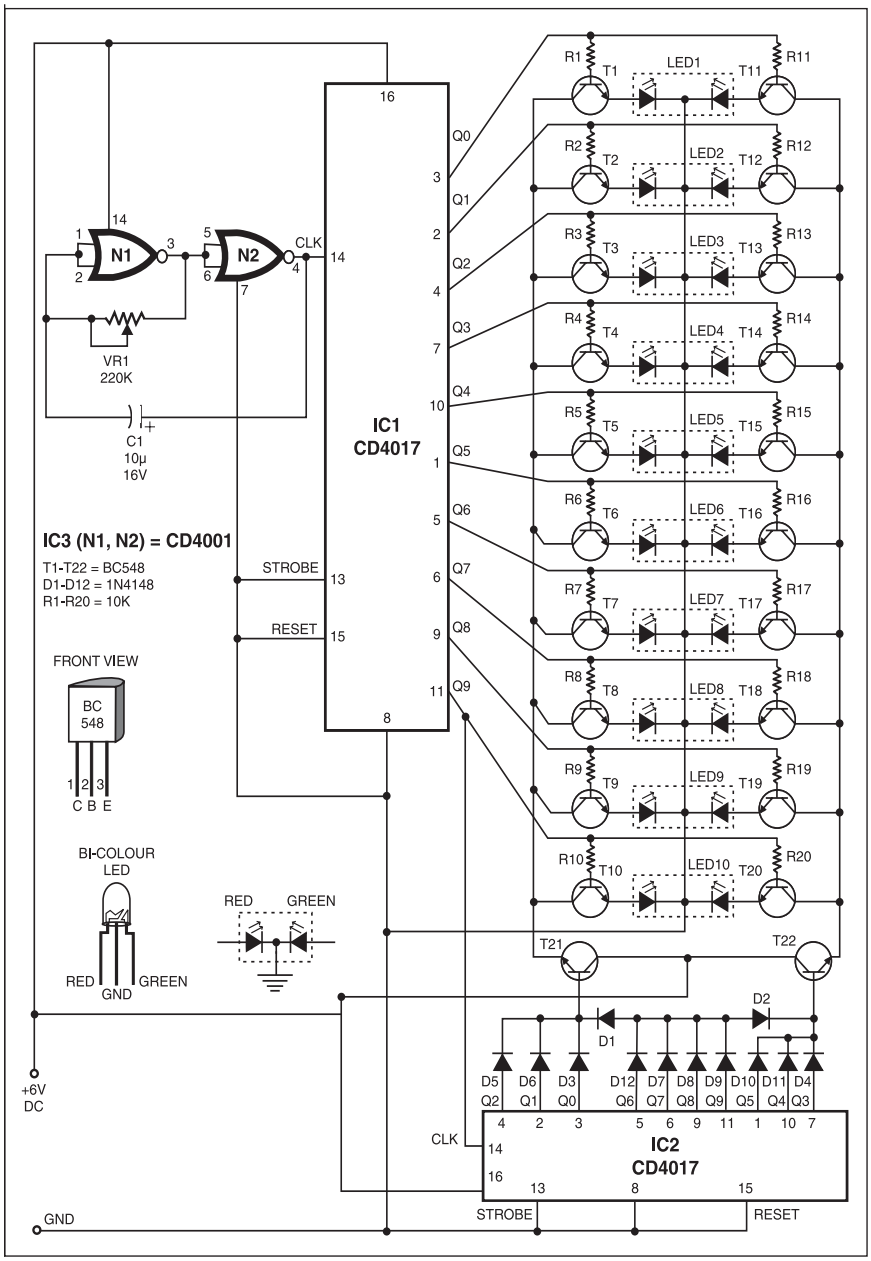
PRIYANK MUDGAL

The circuit presented here uses bicolour LEDs to generate a display in three colours, namely, red, green, and yellowish green.

Transistors T1 through T20 form a grid to which common-cathode bicolour LEDs (LED1 through LED10) are connected. Transistors T1 through T10 have their collector terminals connected to the emitter of transistor T21. Similarly, transistors T11 through T20 have their collector terminals connected to the emitter of transistor T22. The bases of each pair of transistors (i.e. T1 and T11, T2 and T12,..., T10 and T20) are tied to outputs Q0, Q1,..., Q9, respectively, of IC1 (CD4017) through 10-kilo-ohm resistors as shown in the figure. Positive supply to collectors of transistors T1 through T10 is controlled by transistor T21. Similarly, positive supply to collectors of transistors T11 through T20 is controlled by transistor T22.

IC1 and IC2 are decade counters. Clock pulse to IC1 is provided by the oscillator circuit comprising NOR gates N1 and N2. The outputs of IC1 advance sequentially with each clock. (Any other source of squarewave pulses also serves the purpose.) IC2 is used to select the mode of display. Clock input pin 14 of IC2 is connected to Q9 output of IC1. Thus IC2 receives one pulse after every ten pulses received by IC1.

When the circuit is switched on, Q0 output of IC2 is active high. Thus transistor T21 gets forward biased via diode D3 and it conducts to extend positive supply to transistors T1 through T10. Transistors T1 through T10 are forward biased sequentially by Q0 through Q9 outputs of IC1, i.e. at a time only one of these ten transistors is forward biased



(on). Thus only red LED parts of bicolour LEDs light up sequentially. (Transistor T22 is not conducting at this moment.)

When red LED part of LED10 glows, IC2 receives a clock pulse and its Q1 output goes high. Transistor T21 still conducts, as it is forward biased through diode D6, and next again via diode D5. Thus red LEDs complete two more glowing sequences.

After completion of the third glowing sequence of red LEDs, when Q3 output of IC2 goes high, transistor T21 stops conducting and T22 starts conducting with the next three sequences of green LEDs of bicolour LEDs (LED1 through LED10) glowing sequentially.

After completion of three sequences of green LEDs, output Q6 of IC2 goes high. Now both transistors T21 and T22 conduct due to diodes D1 and D2. Thus both red and green LEDs in bicolour LEDs (LED1 through LED10) glow sequentially. The effect of red and green LEDs glowing together is a distinct yellowish orange colour. This sequence repeats four times.

Thereafter, the whole sequence repeats, starting with red LEDs. Thus the bicolour-LED display shows three colours—red, green, and yellowish green—one after the other.

The speed of display can be controlled by preset VR1. One can omit automatic selection of different colours by omitting IC2 and replacing connections to pins 3, 5, and 7 of IC2 with SPDT switches. (Thus diodes D3-D12 are also omitted.)

VERSATILE EMERGENCY LIGHT USING FLUORESCENT TUBES

VIVEK KALKUR M.

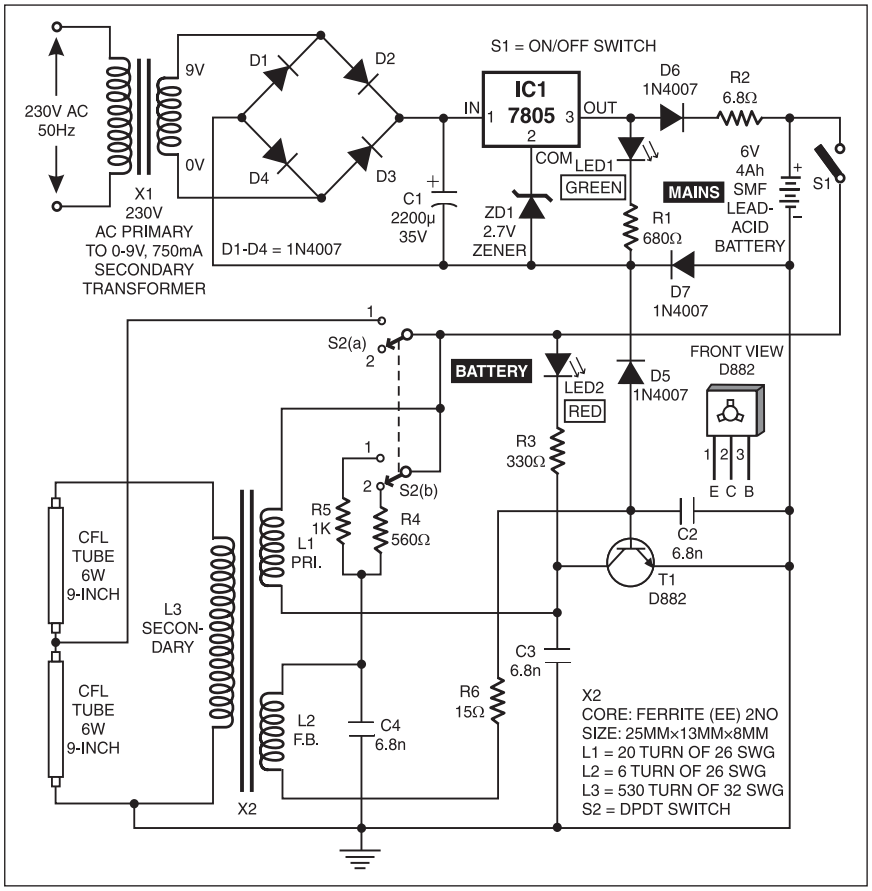
Emergency lights using incandescent bulbs are inherently inefficient compared to those using fluorescent tubes. Here's a versatile emergency light using fluorescent tubes. You can operate it using readymade SUNCA or similar other inverter transformers, which are readily available in the market. With this circuit you can drive two 6W, 22.8cm (9-inch) fluorescent tubes, with the option to use a single tube or a pair of tubes with the help of DPDT switch S2.

Step-down transformer X1, diodes D1-D4, capacitor C1, and 5V regulator IC1 (7805) form a regulated power supply. A 2.7V zener (ZD1) in common terminal of the regulator props up the output voltage to 7.7 volts. The regulated voltage is applied to the battery through diodes D6 and D7, which cause a drop of about 1.4V across them. Thus the effective charging voltage is about 6.3V, which prevents overcharging of the battery as the terminal voltage of the battery cannot exceed 6.3V.

When AC mains supply is present, the battery starts charging and green LED1 glows to indicate the same. Diode D5 reverse biases transistor T1 forming part of the inverter oscillator and thus the tubes don't glow.

When mains supply fails, transistor T1 starts oscillating and supplies power to inverter transformer X2 and the tubes glow. An on/off switch (S1) is used to switch off the light when it is not required.

D882 (actually, 2SD882) is an npn power transistor in TO-126 package. It is mounted on a suitable heat-sink to prevent it from thermal runaway. For good illumination, use Toshiba's FL6D fluorescent tubes.



ELECTRONIC SECURITY SYSTEM

K. BHARATHAN

This reliable and easy-to-operate elec-tronic security system can be used in banks, factories, commercial establishments, houses, etc.

The system comprises a monitoring system and several sensing zones. Each sensing zone is provided with a closed-loop switch known as sense switch. Sense switches are fixed on the doors of premises under security and connected to the monitoring system. As long as the doors are closed, sense switches are also closed. The monitoring system can be installed at a convenient central place for easy operation.

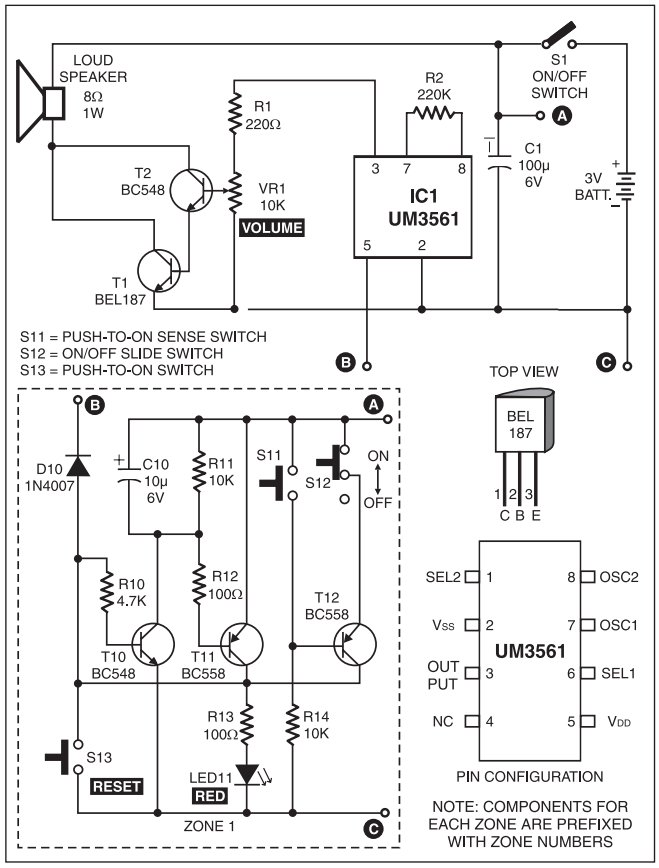


Fig. 1: Monitoring circuit along with the alarm circuit

Fig. 1 shows the monitoring circuit only for zone 1 along with the common alarm circuit. For other zones, the monitoring circuit is identical, with only the prefixes of components changing as per zone number. Encircled points A, B, and C of each zone monitoring circuit need to be joined to the corresponding points of the alarm circuit (upper half of Fig. 1).

When zone 1 sensing switch S11, zone on/off slide switch S12, and system on/off switch S1 are all on, pnp transistor T12 reverse biases to go in cut-off condition, with its collector at around 0 volt. When the door fitted with sensor switch S11 is opened, transistor T12 gets forward biased and it conducts. Its collector voltage goes high, which forward biases transistor T10 via resistor R10 to turn it on. (Capacitor C10 serves as a filter capacitor.) As a result, the collector voltage of transistor T10 falls to forward bias transistor T11, which conducts and its collector voltage is sustained at a high level. Under this latched condition, sensor switch S11 and the state of transistor T12 have no effect. In this state, red LED11 of the zone remains lit.

Simultaneously, the high-level voltage from the collector of transistor T11 via diode D10 is applied to V_{DD} pin 5 of siren sound generator IC1 (UM3561) whose pin 2 is grounded. Resistor R3 connected across pins 7 and 8 of IC1 determines the frequency of the in-built oscillator. As a result, IC1 starts generating the audio signal output at pin 3. The output voltage from IC1 is further amplified by Darlington pair of transistors T1 and T2. The amplified output of the Darlington pair drives the loudspeaker whose output volume can be controlled by potentiometer VR1. Capacitor C1 serves as a filter capacitor.

You can alter the alarm sound as desired by changing the connections of IC1 as shown in the table.

The circuit continues to sound the alarm until zone door is closed (to close switch S11) and the reset switch is pressed momentarily (which causes transistor T10 to cut off, returning the circuit to its initial state).

The system operates off a 3V DC battery or recharging battery with charging circuit or battery eliminator. If desired, more operating zones can be added.

Initially keep the monitoring system switch S1 off. Keep all the zone doors fixed with sensing switches S11, S21, S31, S41, etc closed. This keeps the sensing switches for respective zones in closed position. Also keep zone slide

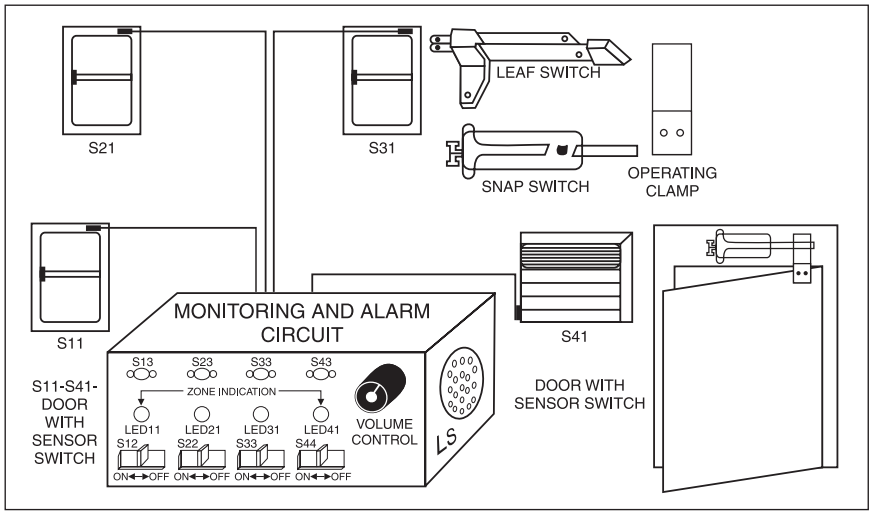


Fig. 2: Physical layout of sensors and monitoring/alarm system

Alarm sound	Circuit connections	
	IC pin 1 connected to	IC pin 6 connected to
Police siren	NC	NC
Ambulance siren	NC	V_{DD}
Fire engine Sound	NC	V_{SS}
Machinegun sound	V_{SS}	NC

Note. *NC indicates no connection*

switches S12, S22, S32, S42, etc in 'on' position. This puts the system in operation, guarding all the zone doors.

Now, if the door of a particular zone is opened, the monitoring system sounds an audible alarm and the LED corresponding to the zone glows to indicate that the door of the zone is open. The alarm and the LED indication will continue even after that particular door with the sensing switch is immediately closed, or even if that switch is removed/damaged or connecting wire is cut open.

Any particular zone in the monitoring system can be put to operation or out of operation by switching on or switching off the corresponding slide switch in the monitoring system.

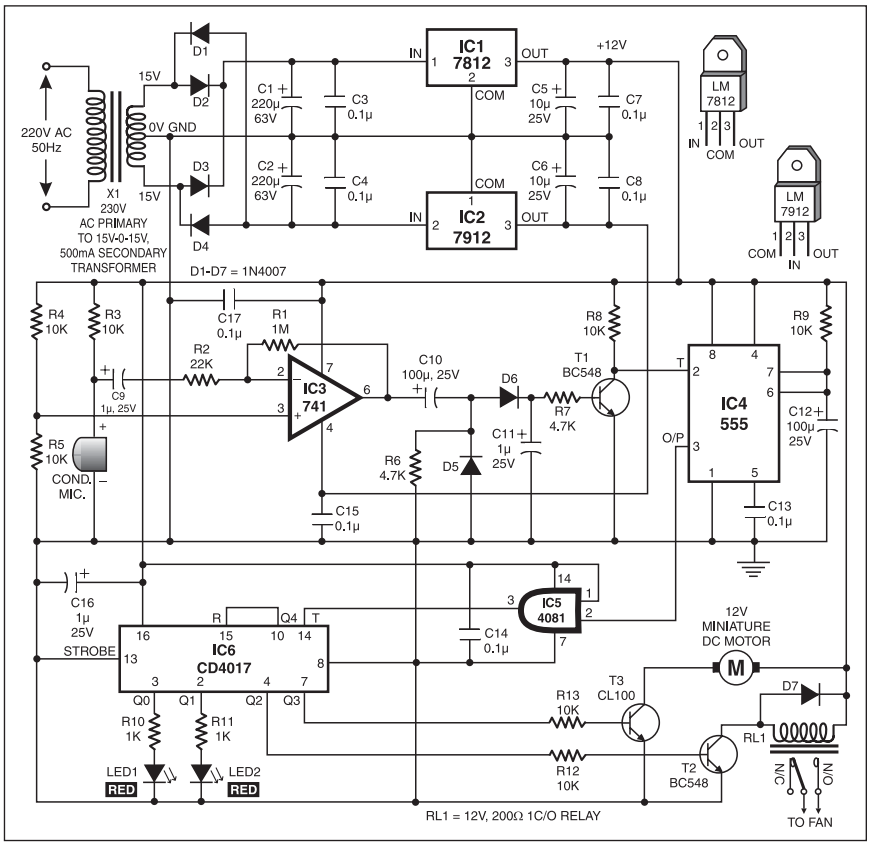
CLAP-BASED SWITCHING FOR DEVICES

MANOJ KUMAR SAHA

It is quite difficult to find the switch board in a dark room to turn on the light. Here's a clap switch that allows you to switch on lights, fans, and motors sequentially by just clapping in the vicinity of the microphone used in the circuit.

The mains supply is stepped down to 15-0-15V AC by step-down transformer X1. The output of the transformer is rectified, filtered, and regulated by diodes D1 through D4, capacitors C1 through C4, and IC1 (regulator IC 7812) and IC2 (regulator IC 7912), respectively. Additional filtering is performed by capacitors C5 through C8 to get +12V, 0V (Gnd) and –12V DC required for the operation of the circuit.

The clap sound impulses are converted into electrical signals by a condenser microphone that forms a Wheatstone bridge together with resistors R4, R5, and R3. The microphone is suitably biased through resistor R3. The output of the microphone is coupled to op-amp IC 741 (IC3) having a voltage gain of 45. The output of IC3, after passing through capacitor C10, is free from any DC component of signal. Capacitors C15 and C17 are used for spike and surge suppression.



Diodes D5 and D6 and capacitor C11 form the detector circuit. Resistor R6 is used here for quick discharge of capacitor C10. The detected clap signal is used to switch on transistor T1. On conduction of transistor T1, its collector voltage falls to trigger timer IC4 connected as a monostable. The combination of resistor R9 and capacitor C12 determine the pulsewidth of the monostable (about one second, with the component values shown).

AND gate IC5 (4081) is used as a buffer between the output of IC4 and clock input to decade counter IC6 (CD4017). Thus each clap causes outputs of IC6 to advance in sequential manner and switch on the corresponding devices.

If you want a lamp to be switched on when output Q1 goes high (after first clap), then in place of R11 and LED2 use a relay driver circuit at Q1 output similar to that used for Q2 output (for fan).

As stated earlier, only one output of CD4017 can be high at any given time. Thus first clap causes LED1 to go off and LED2 to glow. The second clap causes only the fan to switch on via relay RL1. The third clap causes the miniature 12V motor to run. On fourth clap, Q4 output goes high momentarily to reset IC6 since Q4 output is connected to its reset pin 15. In reset state, LED1 connected to Q0 output lights up.

LEAD-ACID BATTERY CHARGER WITH VOLTAGE ANALYSER

D. MOHAN KUMAR

Nowadays maintenance-free lead-acid batteries are common in vehicles, inverters, and UPS systems. If the battery is left in a poor state of charge, its useful life is shortened. It also reduces the capacity and rechargeability of the battery. For older types of batteries, a hygrometer can be used to check the specific gravity of the acid, which, in turn, indicates the charge condition of the battery. However, you cannot use a hygrometer for sealed-type maintenance-free batteries. The only way to know their charge level is by checking their terminal voltage.

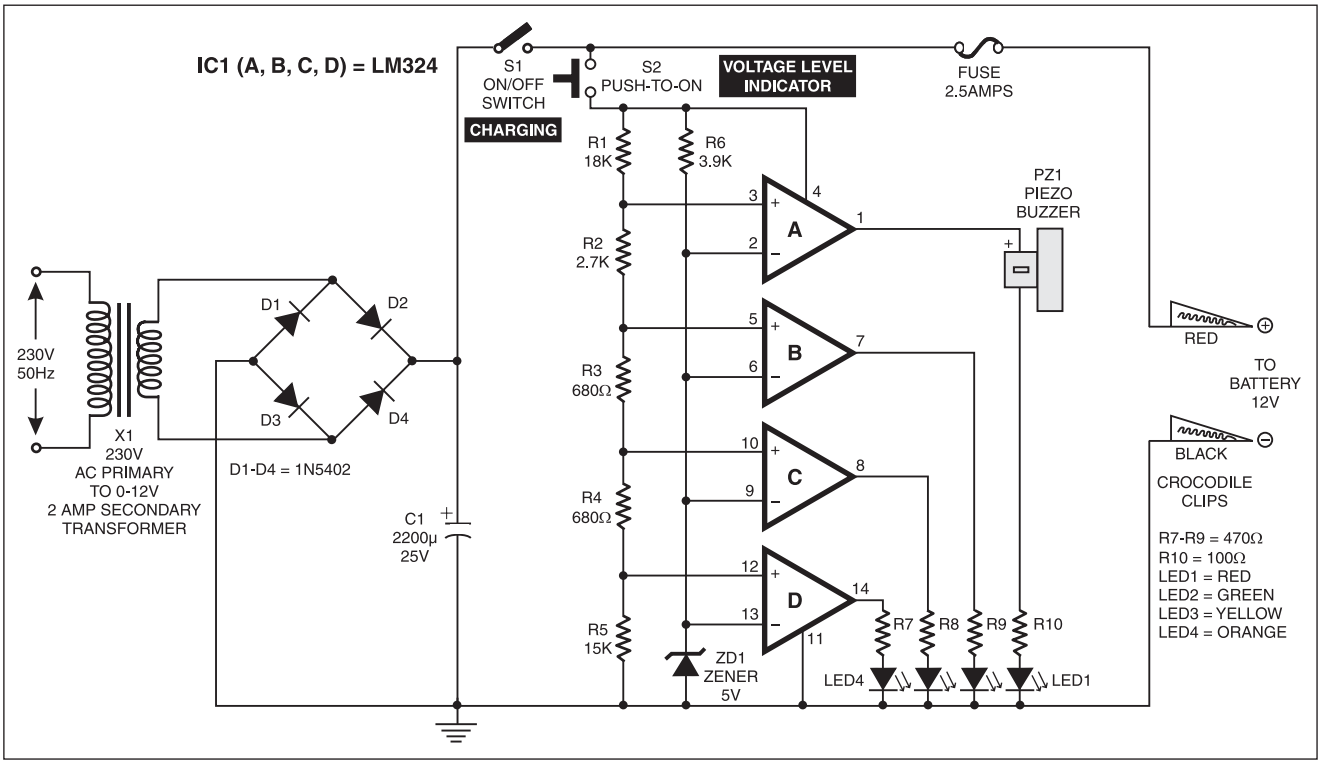
The circuit presented here can replenish the charge in a battery within 6-8 hours. It also has a voltage analysing circuit for quick checking of voltage before start of charging, since overcharging may damage the battery. The voltage analyser gives an audio-visual indication of the battery voltage level and also warns about the critical voltage level at which the battery requires immediate charging.

The charger circuit consists of a standard step-down 12V AC (2-amp) transformer and a bridge rectifier comprising diodes D1 through D4. Capacitor C1 smoothes the AC ripples to provide a clean DC for charging the battery.

The battery voltage analyser circuit is built around the popular quad op-amp LM324 that has four separate op-amps (A through D) with differential inputs. Op-amps have been used here as comparators. Switch S2 is a push-switch,which is pressed momentarily to check the battery voltage level before charging the battery.

The non-inverting terminals of op-amps A through D are connected to the positive supply rail via a potential divider chain comprising resistors R1 through R5. Thus the voltage applied to any non-inverting input is the ratio of the resistance between that non-inverting terminal and ground to the total resistance (R1+R2+R3+R4+R5). The resistor chain provides a positive voltage of above 5V to the non-inverting inputs of all op-amps when battery voltage is 12.5V or more. A reference voltage of 5V

Battery voltage	Status of LEDs				Comments
	Red	Green	Yellow	Orange	
< 9.8V	Off	Off	Off	Off	Buzzer off
> 9.8V	On	Off	Off	Off	Danger level
11.5V	On	On	Off	Off	Low level
12.0V	On	On	On	Off	Normal level
12.5V	On	On	On	On	High level



is applied to the inverting inputs of op-amps via 5V zener diode ZD1.

When the circuit is connected to the battery and pushswitch S2 is pressed (with S1 open), the battery voltage is sampled by the analyser circuit. If the supply voltage sample applied to the non-inverting input of an op-amp exceeds the reference voltage applied to the inverting inputs, the output of the op-amp goes high and the LED connected at its output lights up.

The different levels of battery voltages are indicated by LED1 through LED4. All the LEDs remain lit when the battery is fully charged (above 12.5V). The buzzer connected to the output of IC1 also sounds (when S2 is pressed with S1 kept open) as long as the voltage of battery is above 9.8V. If the voltage level goes below 9.8V, the buzzer goes off, which indicates that it's time to replace the battery. The status of LEDs for different battery voltages is shown in the table.

The circuit can be assembled on a general-purpose PCB or a veroboard. Use 4mm wire and crocodile clips to connect the charger to the battery. A 2.5-amp fuse connected to the output of the charger protects the analyser circuit against accidental polarity reversal.

KEYPAD CONTROL FOR MULTIPLE APPLIANCES

S. RAMASAMY
R.G.THIAGARAJ KUMAR

This circuit employs DTMF technique to switch on/off up to ten appliances. It can be modified to operate up to 100 appliances using the same keypad.

The controller uses telephone-type keypad with 12 press-to-on switches. These switches are arranged in four rows (R1 through R4) and three columns (C1 through C3) using seven lines that are terminated at corresponding inputs of DTMF encoder UM91214B (IC1). IC1 generates 12 distinct dual-tone signals corresponding to the switch pressed. This signal is routed to the receiver using a wired link. (It can also be used for remote control using IR or FM.)

The circuit diagram of the wire link unit is shown in Fig. 1. No modification is required in this unit whether you want to control 10 or 100 appliances. Only the subsequent stages differ for the two models. The wired link unit can be operated off a 9V PP battery or using a suitable AC mains adaptor.

The receiver unit, as shown in Fig. 2, decodes the received DTMF signal with the help of DTMF decoder IC KT3170/MT8870 (IC1) and provides the binary output according to the switch pressed in the handheld unit. It also provides StD signal that indicates the receipt of a valid DTMF code. When the system is initially reset, all the Q outputs of CD4013 dual D flip-flops (IC3 through IC7) are cleared and all the appliances are turned off.

Whenever a particular key is pressed in the control unit, the signal transmitted via the wire link is received by the DTMF decoder and it generates the corresponding binary code in its output lines A, B, C, and D. The delayed steering

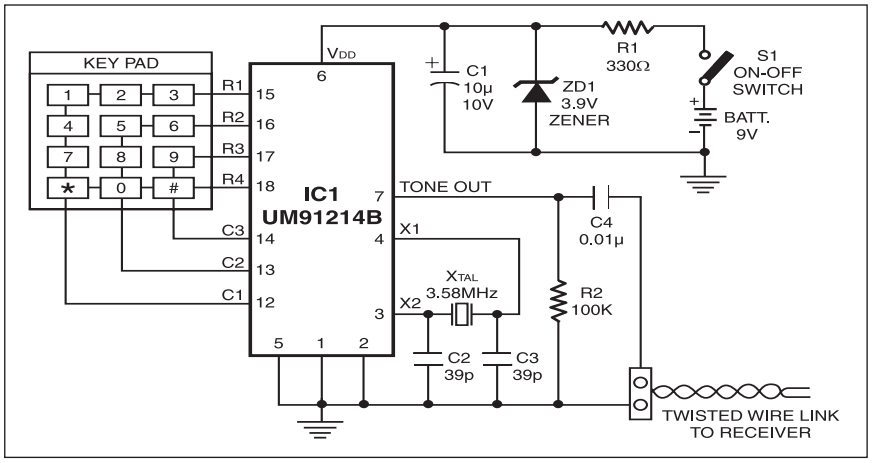


Fig. 1: DTMF transmitter with keypad

trigger output (StD) also goes high, which becomes low when the key is released.

The binary output of the DTMF decoder is connected as the input to the 4-line-to-16-line decoder CD4514 (IC2). One of the 16 output latches [IC3(A) through IC7(B)], corresponding to the input data, goes high as latch-enable (LEN) is kept permanently high (active), while the control input EN is connected to the StD signal, which goes high as long as a switch on the keypad (Fig. 1) remains pressed and all the output lines of IC2 remain low. However, when the user releases the keypad switch in the control unit, the transmission of the DTMF signal stops and the StD signal output goes low to output the already latched data. The output line of IC2 corresponding to the released key on the keypad goes high. Thus, a low-to-high transition occurs in one of the decoder output lines corresponding to the switch pressed and released in the hand unit (remote transmitter with keypad).

The positive-going pulse triggers the corresponding D flip-flop, which is wired in toggle mode to control the desired appliance. Hence, a particular appliance (say,

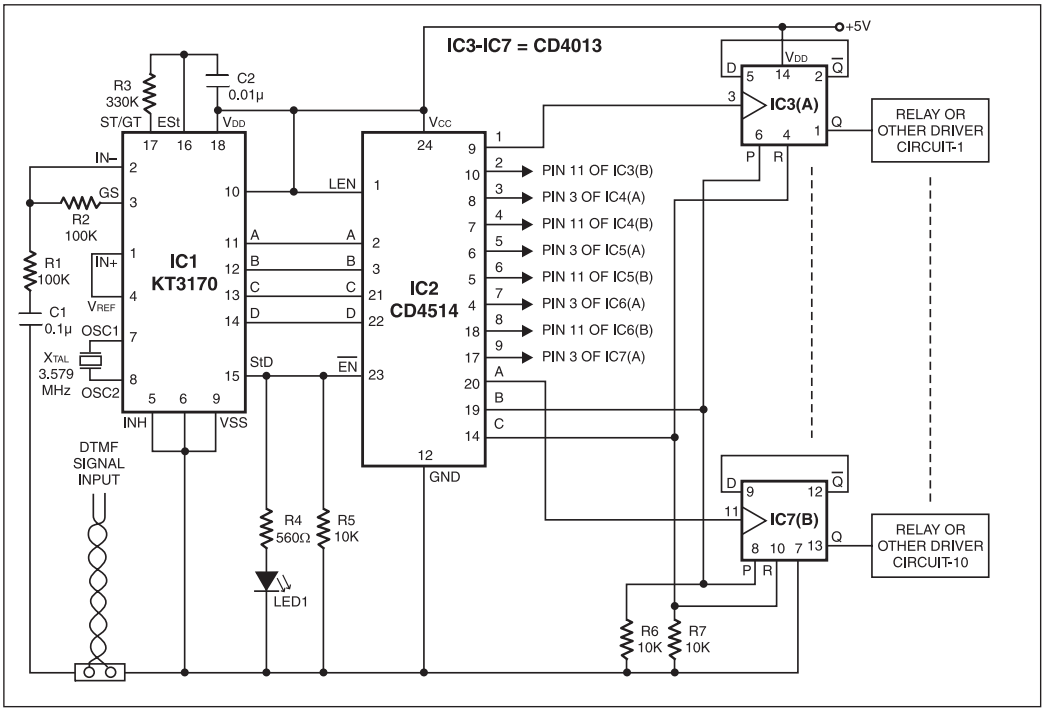


Fig. 2: Receiver including appliance switching circuit

No. 5) can be turned on by momentarily pressing the corresponding key (5). Subsequently, it can be turned off by momentarily pressing the same key again.

Keys marked '*' and '#' in the keypad act as master switches for turning on and off, respectively, all the appliances. When keys '*' and '#' in the keypad are pressed (one at a time), they generate signals that are decoded by IC1 as B and C (hex), respectively. The corresponding output lines (marked B and C) of IC2 are connected to Set and Reset terminals, respectively, of all the toggle flip-flops, to turn on/off all the appliances simultaneously.

The flip-flop outputs can be used to control relays through ULN2001 relay drivers or similar ICs. If relays are not suitable, these outputs can be used to drive optocouplers (such as MOC3040), which, in turn, control triacs to turn on/off the power appliances. This scheme provides a total isolation between AC mains and the controller and hence it is quite safe to operate.

Readers Comments:

❑ In the 'Keypad Control for Multiple Appliances' circuit, I have observed:

1. The output of DTMF decoder IC1 KT3170 (shown in Fig. 2) for decimal 0 is 10102 and not 00002. Hence, there must be a logic circuit between IC1 and IC2 (CD4514) that converts 10102 into 00002 without interfering with numbers 1 through 9.

2. Why this cumbersome circuit of producing a tone generator and a decoder is employed? There are ICs which convert decimal into BCD directly. The IC 74C922 encodes hexadecimal input from keypad into BCD output.

P.G. Rajagopalan
Cochin

The author S. Ramasamy and R.G. Thiagaraj Kumar replies:

1. The circuit has been practically wired, tested, and is being used for more than a year now without any problem. It is true that the '0' key pressed in the keypad generates a DTMF signal corresponding to 1010 (A-hex). The signal is transmitted over two wires and is decoded by KT3170 again as 1010, which is fed to the 4-line-to-16-line decoder IC CD4514. The output 'A' (pin 20) of this IC activated by the 1010 signal from KT3170 is used to drive a flip-flop corresponding to the device '10'. Hence there is no need to have any logic circuit to convert 1010 into 0000.

2. The purpose of using a tone generator and decoders is to have a simple two-wire control for multiple (up to 10 here) appliances.

In our prototype, we have arranged the complete keypad and the necessary circuits in a toy cell phone unit available in the market, hence it is very handy. This unit was powered by a 9V PP3 battery and linked to the switcher unit by two wires only. The ICs used for hex to BCD conversion cannot be used to control multiple devices with two wires.

WIRELESS TV HEADPHONE CIRCUIT

PRADIPTA BANERJEE

This circuit allows you to watch your favourite TV programmes late at night without disturbing other family members. As against imported stereo wireless TV headphones available in the market for around Rs 1200, it costs just Rs 30, or even less, if the components are taken from a discarded transistor receiver, with no compromise on performance.

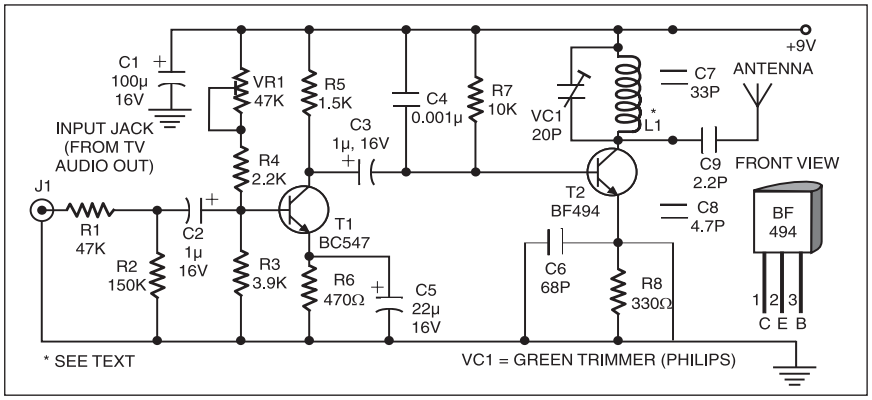
The unit is basically a simple FM transmitter housed in a plastic or metal enclosure. Transistor T1 acts as an audio preamplifier. Transistor T2 works as an FM oscillator and modulator in conju-nction with other passive components. Trimmer capacitor VC1 connected across inductor L1 can be varied to achieve the desired frequency. Inductor L1 comprises 4 to 6 turns of closely wound 25SWG enamelled copper wire on a 4mm dia. air core. A 20-30cm

long wire serves as an antenna.

Most modern TVs are nowadays equipped with audio-in/out and video-in/out RCA sockets. Using an RCA-to-RCA cord, connect the audio output of your TV to the transmitter's input. Adjust the gain of the audio preamplifier with the help of preset VR1 for clear reception in a portable FM receiver equipped with an earphone socket. Use a good-quality earphone.

This transmitter draws only a few milliamperes of current and doesn't require on/off switch. It can be fabricated on a small piece of stripboard. All connectors should be firm and as short as possible to prevent unwanted oscillations. The circuit operates off two AA-size penlight torch cells.

The circuit is meant for mono reception.



EFY note. 1. All TVs don't have headphone jacks.

2. For better result replace the transistor BF494 (T2) with 2N3866 and C9 (2.2pf) with L2. Because adding inductance to an anteena effectively increase its length whereas adding capacitance decrease its length.

L2 = 5 turn of 24SWG wire (enamelled copper) on 5mm dia air core.

Readers Comments:

Q1 I have following queries concerning 'Wireless TV Headphone Circuit'.

1. Can I connect a condenser microphone in the circuit?
2. As green trimmer is not available in our town, please suggest a fixed value of capacitor that can work between 88 and 108 MHz.
3. Can I take parallel connection from the speaker to the input of this circuit if the TV does not have headphone out?
4. What is the range of the circuit?
5. What is the current consumption of the circuit?

Ravi
Rajahmundry, AP

Q2 The author of the circuit 'Wireless TV Headphone' published in May has wrongly mentioned the range of that project. The range of this low-power transmitter is below 25 metres.

Also, the author says that on/off switch is not needed for the circuit due to 10mA current drain. Actually, 10mA current drain at 3V supply is not ignorable if the circuit runs on a battery. An on/off switch should be used if the circuit is not in use. Then only the battery will last a few weeks. Without on/off switch, the battery needs to be replaced every five days.

Pradeep G.
Bangalore

Q3. Changing the coil's specifications increases the circuit's range to 20 metres and makes the noise almost null. The coil should have 4 turns of 20 SWG wire on a 3mm dia. air core with a slight spacing to a length of 1.5 cm. Also use a 9V battery in place of 3V battery and a 10-15cm connecting wire as the antenna.

Yogesh Tarte
Aurangabad

The author Pradipta Banerjee replies:

A1. 1. You can't use a condenser mic as its gain is very low.

2. If green trimmer is not available, you need not use it. Simply tune in the FM receiver and you will get the required sound.

3. Connecting speaker wires to the input (J1) of this circuit will lead to severe distortion. However, you can try it by lowering the volume of the TV.

4. With a good-quality superhet FM receiver, the range is 10 to 12 km if a whip antenna is used.

5. The current consumption is very low—of the order of 10 mA, so an on/off switch is not necessary.

A2. The range of the transmitter is 10-12 metres and not 10-12 km as given in 'Letters' section. Regarding the use of on/off switch, I would like to tell Mr Pradeep that I am using the prototype for months with a new battery, and even with used batteries without any trouble. I admire Mr Pradeep for building and testing the prototype himself.

A3. I am very glad that the reader took interest in my circuit and modified it for efficient operation. I myself will make the necessary changes mentioned by him to make it more powerful.

AUTOMATIC WATER PUMP MOTOR CONTROLLER

DINESH KUMAR RAHEJA

Municipal corporations in many cities supply water during early morning hours. So, you have to wake up early, just to switch on your motor pump and wait till your water tank is filled up. Further, there is no control for overflow of the tank. Many times you come to know of your overflowing tank only when your neighbour informs you.

Here is a low-cost and simple automatic water pump controller circuit (Fig.

1) to avoid the aforesaid problem. You just have to set your quartz alarm clock (connected to this system) at the appropriate time of water supply. Keep the clock nearby your sleeping bed and switch on the circuit before going to sleep. In the morning, as the alarm rings, you can switch off the alarm and, if you like, go to sleep again. The controller system will automatically switch on the pump motor immediately at the predetermined time. When the overhead tank is full, the pump motor will get switched off automatically, preventing overflow of the tank. The controller system works with a water-level sensor assembly. The sensor assembly has to be fixed up properly inside your overhead water tank.

You can assemble a simple water-level sensor (Fig. 2) at home. Take an empty cylindrical plastic vial having outer diameter smaller than 1.3 cm (0.5 inch), which is commonly used for dispensing medicines. Make it opaque by pasting a piece of black paper or PVC tape on its inner side. Fix up the vial's lid firmly with a suitable material to make it airtight.

Now take a 15.2cm (6-inch) piece of opaque PVC conduit pipe having inner diameter of 1.3 cm (0.5 inch). Ensure that the vial can freely slide along the axis inside the conduit pipe; else use a PVC pipe having larger inner diameter. Drill two through-holes (of diameter 3.5 mm) along the diameter of the PVC pipe, at least 15 mm away from each end. Also drill two 5mm dia. through-holes along the diameter of the PVC pipe, 25-30 mm away from one end.

Insert the airtight plastic vial inside the PVC pipe and fix up two screws (M3x25mm) through 3.5mm dia. holes with suitable M3 nuts near the ends. These screws will restrict sliding of the vial within the PVC pipe.

Then fix up a 5mm red LED perpendicular to 5mm dia. hole, on the outer surface of PVC pipe, using suitable adhesive like M-seal. Also, opposite to this, fix up a light-dependent resistor (LDR) in similar way as shown in Fig. 1. Ensure that both the LDR and the LED are firmly placed outside the PVC pipe and they don't interrupt movement of the vial inside the PVC pipe. Also ensure that the maximum light produced by the LED falls on the surface of the LDR through 5mm dia. holes along the diameter of the PVC pipe. Then plug the open ends of PVC pipe with dark-coloured (preferably black) sponge pieces, to avoid ambient light entering inside the pipe.

Solder the leads of LDR and LED with connecting wires of required length, according to the location of your overhead water tank. Then fix up this water-level sensor assembly vertically at a suitable height along the inner wall of the overhead water tank, as shown in Fig. 3.

Remove 1.5V (AA size) battery from the quartz alarm clock. Carefully open the back lid of the clock using a small screwdriver. Solder a braided pick-up wire parallel to the connected points of your alarm buzzer (clock). Make a small hole on the back lid of the clock and pull out free ends of the pick-up wire through this hole. Then, carefully fix up the back lid in proper position, without disturbing any mechanical part of the clock. Solder the free ends of pick-up wire on the PCB at input terminals. This pick-up wire will provide trigger pulses from the alarm clock to water pump motor controller system. Then set the time and place the battery in alarm clock.

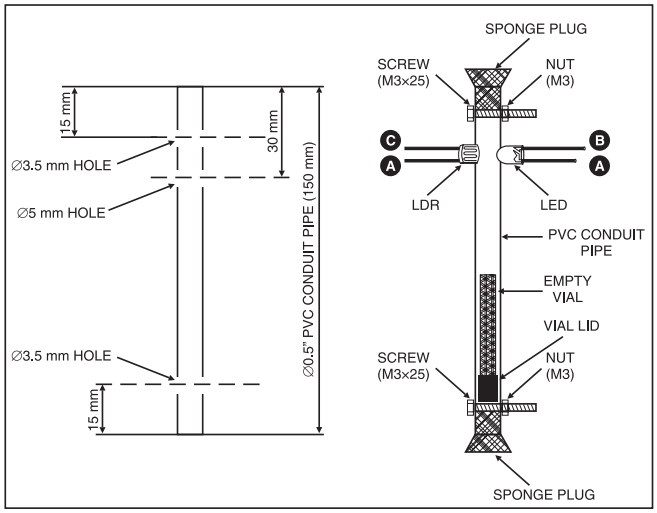


Fig. 2: Water-level sensor assembly

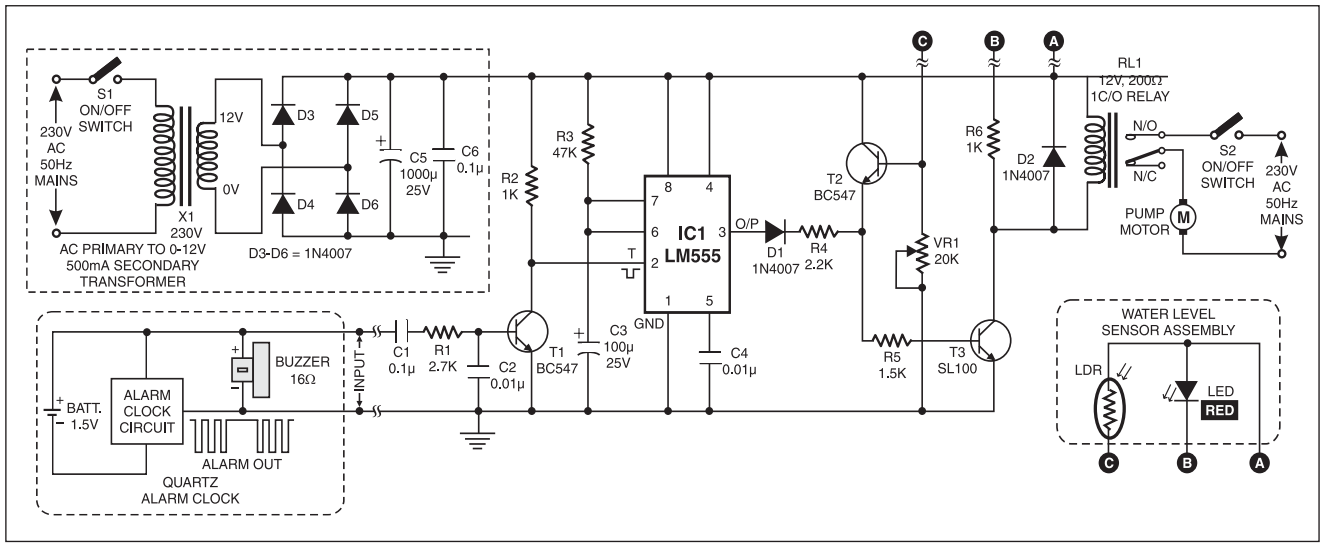


Fig. 1: Automatic water pump motor controller

As the alarm rings, trigger pulses appear at the input of the water pump motor controller circuit shown in Fig. 1. A low-pass filter is connected at the input to avoid false triggering of the circuit due to excessive EMI produced by the pump motor and the relay contacts. Timer LM555 (IC1) is configured as a monostable multivi-brator. The voltage at its output (at pin 3) goes high immediately after receiving trigger pulses (at pin 2), and it remains in this state for about five seconds, as set by the values of R3 (47k) and C3 (100μ, 25V). Diode D1 gets forward biased and sufficient base current flows through R4 and R5 to drive transistor T3 into saturation. A free wheeling diode (D2) is used to protect transistor T3 from the reverse current caused by self-induction of the relay coil (RL1).

Transistor T3 immediately switches on the relay. This, in turn, switches on the pump motor by the relay and at the same time the LED is turned on. The water is pumped by the motor and starts filling up the overhead water tank from the underground reservoir.

The light produced by LED continuously falls upon the LDR. This causes a decrease in the resistance value of the LDR and the base of transistor T2 (BC547) gets a high voltage (about 10V DC), which results in conduction of transistor T2. Since transistor T2 is biased as an emitter follower, the emitter voltage also becomes high. Due to this voltage feed, transistor T3 (SL100) remains in saturation.

Thus, once the circuit is triggered from the output of IC1, it will stay continuously on even when the input is withdrawn. As such, the pump motor will continuously remain on through the relay contact. (***Note.*** Reflection of light from any moving/still object near the LDR affects the proper functioning (latching) of the circuit. The value of VR1 (20k) has to be chosen for proper functioning of the circuit and according to the amount of light received in your circuit from the surroundings.)

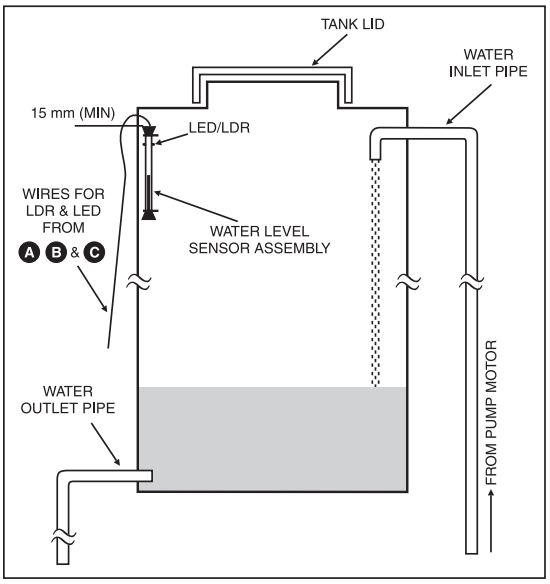


Fig. 3: Water-level sensor assembly fixed vertically at a suitable height along the inner wall of the overhead water tank

As the water level in the overhead tank rises, the empty vial floating along the water level inside the PVC pipe also goes up. When the water level reaches the desired level, the floating vial interrupts the path of light falling on the surface of LDR. Thus, the resistance of LDR increases sharply, resulting in voltage reduction at the base of transistor T2. Hence voltage at the emitter of T2 also goes low, which reduces the base current of transistor T3. This pulls T3 into cut-off region and both the LED and the relay are switched off. Ultimately, the AC supply via the relay to the pump motor is also cut off and the motor stops.

ELECTRIC SHOCK GUN

PRAVEEN KUMAR

This is a fantastic circuit for self-protection. In case a burglar intrudes your house, you can use this security circuit as a weapon for self-protection by giving a mild electric shock to the attacker.

This circuit comprises astable multivibrator, inverter, and voltage quadrupler sections. The astable multvibrator is designed for 1 kHz with a 9V DC supply. The inverter section consists of switching transistors and an inverter transformer. The primary of transformer is of 9V-0-9V and the secondary is of 100V, 100mA. For compatibility, a driver transformer that is used in radio is used as the inverter transformer. The secondary output current of 100 mA gives a good enough shock to human body.

The astable multivibrator consists of two BC107 transistors (T1 and T2), two 0.01μF capacitors (C1 and C2), two 4.7-kilo-ohm resistors (R1 and R4), and two

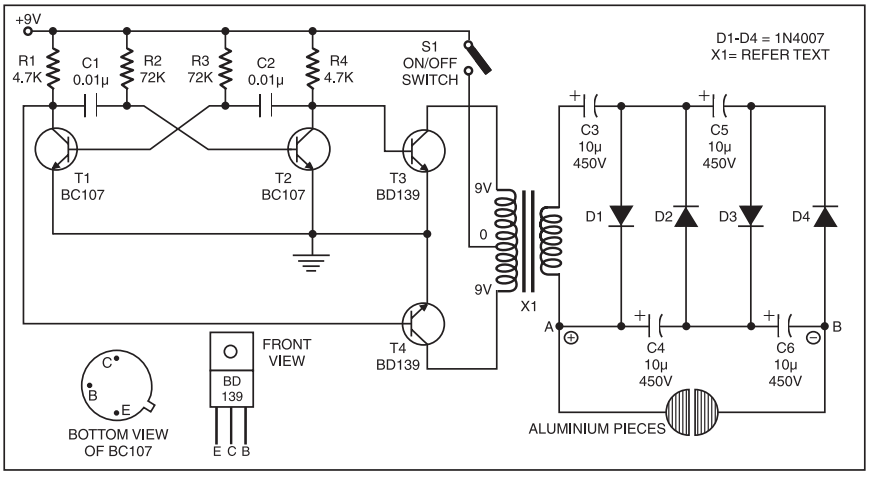
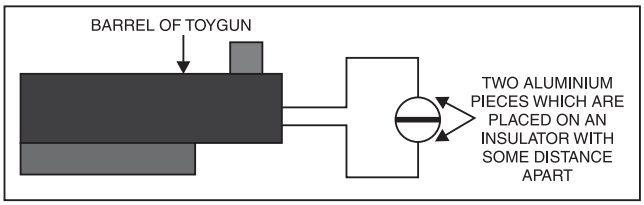


Fig. 1: Schematic diagram of toy shock gun



72-kilo-ohm resistors (R2 and R3). A squarewave output of 1 kHz (with a phase shift of 180 degrees) is obtained at the collector of transistor T1 or T2. This squarewave output is given to the base of switching transistors BD139 (T3 and T4).

The collectors of transistors T3 and T4 are connected to the primary of the transformer and their emitters are grounded. A DC supply of +9V is also applied to the center-tapping of transformer X1 through switch S1. This is an inverter action, so we get around 100V AC at the secondary of the transformer. This AC is given to a quadrupler circuit consisting of capacitors C3 through C6 and diodes D1 through D4.

The voltage quadrupler develops a DC voltage output equal to three or four times the input AC voltage. During the first positive half cycle, diode D1 conducts, charging C1 to Vm with polarity as shown in Fig. 1. During the first negative half cycle, diode D2 conducts, charging C2 to 2Vm. During the second positive half cycle, diodes D1 and D3 conduct, charging capacitors C1 and C3, while the voltage across capacitor C2 charges capacitor C3 to the same value 2Vm. During the second negative half cycle, diodes D2 and D4 conduct and capacitor C3 charges C4 to 2Vm. Thus, the voltage across C2 is 2Vm, across C1 and C3 is 3Vm, and across C2 and C4 is 4Vm. Therefore we get around 350V at the output of voltage quadrupler (across points A and B, as shown in Fig. 1).

Press the pushbutton switch (S1) of the circuit and touch the output connectors to any object. There will be a heavy electric discharge, which is enough for a good shock.

The circuit assembly and testing procedure is as follows:

1. Use a can type +9V battery for Vcc.

2. The transformer (X1) is designed to have 9V-0-9V primary and 100V, 100mA secondary, with primary winding having 80 turns (40+40, i.e. centre tapping at 40th turn) of 26 or 27 SWG and secondary winding having 450 turns of 35 or 36 SWG.

3. Mount the circuit in a toy gun with output connectors at the front end of the barrel. The output connectors may be connected to two aluminium pieces with an insulator placed between them to avoid short circuit. The arrangement is shown in Fig. 2.

4. Press switch S1 (here the trigger point of toy gun) and touch the front end of the barrel of toy gun to a person. The current in the voltage quadrupler will get discharged through the metal or aluminium pieces via the human body and the person will feel the electric shock.

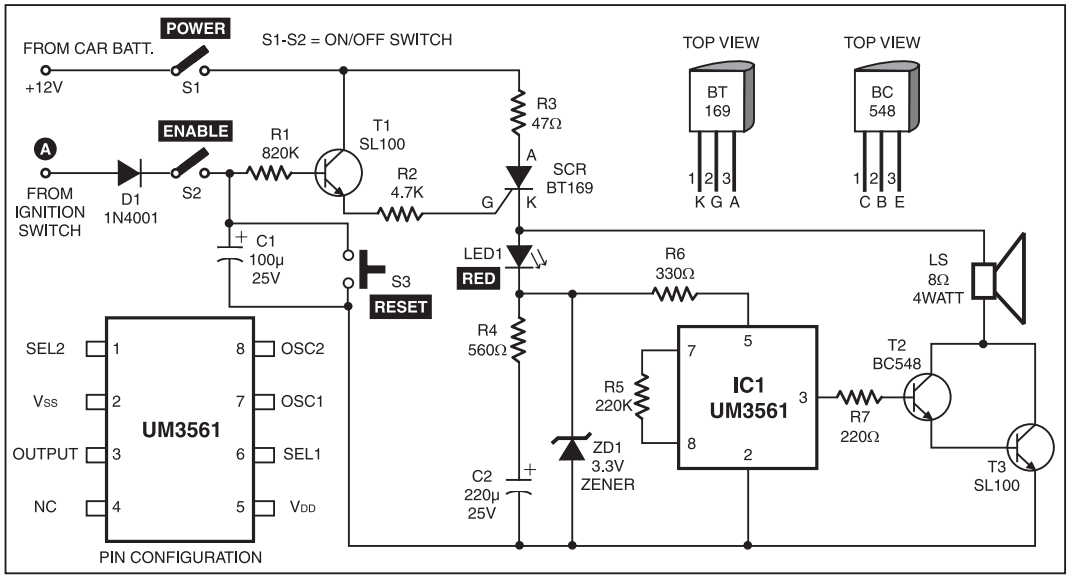
Caution. Check the circuit thoroughly before testing on anyone and use it judiciously, only when necessary.

ANTI-THEFT ALARM FOR VEHICLES

D. MOHAN KUMAR

This simple and inexpensive anti-theft circuit for vehicles sounds an alarm simulating a police siren whenever someone attempts theft of your vehicle. The alarm sounds continuously for a few seconds even when the intruder switches off the ignition key. The circuit uses only a few components and can be easily assembled and installed on a car with negative grounding.

The circuit consists of an SCR-based trigger circuit and audio alarm circuit. When the ignition key of the vehicle is switched off, base voltage of transistor T1 is low and it remains turned off. When the ignition key is switched on for starting the vehicle, a positive voltage is applied to the base of transistor T1 through diode D1, switch S2, and resistor R1, which slowly charges capacitor C1.



As a result, the base voltage of T1 rises. As soon as the biasing voltage crosses cut-in voltage, T1 turns on and SCR fires, giving 12V DC to the alarm circuit.

The alarm circuit is built around the siren-sound generator ROM UM3561 (IC1). It has a built-in oscillator, whose oscillation depends on resistor R5. Resistor R6 and zener diode ZD1 limit the voltage to IC1 to a safer level of 3.3V. The output from IC1 is fed to a transistor amplifier built around transistors T2 and T3.

The circuit gives sufficient time delay to switch on the alarm and to leave the vehicle. The alarm, once triggered, will sound until switch S1 is pressed to switch off the power supply.

Capacitor C2 is provided to sound the alarm even when the intruder switches off the ignition key. When the ignition key is switched off immediately, C2 discharges through R4 and keeps the alarm activated for half a minute. Reset switch S3 can be used to reset the alarm if needed.

The circuit can be assembled on a vero board. Use a small heat-sink for transistor T1. Connect point A to the ignition switch terminal that goes to the ignition coil. The hidden switch S1 is used for power on/off and switch S2 enables the circuit.

Note. Keep switches S1 and S2 on before leaving the vehicle. And don't forget to switch off S1 and S2 before starting the vehicle.

MULTI-SWITCH DOORBELL WITH INDICATORS

T.K. HAREENDRAN

Here's the circuit of a multi-switch input musical doorbell (shown in Fig.1). The circuit is built around the popular and less expensive quad D-latch CD4042B (IC1). When switch S6 is pushed to 'on' condition, the circuit gets +9V and the four data inputs (D1 through D4) of ICI are in low state because these are tied to ground via resistors R1 through R4. Polarity input (POL) pin 6 of IC1 is also pulled down by resistor R5. Clock input (pin 5) of the quad D-latch is wired in normally low mode and hence all the four outputs (Q0 through Q3) have the same states as their corresponding data inputs. As a result, LED1 through LED4 are in off condition.

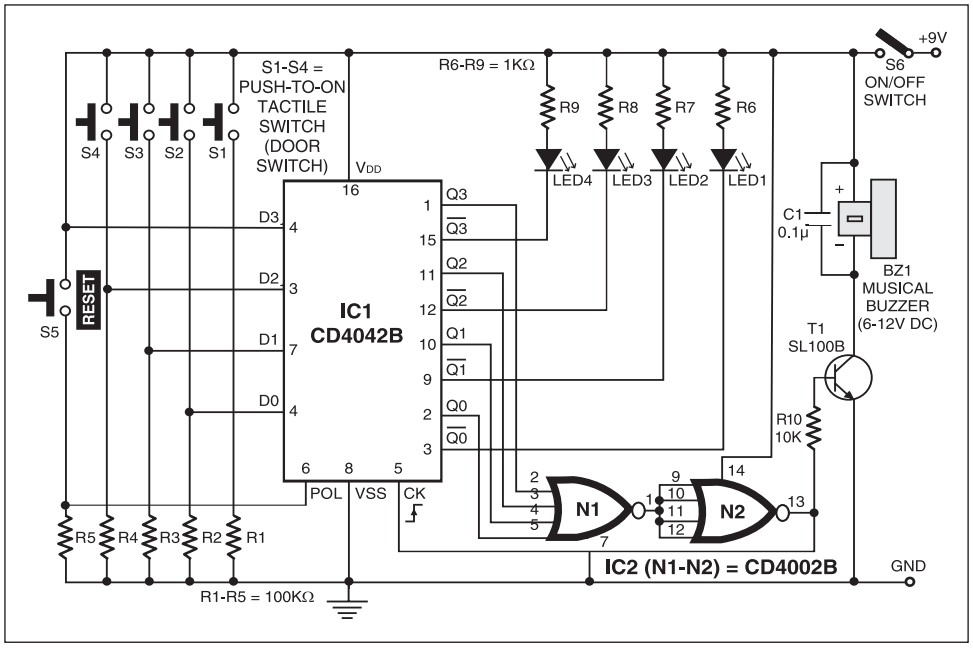


Fig. 1: Multi-switch doorbell with indicators

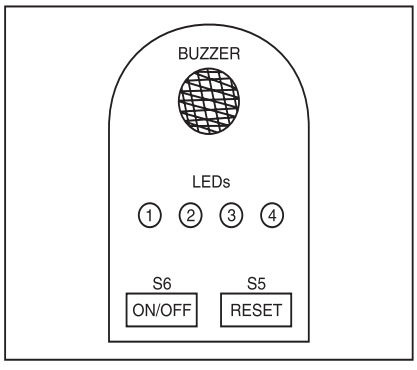


Fig. 2: Suggested panel layout of musical doorbell

There are four switches fitted at four different doors/gates outside the home and a monitoring panel (as shown in Fig. 2) in the common room of the home. If any switch is pressed by a visitor (for example, switch S1 at door 1), pins 2 and 4 of IC1 go high.

Simultaneously, pin 3 to IC1 (Q0 output) goes low and LED1 starts glowing to indicate that switch S1 is pressed by someone.

Next, output pin 13 of the dual 4-input NOR gate (IC2, here wired as a single 4-input OR gate) goes high to forward bias buzzer-driver transistor T1 via resistor R10.

The final result is a soft and pleasing musical bell, which lasts until reset switch S5 is pressed by the owner. For this latching arrangement , output pin 13 of IC2 from the NOR gate is fed back to the clock input of IC1.

SONG NUMBER DISPLAY

PRABHASH K.P.

Here's a circuit to display the song number in an audio system for quick reference to songs. It also serves the purpose of an extra visual indicator in modern audio systems.

When the power is switched on, the power-on-reset circuit comprising 3.3k resistor R20 and 1μF, 25V capacitor C6 resets the counters, showing '00' in the display. One can also reset the display to zero at any time by pressing reset switch S1.

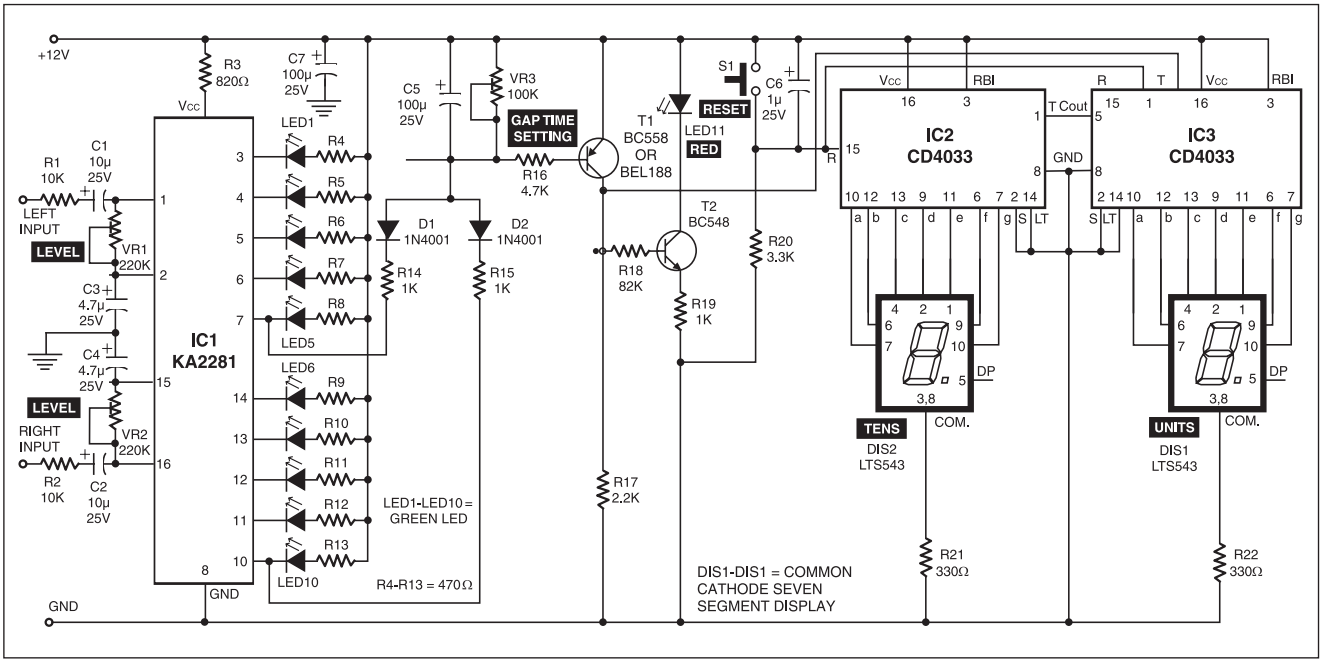
When the first song starts playing, the output pins of IC1 (KA2281) go low and capacitor C5 starts charging. This forward biases transistor T1 and hence the input to IC3 at pin 1 goes to high state. As a result, the output of the counter goes to the next state, showing 01 on the display. The counter remains in this state until the song is completed.

During the time gap before the next song starts playing, capacitor C5 discharges. After discharging of capacitor C5, the input to IC3 becomes low again. When the song starts, the process described above is repeated and the display shows 02. You can adjust VR3 to change the time gap setting. This must be set such that the circuit doesn't respond to short gaps, if any, within a song and responds only to long gaps between different songs.

Transistor T2 helps in gap-delay adjustment. The intensity of LED11 diminishes when a song is completed and the counter is ready to accept the next pulse.

Connect the input to the preamp output or equaliser output of the audio system. Adjust VR1 and VR2 to get the correct audio-level indication. If you are already using KA2281 for audio-level indication, just connect diodes D1 and D2 as shown in this circuit.

Note that the counter counts the songs by detecting the gaps. Therefore any long gap within a song may cause false triggering and the display will also be incremented. However, as this is very unlikely to happen, the circuit shows the correct song number almost all the time.



LOW-RANGE AM RADIO TRANSMITTER

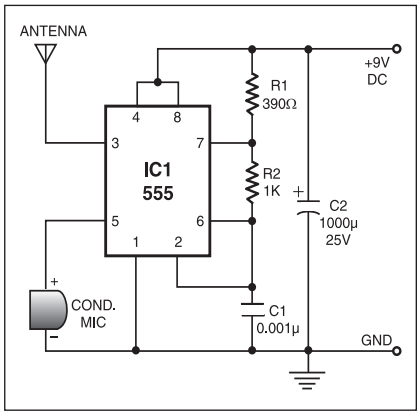
PARAG PURUSHOTTAM INGLE

Here is a simple radio transmitter for transmission up to 25 metres. It is basically an AM modulator whose signal can be received on the normal AM radio. It can also be used as an AM radio tester.

IC 555 (IC1) is used as a free running multivibrator whose frequency is set above 540 kHz. Here the circuit is designed for a frequency of around 600 kHz. The frequency of the multivibrator can be calculated as follows:

f=1.443/(R1+2R2)C1

where resistors R1 and R2 are in ohms, capacitor C1 is in microfarads, and frequency f is in hertz. This frequency can be changed by simply replacing R2 with a variable resistor or C1 with gang capacitors. But it may increase the complexity of the circuit. A condenser microphone is used for speaking.



The IC 555 multivibrator is used as a voltage-to-frequency converter. The output of the condenser microphone is given to pin 5 of IC1, which converts the input voltage or voice signal into its appropriate frequency at output pin 3. This frequency produces an electromagnetic wave, which can be detected by a nearby radio receiver, and you can hear your own voice in that radio. Note that the receiver should be AM type. If there is no noise in receiver, tune it to 600 kHz.

The circuit operates off a 9V regulated power supply or a 9V battery. For antenna, connect 2-3m long wire at pin 3.

0-100°C TEMPERATURE DETECTOR

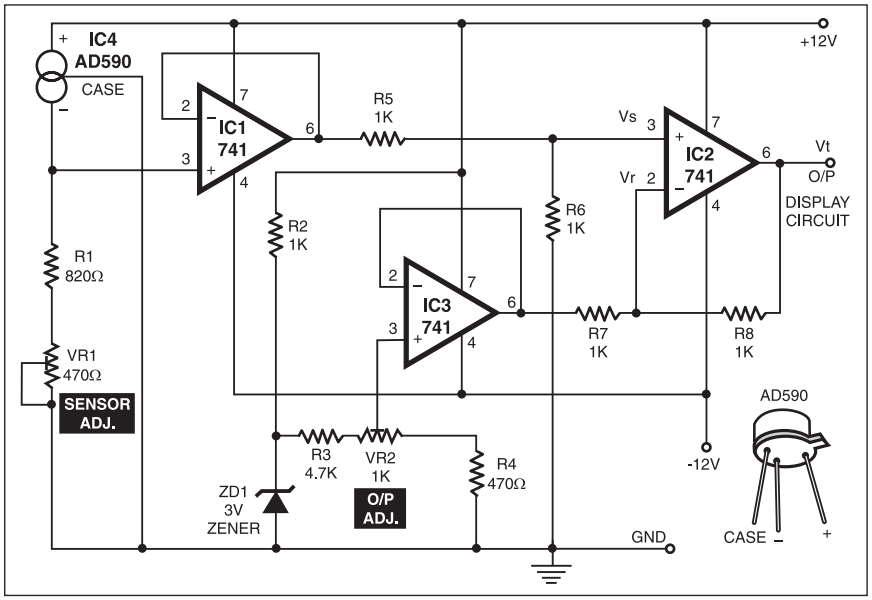
MANDEEP SINGH WALIA

Here's a temperature monitoring system capable of remote temperature reading without transmission losses in the conditioning circuit. The IC AD590 (IC4) is used as a temperature transducer. This semiconductor transducer provides a current output proportional to absolute temperature.

The sensor is a high-impedance, constant-current source over the temperature range of –55°C to 150°C. It has a nominal current sensitivity of 1 µA/Kelvin.

The transducer gives reading in mV/°C. The signal is conditioned and subtracted from a reference signal of 273 mV and the output of the circuit can be read directly in degree Celsius on the display. The sensor gives 1 µA current per degree rise in Kelvin, where 1°K=273+t. At room temperature (25°C), 1°K becomes 273+25=298. So the sensor outputs a current of 298 µA.

The current-to-voltage converter has a 1k resistor (R5). As the current is 298 µA, we get 298 µA x 1 kilo-ohm=298 mV. Now according to our design, we set a reference value of voltage that is to be compared with 273 mV. The voltages are fed to the buffer amplifiers to prevent loading of the source.



The circuit consists of three operational amplifiers (IC1 through IC3). IC1 and IC3 are used as voltage followers for the sensor output and the reference voltage, respectively. IC2 is used as a subtracter. The resultant of the voltages, i.e. Vs–Vr = Vt, is displayed on the LCD or on the multimeter.

For calibration, the sensor is first kept in ice (0°C) and VR2 is adjusted until the multimeter reads 0 mV. After this, the sensor is dipped in boiling water (100°C) and VR2 is adjusted until the multimeter reads 100 mV.

For any rise in temperature, the value of sensor voltage changes and then the resultant is found by subtracting the two voltages by IC2. Thus continuo-us monitoring of the temperature takes place.

PRECISION NULL DECTECTOR

SOMNATH CHAKRABARTI

Precise determination of null point in AC/DC bridges helps us to evaluate accurate values of circuit components like L and C. Here's a null detector that has the following merits:

• Same two input terminals of the detector can be employed for both DC and AC signals.

• There is no moving part like galvanometer. An array of ten LEDs operating in bar mode serves as visual means for detection of balance condition.

• The sensitivity of the instrument is quite high. Even when the out of balance is as low as 25 μV, detection is possible. This serves the purpose well for most measurements.

As is known, a sensor can be employed as detector for DC signal. It seems interesting at first sight that we can employ the same DC instrument in an AC circuit if we rectify the signal prior to applying it to the galvanometer. But here comes the problem. A silicon diode gets cut-off when the applied voltage falls below 0.7 volt and the galvanometer would read zero, although you are then far from the true balance point. The solution is to develop a precision rectifier using operational amplifier. This is the central theme of the null detector circuit presented here.

FET input operational amplifier IC1(Op-07), with very small offset voltage and extremely low bias current, serves as a non-inverting amplifier to increase the input signal level as shown in the figure. The gain of the amplifier is:

$A=(1+R2/R1)=(1+39k/1k)=40$...(1)

If V_i represents the input voltage and V_{o1} the output voltage of amplifier IC1, we have

$V_{o1}=AV_i$ ...(2)

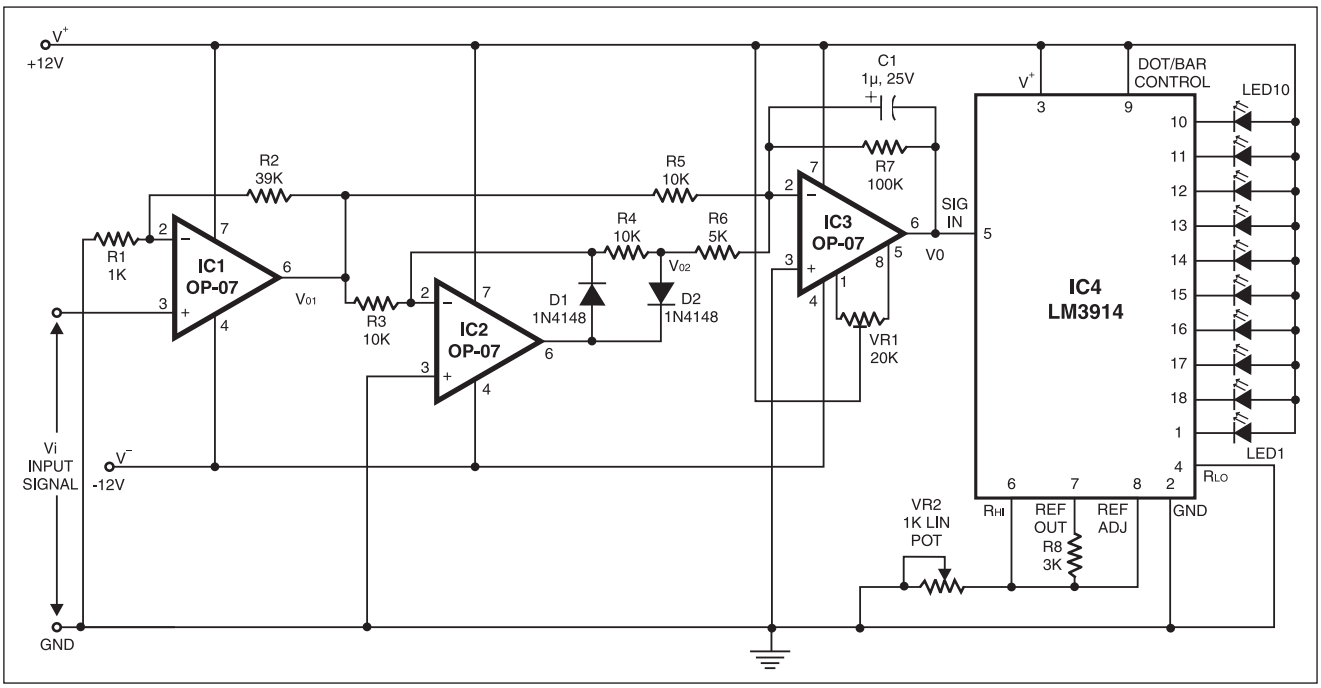
IC2 is used as a precision half-wave rectifier and the output of the rectifier is

$V_{o2} = -AV_i$, if V_i is positive

$= 0$, if V_i is negative ...(3)

IC3 plays a dual role. It acts as an inverting type adder and also as a low-pass filter. Low-pass filtering action is due to capacitor C1 in parallel to R7. For DC and low-frequency signals, the reactance of the capacitor is extremely high and the effective impedance in the feedback path is close to R7. For high frequencies, the capacitor looks short and the output is close to 0V. This is the low-pass filter action. The cut-off frequency of the low-pass filter is:

$f_c = 1/(2\pi R7C1) \approx 1.59$ Hz ...(4)



Hence the operating frequency of the detector for AC signals must be kept above cut-off frequency f_c. As far as only the adder action is concerned, output V_o is given by:

$V_o = (R7/R5 . V_{o1}) + (R7/R6 . V_{o2})$ ...(5)

Or,

$V_o = -(10V_{o1} + 20V_{o2})$ ...(6)

For DC voltage, $V_i = \pm V$ volt. Therefore, it follows:

$V_{o, DC} = 10AV$ ...(7)

For AC signal, V_i = a sin ωt,

$V_o = 10aA \sin \omega t$, $0 < \omega t < \pi$

Or

$V_o = -10aA \sin \omega t$, $\pi < \omega t < 2\pi$...(8)

Because of low-pass filtering action, the actual output is a positive DC voltage with very little ripple:

$V_{o, DC} = (2/\pi)\ (10aA) = 20aA/\pi$ ...(9)

Or,

$V_{o, DC} = 20aA/\pi(\sqrt{2}\ V_{i, rms})$

$\approx 9AV_{i, rms}$ ...(10)

IC LM3914 (IC4), commonly known as dot/bar display driver, has a set of ten comparators that detect ten voltage levels and drive ten LEDs accordingly. Here, IC4 is used in bar mode.

The internal circuitry of IC4 develops reference voltage V_{REF} of 1.25V between its pin 7 (REF OUT) and pin 8 (REF ADJ). A nominal current (I_{ADJ}) of typically 0.075 mA (maximum 0.120 mA) flows out of pin 8. The voltage difference between pin 6 (R_{HI}) and pin 4 (R_{LO}), here grounded, is applied to an internal string of ten 1k resistors and sets up ten evenly spaced comparator levels. With reference to the figure, the voltage at pin 6 is given by:

$V_{RHI} = (V_{REF}/R8 + I_{ADJ})VR2/(1+VR2/10)$ volt ...(11)

where all resistor values are in kilo-ohm and current in mA.

In the present case, the value of R8 is 3 kilo-ohms and VR2 is adjusted such that V_{RHI} becomes exactly 100 mV. (The approximate value of VR2 is 0.2 kilo-ohm.) The output DC voltage of IC3 is routed to pin 5 (SIGNAL IN) of IC4. As a consequence, the first LED will turn on if the voltage at pin 5 just exceeds 10 mV, followed by glowing of all the succeeding LEDs, one by one, for every 10mV rise in this voltage level. This provides visual means for precise determination of balance point.

Current I1, drawn out of pin 7, determines the brightness of the LEDs. Each glowing LED draws a current approximately ten times of I1, viz, 10 V_{REF} /R8 ≈ 4 mA.

For adjustment, make input voltage Vi=0. Connect a digital voltmeter in 200mV DC range between pin 5 of IC4 and ground (GND). Adjust the offset null by varying VR1 until the voltmeter reads zero.

Now connect a DC or AC signal source between the input pin 3 of IC1 and ground. Adjust the input voltage such that the voltmeter reads exactly 100 mV at pin 5 of IC4.

Shift the voltmeter to pin 6 of IC4 and trim VR2 such that the voltmeter reads 100 mV. At this instance, the tenth LED begins to glow without any flicker along with all the preceding nine LEDs. Your null detector is now ready for use.

The minimum input voltage $V_{i\ (min)}$ to which the null detector can respond obviously corresponds to the voltage that just turns on the first LED, viz, $V_{o, DC}$ =10 mV.

For DC signal, $V_{i, DC}$ (min) = 10 mV/10A = 10 mV/400 = 25 μV.

For AC signal, $V_{i, rms\ (min)}$ = 10 mV/9A =10 mV/360 = 27 μV.

SOUND SCANNER

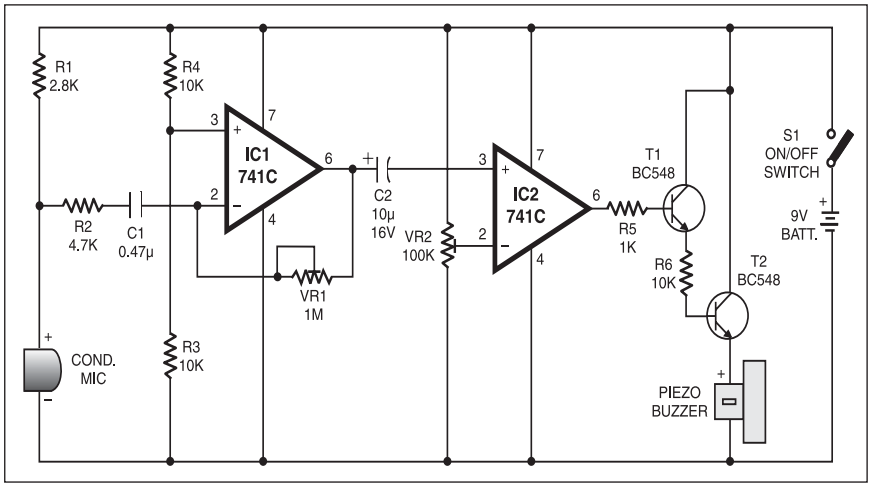
D. MOHAN KUMAR

This novel sound scanner sweeps all sound vi-brations in its vicinity and converts them into audible beeps. It can sense sound vibrations up to a distance of 6 metres and can be used to monitor sitouts, car porchs, and other places of your house. The sound scanner operates a beeper whenever the microphone detects a sound.

Sound vibrations are sensed by the input section comprising the condenser microphone and op-amp IC 741C (IC1). Resistor R1 determines the sensitivity of the microphone. Condenser microphone picks up sound vibrations and converts them into electrical signals, which are fed to the input (pin 2) of IC1 via coupling capacitor C1. Amplified signals from IC1 are taken to the non-inverting input pin 3 of IC2 (IC 741C) through C2. IC2 is configured as a comparator.

A reference voltage controlled by VR2 is applied to the inverting input pin 2 of IC2. The output of IC2 is used to trigger Darlington pair transistors T1 and T2. A piezobuzzer connected to the emitter of T2 produces audible beeps as the microphone senses sound.

The circuit can be easily assembled



on a common PCB or Veroboard. Adjust VR1 to get the maximum gain of IC1. Adjust VR2 to get the maximum sensitivity of IC2.

If a continuous beep is heard through the piezobuzzer, adjust the wiper of VR2 towards the ground line. Keep the piezobuzzer inside the room and the sensor in the place that is to be monitored. Connect the condenser microphone using a two-core shielded wire and enclose it in a small case to increase its sensitivity. Battery operation is recommended as the circuit may pick up noise from AC mains.

CLAP SWITCH

MOHAMMAD USMAN QURESHI

Here's a clap switch free from false triggering. To turn on/off any appliance, you just have to clap twice. The circuit changes its output state only when you clap twice within the set time period. Here, you've to clap within 3 seconds.

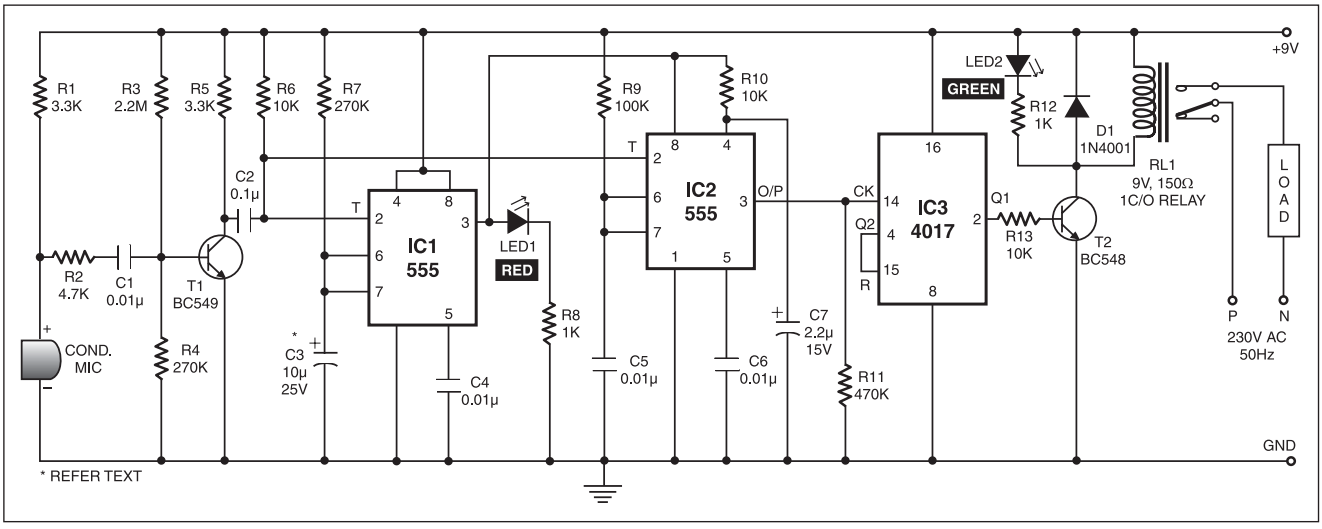
The clap sound sensed by condenser microphone is amplified by transistor T1. The amplified signal provides negative pulse to pin 2 of IC1 and IC2, triggering both the ICs. IC1, commonly used as a timer, is wired here as a monostable multivibrator. Trigging of IC1 causes pin 3 to go high and it remains high for a certain time period depending on the selected values of R7 and C3. This 'on' time (T) of IC1 can be calculated using the following relationship:

T = 1.1R7 x C3 Seconds

Where R7 is in ohms and C3 in microfarads. On first clap, output pin 3 of IC1 goes high and remains in this standby position for the preset time. Also, LED1 glows for this period. The output of IC1 provides supply voltage to IC2 at its pins 8 and 4. Now IC2 is ready to receive the triggering signal. Resistor R10 and capacitor C7 connected to pin 4 of IC2 prevent false triggering when IC1 provides the supply voltage to IC2 at first clap.

On second clap, a negative pulse triggers IC2 and its output pin 3 goes high for a time period depending on R9 and C5. This provides a positive pulse at clock pin 14 of decade counter IC 4017 (IC3). Decade counter IC3 is wired here as a bistable.

Each pulse applied at clock pin 14 changes the output state at pin 2 (Q1) of IC3 because Q2 is connected to reset pin 15. The high output at pin 2 drives transistor T2 and also energises relay RL1. LED2 indicates activation of relay RL1 and on/off status of the appliance. A freewheeling diode (D1) prevents damage of T2 when relay de-energises.



INFRARED REMOTE CONTROL TIMER

DIPANJAN BHATTACHARJEE

This infrared remote control timer can be used to turn an appliance on/off for a period of 0.11 second to 110.0 seconds.

The circuit comprises two sections, namely, the transmitter section and the receiver section.

Fig. 1 shows the IR transmitter section. The astable multivibrator NE555 (IC1) is used to generate a 10kHz modulated IR signal. The output of IC1 is connected to the base of pnp transistor T1 via resistor R2. Two infrared LEDs (IR1 and IR2) are connected in series between the collector (via resistor R3) and ground.

When switch S1 is pressed, the IR LEDs transmit the modulated IR signal of 10-11 kHz. This frequency can be changed with the help of VR1 potmeter.

In the receiver section shown in Fig. 2, two photodiodes (IR3 and IR4) receive the IR signal transmitted by the IR transmitter. Transistors T2 and T3 amplify the weak signal. The amplified signal is filtered by capacitors C6 and C7. The amplified and filtered signal is now fed to the inverting input pin 2 of op-amp IC2 (IC 741). The output of IC2 is further connected to trigger pin 2 of timer NE555 (IC3) that is used as a monostable multivibrator whose frequency may be varied with the help of potmeter VR3.

When switch S1 of the transmitter is pressed, the modulated IR rays are generated, which are received by photodiodes in the receiver section and amplified by the amplifier circuit. The output of op-amp goes low to trigger the monostable. Then high output at pin 3 of IC3 activates the two-changeover relay RL via transistor T3 (BC548) for a preset time.

The on/off time can be set in the timer with the help of VR3 and C10. Switch S2 is used to reset the monostable. If you want to turn the appliance on for a preset time, connect the appliance via relay RL(a). On the other hand, if you want to turn the appliance off for a preset time, connect the appliance via relay RL(b). The timer can be reset by pressing reset switch S2.

The circuit works up to 3 metres without using any focusing lens. However, you can increase the operating range by using focusing lens.

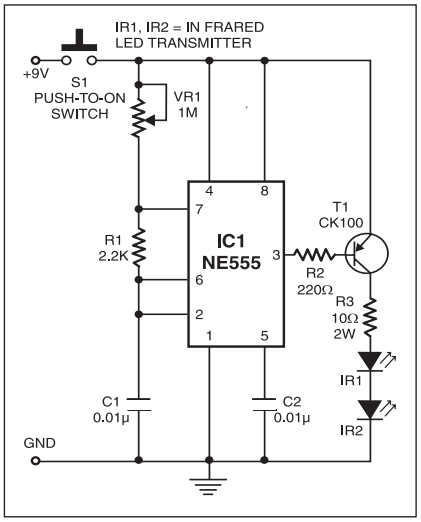


Fig. 1: IR transmitter section

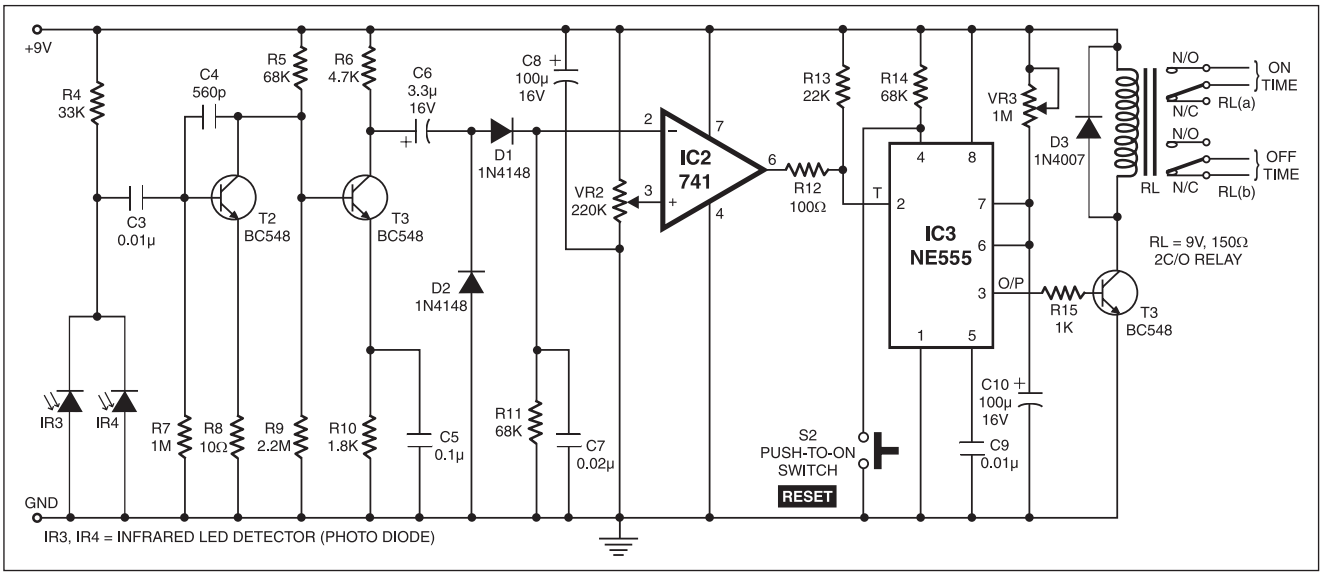


Fig. 2: IR receiver section

EARTH FAULT PROTECTOR

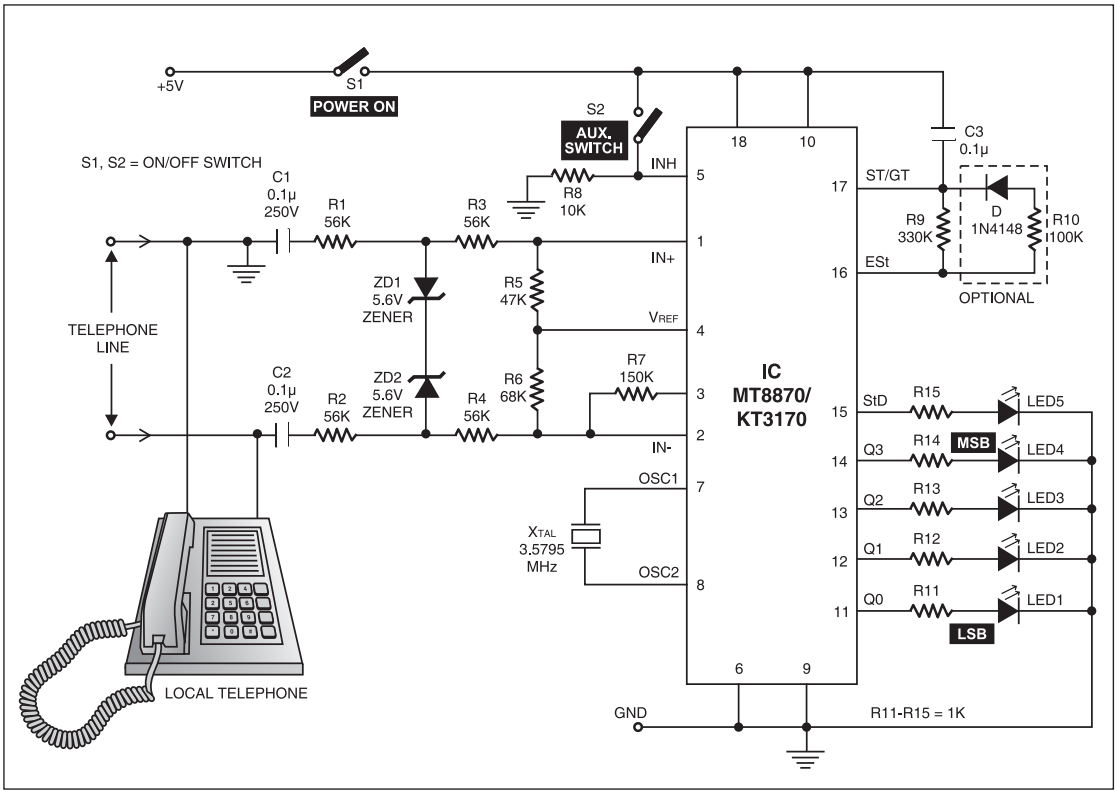
V. DAVID

This circuit trips immediately when any earth fault occurs, whatever be the load current. It is designed for a load current of 5 amp. For higher load currents, the ratings of the power sockets and relay contacts should be increased or a separate relay (RL) with a higher contact rating should be used.

Whenever earth fault occurs, the LED inside optocoupler MCT2E (IC2) glows, so the phototransistor inside IC2 conducts to provide a low triggering pulse at pin 2 of IC1. This triggers IC1 and its output at pin 3 goes high to energise relay RL, which disconnects the power supply from line socket to load socket via RL(a) contact. Relay RL gets latched by receiving the 9V DC supply through RL(b) contact even though output pin 3 goes low. Here, NE555 (IC1) is wired in the monostable mode with 'on' period of approximately one second to avoid chattering or contact bounce

Fuse Ratings for Different Wires

Current rating (amp) of fuse	Copper SWG	Aluminium SWG	Tin SWG	Tin alloy SWG	Lead SWG
1	47	46	37	45	35
2	43	42	31	29	30
3	41	39	28	27	27
4	39	37	26	25	25
5	38	36	25	23	23
10	33	30	21	19	20



ing LEDs will glow. Thus, a non-defective IC should indicate proper binary values corresponding to the decimal number pressed on your telephone keypad.

To test the DTMF IC 8870/ KT3170, proceed as follows:

1. Connect local telephone and the circuit in parallel to the same telephone line.

2. Switch on S1. (Switch on auxiliary switch S2 only if keys A, B, C, and D are to be used.)

3. Now push key '*' to generate DTMF tone.

4. Push any decimal key from the telephone keypad.

5. Observe the equivalent binary as shown in the table.

6. If the binary number implied by glowing of LED1 to LED4 is equivalent to the pressed key number (decimal/A, B, C, or D), the DTMF IC 8870 is correct.

Keys A, B, C, and D on the telephone keypad are used for special signalling and are not available on standard pushbutton telephone keypads. Pin 5 of the IC is pulled down to ground through resistor R8. Switch on auxiliary switch S2. Now the high logic at pin 5 enables the detection of tones representing characters A, B, C, and D.

The Status of LEDs on Pressing Keys on the Telephone Keypad

Key No.	LED4 (MSB)	LED3	LED2	LED1 (LSB)
1	Off	Off	Off	On
2	Off	Off	On	Off
3	Off	Off	On	On
4	Off	On	Off	Off
5	Off	On	Off	On
6	Off	On	On	Off
7	Off	On	On	On
8	On	Off	Off	Off
9	On	Off	Off	On
0	On	Off	On	Off
A	On	On	Off	On
B	On	On	On	Off
C	On	On	On	On
D	Off	Off	Off	Off

Note. *1. LED5 blinks momentarily whenever any key is pressed.*
2. On = 1, while Off = 0

The LEDs connected via resistors R11 to R14 at pins 11 through 14, respectively, indicate the output of the IC. The tone-pair DTMF (dual-tone multi-frequency) generated by pressing the telephone button is converted into binary values internally in the IC. The binary values are indicated by glowing of LEDs at the output pins of the IC. LED1 represents the lowest significant bit (LSB) and LED4 represents the most significant bit (MSB).

So, when you dial a number, say, 5, LED1 and LED3 will glow, which is equal to 0101. Similarly, for every other number dialled on your telephone, the corresponding LEDs will glow. Thus, a non-defective IC should indicate proper binary values corresponding to the decimal number pressed on your telephone keypad.

To test the DTMF IC 8870/ KT3170, proceed as follows:

1. Connect local telephone and the circuit in parallel to the same telephone line.

2. Switch on S1. (Switch on auxiliary switch S2 only if keys A, B, C, and D are to be used.)

3. Now push key '*' to generate DTMF tone.

4. Push any decimal key from the telephone keypad.

5. Observe the equivalent binary as shown in the table.

6. If the binary number implied by glowing of LED1 to LED4 is equivalent to the pressed key number (decimal/A, B, C, or D), the DTMF IC 8870 is correct.

Keys A, B, C, and D on the telephone keypad are used for special signalling and are not available on standard pushbutton telephone keypads. Pin 5 of the IC is pulled down to ground through resistor R8. Switch on auxiliary switch S2. Now the high logic at pin 5 enables the detection of tones representing characters A, B, C, and D.

PULSE GENERATOR

A. JEYABAL

This circuit is very useful while checking/operating counters, stepping relays, etc. It avoids the procedure of setting a switch for the required number of pulses. By pressing appropriate switches S1 to S9, one can get 1 to 9 negative-going clock pulses, respectively.

Schmitt trigger NAND gate N1 of IC2, resistor R1, and capacitor C1 are wired to produce clock pulses. These pulses are taken out through NAND gate N3 that is controlled by decade counter CD4017 (IC1).

Initially no switch from S1 to S9 is

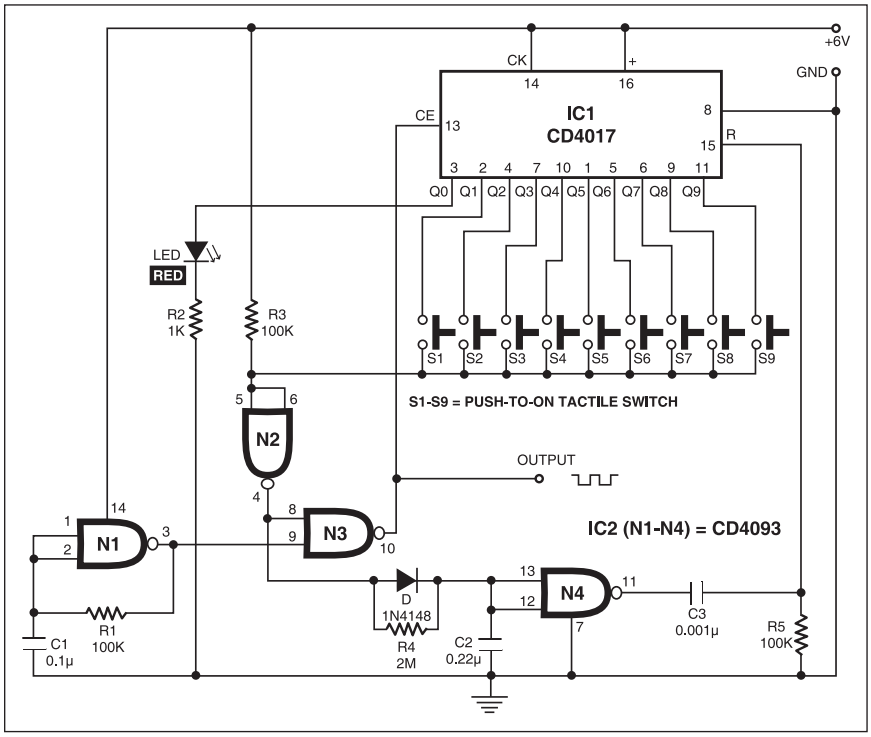
depressed and the LED is glowing. As pins 5 and 6 of NAND gate N2 are pulled up by resistor R3, its output pin 4 goes low. This disables NAND gate N3 to take its output pin 10 to high state, and no pulse is available.

IC1 is a decade counter whose Q outputs normally remain low. When clock pulses are applied, its Q outputs go high successively, i.e. Q0 shifts to Q1, Q1 shifts to Q2, Q3 shifts to Q4, and so on.

If any one of switches S1 through S9, say, S5 (for five pulses), is momentarily depressed, pins 5 and 6 of NAND gate N2 go low, making its output pin 4 high, which fully charges capacitor C2 via diode D. At the same time, this high output of N2 enables NAND gate N3 and clock pulses come out through pin 10. These are the required number of pulses used to check our device.

The clock pulses are fed to clock-enable pin 13 of IC1, which starts counting. As soon as output pin 1 (Q5) of IC1 turns high, input pins 5 and 6 of NAND gate N2 will also become high via switch S5 because high-frequency clock allowed five pulses during momentary pressing. This high input of N2 provides low output at pin 4 to disable NAND gate N3 and finally no pulse will be available to advance counter IC1.

Before the next usage, counter IC1 must be in the standby state, i.e. Q0 output must be in the high state. To do this, a time-delay pulse generator wired around NAND gate N4, resister R4, diode D, capacitor C2, and differentiator circuit comprising C3 and R5 is used.



When output pin 4 of NAND gate N2 is low, it discharges capacitor C2 slowly through resistor R4. When the voltage across capacitor C2 goes below the lower trip point, output pin 11 of NAND gate N4 turns high and a high-going sharp pulse is produced at the junction of capacitor C3 and resistor R5. This sharp pulse resets counter IC1 and its Q0 output (pin 3) goes high. This is represented by the glowing of LED.

Ensure the red LED is glowing before proceeding to get the next pulse. Press any of the switches momentarily and the LED will glow. If the switch is kept pressed, the counter counts continuously and you cannot get the exact number of pulses.

INTRUDER DETECTOR USING LASER TORCH

G. SUSINDER RAJAN

Here is a simple, low-cost intruder detector that uses an invisible laser beam to detect the intruder. The laser beam is produced using a 3V DC or 4.5V DC laser pointer or torch that is available in the market. The 3V DC or 4.5V DC power supply for the laser transmitter can also be given using a bridge rectifier or full-wave rectifier.

Fig. 1 shows the block diagram of the complete unit comprising the transmitter and receiver sections. The laser beam from the transmitter after reflection from various mirrors (M1 through M6, as shown in Fig. 1) is made to fall on the photodetector in the receiver circuit.

Once the laser beam is positioned, the receiver circuit is powered by closing switch S. An alarm unit operating on 230V AC is connected to the relay RL in the receiver circuit.

When an intruder interrupts the path of the beam or switches off the laser torch, the alarm unit becomes activated. The alarm unit remains activated until reset switch S is opened. To activate the alarm

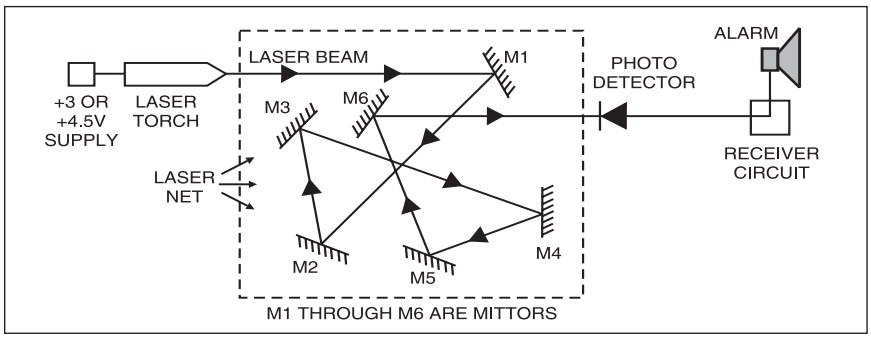


Fig. 1: Block diagram of intruder detector using laser torch

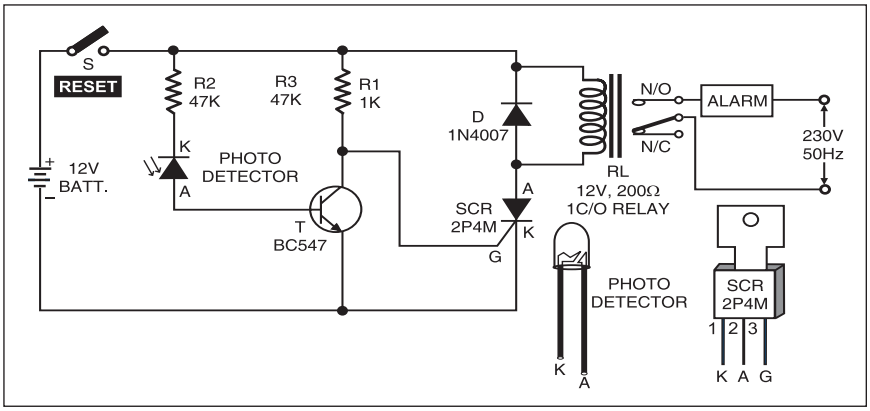


Fig. 2: Receiver circuit

circuit again, reset switch S should be closed. The total distance travelled by the laser beam should be less than 800 m for 4.5V laser torch and 500 m for 3V laser torch.

The circuit of the receiver is shown in Fig. 2. When reset switch S is closed, the circuit is powered on. As the laser beam falls on the photodetector, transistor T (BC547) conducts, resulting in the collector being pulled down to ground potential. Thus no current flows to the gate of the SCR and it remains off.

Once the path of the laser beam is interrupted, the base current of the transistor becomes very low and the transistor is driven to cut-off. Now the current starts to flow through resistor R1 and to the gate of SCR. Hence the SCR is fired and it begins to conduct. Thus relay RL connected to the anode of SCR is switched on and the alarm is activated. The alarm sounds until reset switch S is opened to turn off power to the circuit.

EFY Lab note. We tested the circuit using only one mirror and found its range to be 25-30 metres. The range depends on the intensity of laser beam falling on the photodetector.

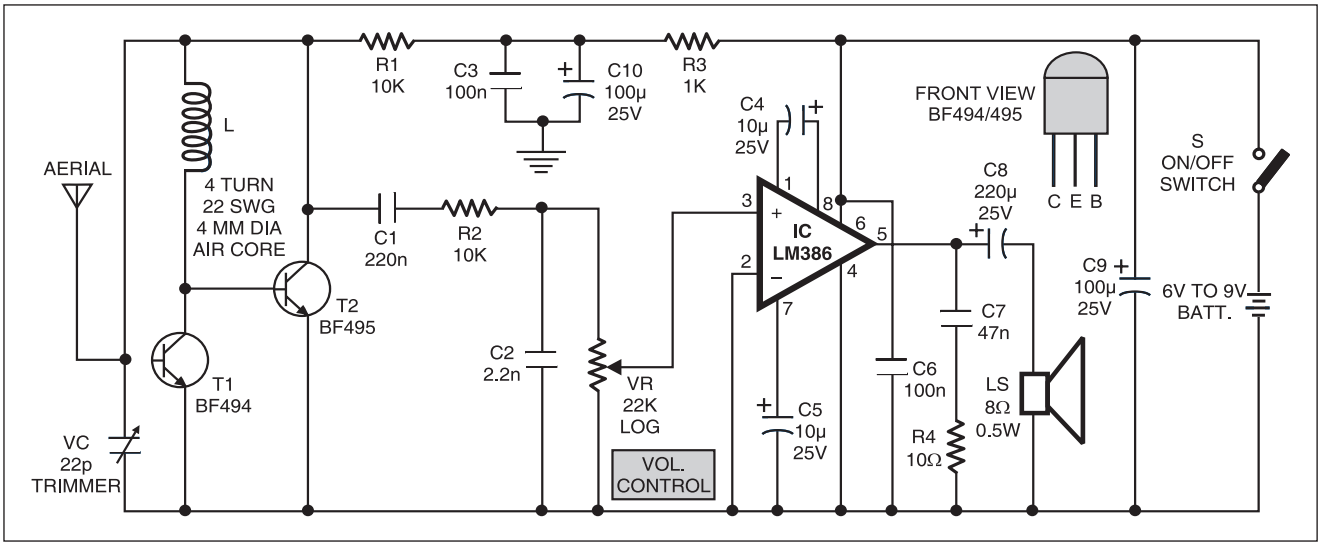
SIMPLE FM RECEIVER

D. PRABAKARAN

Frequency modulation is used in radio broadcast in the 88-108MHz VHF band. This bandwidth range is marked as FM on the band scales of radio receivers, and the devices that are able to receive such signals are called FM receivers. The FM radio transmitter has a 200kHz wide channel. The maximum audio frequency transmitted in FM is 15 kHz as compared to 4.5 kHz in AM. This allows much larger range of frequencies to be transferred in FM and thus the quality of FM transmission is significantly higher than of AM transmission.

Here's a simple FM receiver with minimum components for local FM reception. Transistor BF495 (T2), together with a 10k resistor (R1), coil L, 22pF variable capacitor (VC), and internal capacitances of transistor BF494 (T1), comprises the Colpitts oscillator. The resonance frequency of this oscillator is set by trimmer VC to the frequency of the transmitting station that we wish to listen. That is, it has to be tuned between 88 and 108 MHz. The information signal used in the transmitter to perform the modulation is extracted on resistor R1 and fed to the audio amplifier over a 220nF coupling capacitor (C1).

You should be able to change the capacitance of the variable capacitor from a couple of picofarads to about 20 pF. So, a 22pF trimmer is a good choice to be used as VC in the circuit. It is readily available



in the market. If you are using some other capacitor that has a larger capacitance and are unable to receive the full FM bandwidth (88-108 MHz), try changing the value of VC. Its capacitance is to be determined experimentally.

The self-supporting coil L has four turns of 22 SWG enamelled copper wire, with air core having 4mm internal diameter. It can be constructed on any cylindrical object, such as pencil or pen, having a diameter of 4 mm. When the required number of turns of the coil has reached, the coil is taken off the cylinder and stretched a little so that the turns don't touch each other.

Capacitors C3 (100nF) and C10 (100μF, 25V), together with R3 (1k), comprise a band-pass filter for very low frequencies, which is used to separate the low-frequency signal from the high-frequency signal in the receiver.

You can use the telescopic antenna of any unused device. A good reception can also be obtained with a piece of isolated copper wire about 60 cm long. The optimum length of copper wire can be found experimentally.

The performance of this tiny receiver depends on several factors such as quality and turns of coil L, aerial type, and distance from FM transmitter. IC LM386 is an audio power amplifier designed for use in low-voltage consumer applications. It provides 1 to 2 watts, which is enough to drive any small-size speaker. The 22k volume control (VR) is a logarithmic potentiometer that is connected to pin 3 and the amplified output is obtained at pin 5 of IC LM386. The receiver can be operated off a 6V-9V battery.

PARALLEL TELEPHONE WITH SECRECY AND CALL PREVENTION

SANJIB CHOWDHURY

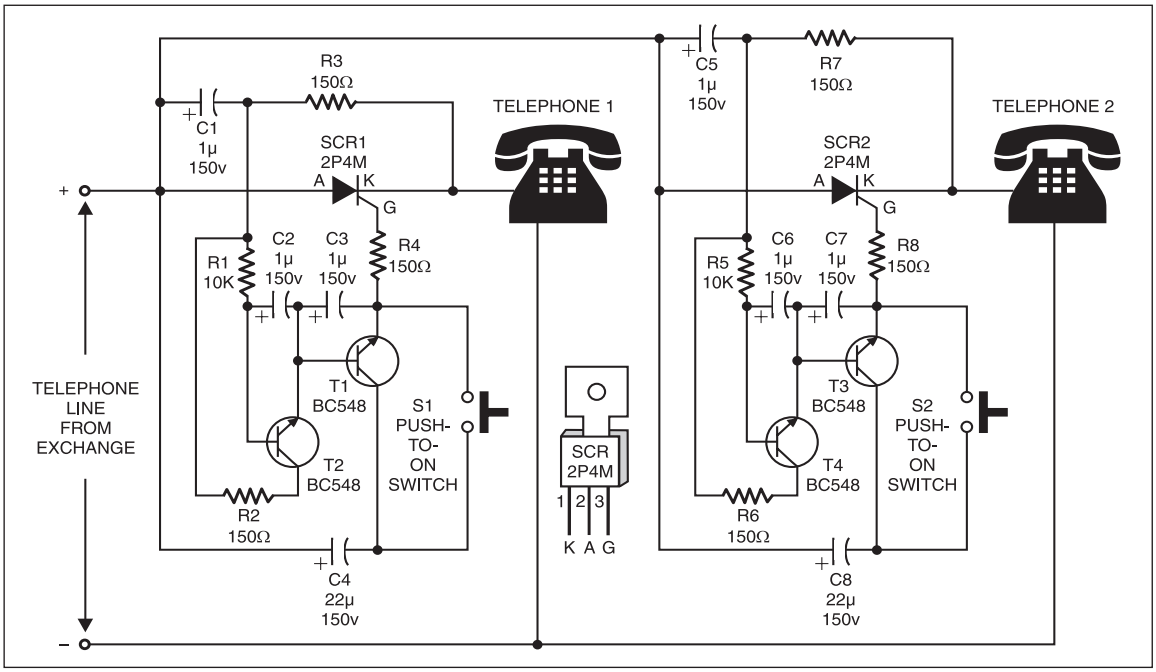
This circuit provides secrecy when two or more telephones are connected in parallel to a telephone line. The circuit also prevents incoming calls to as well as outgoing calls from other telephones connected in parallel, except from the one lifted first.

When someone picks up the handset of the telephone connected in parallel to the original (master) phone for making an outgoing call, no dial tone is heard and the phone appears to be dead. But when a call comes, the ring signal switches the SCRs 'on' and conversation can be carried out. As soon as the handset is kept on the hook, the SCR goes off and the telephone can again only receive incoming calls.

When a call comes, conversation can be made only from the telephone which is lifted up first. To carry out conversation from the other telephone, the handset of the telephone that was lifted up first has to be placed on the hook and then the push-to-on switch of the associated circuit of the other telephone has to be pressed after lifting up its handset. Thus the circuit ensures privacy because both the telephones cannot be active at the same time.

Those who are don't need parallel telephones can rig up the associated circuit of a single telephone to work as an outgoing call preventer. An outgoing call can be made only when one lifts up the handset and pre-sses the push-to-on switch of its associated circuit.

The polarity of the telephone line can be determined by a multimeter. To avoid confusion, a bridge rectifier can be used at the input of the circuit.



PC-DRIVEN LED DISPLAY

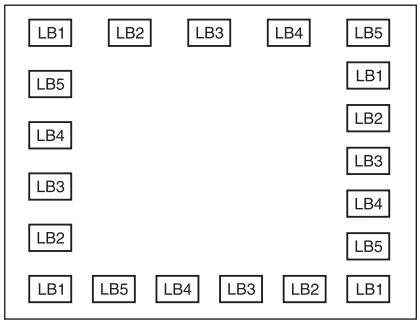
R. KARTHICK

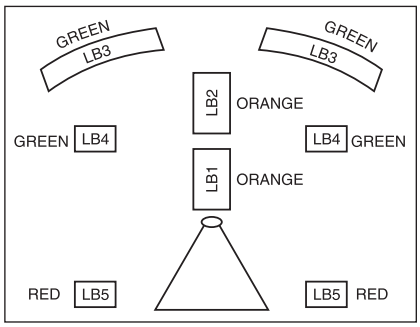
Here is a circuit to generate sequentially running light effects using a simple program written in C. The output of the program is taken from the LPT port of a PC and then fed to the interfacing circuit for the LED display. The outputs of the interfacing circuit are decoded, inverted, and then connected to LEDs through optocouplers.

The interfacing circuit along with the 25-pin parallel port is shown in Fig. 1. IC1 (74LS138) is a high-speed 1-of-8 decoder/demultiplexer. In the circuit, only five outputs (pins 10 through 14) of IC1 are used. These outputs are inverted using NOT gate IC2 (7404). Optocouplers IC3 through IC7 (MCT2E) are used to prevent the circuit from damage in the event of short circuit in the load. Thus the loads comprising LED blocks LB1 through LB5 are isolated from the interfacing circuit including the PC. Each LED block contains a number of LEDs connected in series.

The LEDs can be arranged, for instance, to display a sequential running light and fountain pot as shown in Figs 2 and 3. The colour of LED blocks can be green and red alternately.

The program can be compiled and run through Turbo C compiler. In the program, the function outport(p,x) is used, where 'p' is the address of the controller port and 'x' is the value sent to it. Here, controller port LPT1 is used, whose base address is 378H. If LPT2 is to be used, the base address must be 278H. When the delay time delay() is increased, the running speed of LEDs decreases and vice versa.

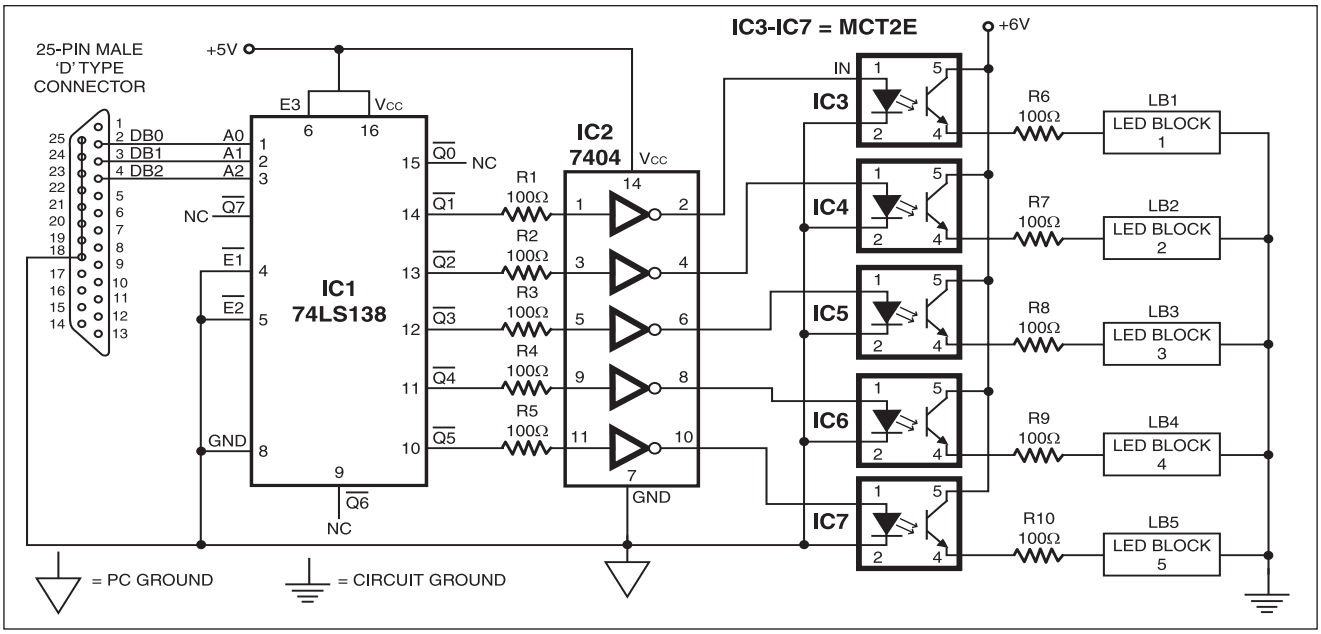




In the circuit, 100-ohm fixed resistors R6 through R10 can be replaced with 100-ohm or 200-ohm presets for decreasing and increasing the intensity of LEDs.

The program for LED display is as follows:

```
#include<stdio.h>
#include<conio.h>
#include<dos.h>
void main()
{
int i, p=0x0378, x=10;
clrscr();
for(i=0;i<600;i++)
{
outport(p,x);
x++;
delay(500);
if(x==5)
x=1;
}
printf("By Karthi");
getch();
}
```



SOLIDSTATE SIGNAL LAMP

T.K. HAREENDRAN

This solidstate signal lamp is a good alternative to revolving mechanical signaling lamps used in ambulances. It uses ultra-bright blue LEDs for signaling. However, you can replace blue LEDs with bright red LEDs for other applications.

Fig. 1 shows the circuit of the solidstate signal lamp, and its construction plan is shown in Fig. 2. Interconnect point A, +12V supply, and GND shown in Figs 1 and 2.

Before the vehicle driver closes the front door, switch S2 is in normally opened (N/O) condition as per the mechanical arrangement. After closing the door, closing of switch S1 powers up the whole circuit via polarity-protection diode D. Gates N1 and N2 are immediately set in the bistable mode by R2 and C1 and the resulting high output at pin 3 of gate N2 enables the low-frequency oscillator wired around gate N3. Components C2, R3, and VR determine the oscillator frequency, which can be changed to some extent by varying VR.

Consequently, IC2 is clocked by gate N3 and the output of IC2 changes sequentially from Q0 (pin 3) to Q3 (pin 7). These outputs are used to switch four LED driver transistors T1 through T4 via resistors R4 through R7, respectively. The sets of three ultra-bright blue LEDs, each connected to the collectors of driver transistors T1 through T4, respectively, glow sequentially as the outputs Q0 to Q3 go high. The result is a revolving blue light effect, the same as that obtained from a mechanical signal lamp.

Whenever the front door is opened, switch S2 gets activated (opens) and makes the bistable to immediately reset the output of gate N2. Thus the clock oscillator is disabled and the signaling function is stopped.

For good visible effect, high-quality miniature reflectors should be used.

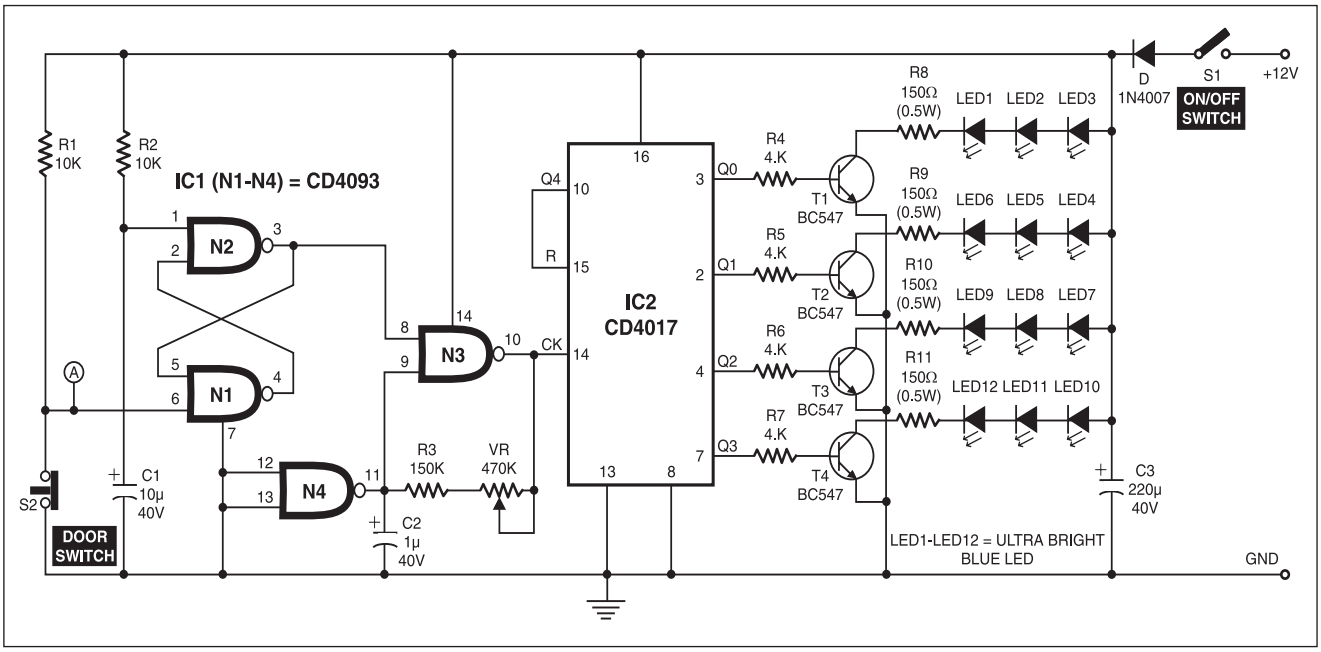


Fig. 1: Circuit of solidstate signal lamp

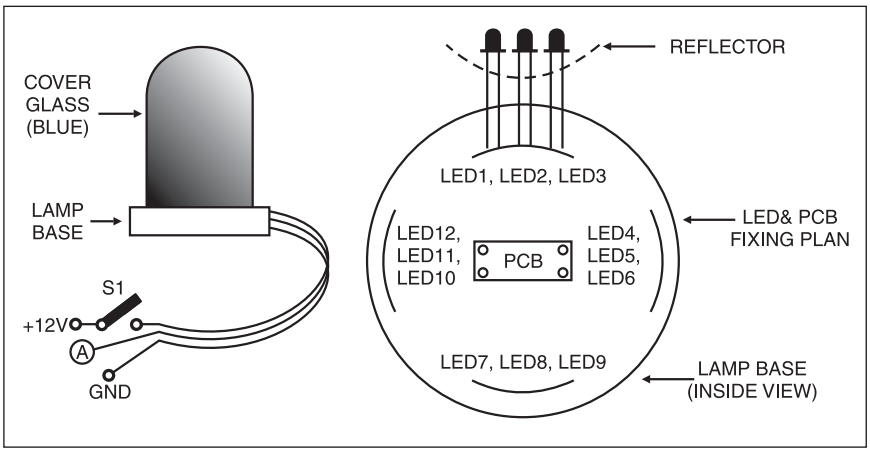


Fig. 2: Construction plan

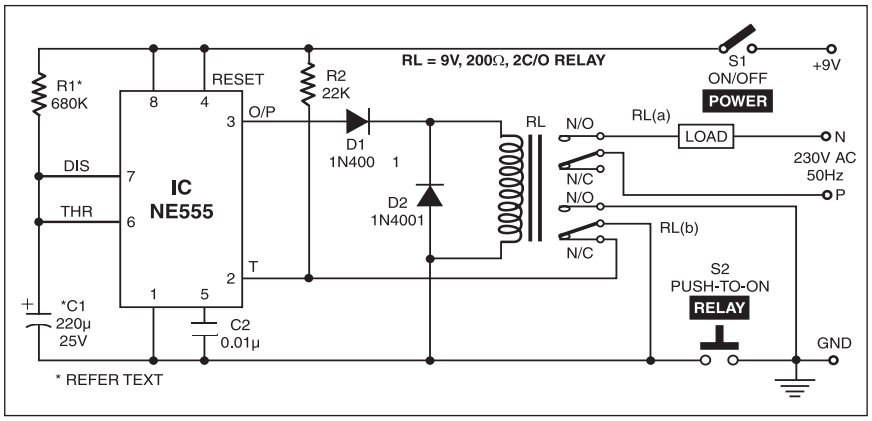
FALSE TRIGGERING ELIMINATOR FOR TIMER 555

AJAY SINGH

Normally, false triggering of timer IC 555 takes place during power on, resulting in unwanted output, which starts the timer's time cycle. The circuit becomes inefficient especially when the load has to be energised only when desired. Here is a simple circuit to eliminate false triggering of timer 555.

The circuit is wired in monostable mode and grounded via N/O contact of RL(b) as well as switch S2. When power switch S1 is switched on, the circuit will not be grounded until switch S2 is momentarily pressed. To provide the triggering pulse to the timer at pin 2, press switch S2 momentarily.

To activate the relay for operating the load, switch on the power to the circuit by pressing switch S1 and then S2 momentarily. The resulting output at pin 3 goes high and energises relay RL to operate the load. Now after momentarily pressing S2, the circuit remains on as GND gets connected to N/O contact of RL(b). At the same time, pin 2 is disconnected from N/C contact of RL(b), which prevents further triggering.



The time period of relay energisation (approximately 3 minutes) can be easily changed by changing the values of resistor R1 and capacitor C1 according to the requirement to operate the load. At the end of the cycle, the relay gets de-energised and the circuit becomes ungrounded again.

TRANSISTOR TESTER

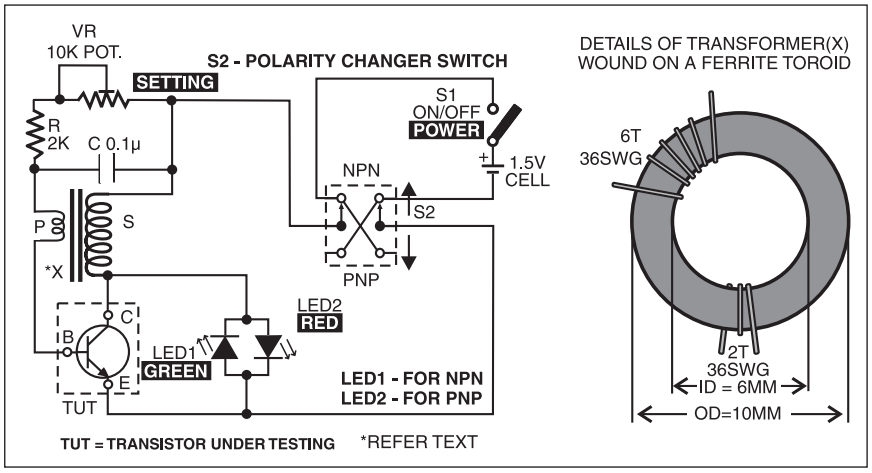
T.A. BABU

This simple bipolar transistor tester comes handy on the hobbyist work-bench. It quickly checks whether a transistor is functional or not, and helps to identify whether a transistor is npn or pnp without a trip to the data book. It uses very few components.

The circuit is basically a self-oscillating flyback converter circuit. Initially, move switch S1 to 'on' state to provide power supply to the circuit. Now on sliding polarity-changer switch S2 towards npn or pnp transistor, a base current flows to the transistor under test via potmeter VR, resistor R, and primary winding P. The current flowing through primary winding P induces voltage in secondary winding S, and the voltage at the transistor's collector rises.

When the collector voltage exceeds the forward bias voltage of the LED (red or green), the LED glows. For npn transistor green LED (LED1) glows, and for pnp transistor red LED (LED2) glows. The red and green LEDs are connected in parallel with reverse polarity.

Winding details of the transformer (X) are shown in the figure. The transformer is wound on a ferrite toroid of 10mm outer diameter, 6mm inner diameter, and 4mm thickness. Primary winding has two turns of 36SWG, while secondary winding has six turns of 36SWG wire on the ferrite toroid. The wire's gauge is not an important parameter. If the tester does not work, interchange the



connections of either primary or secondary windings.

For testing the working of transistor, first switch off the power supply to the circuit using switch S1 and then insert a good known transistor in the terminals marked C, B, and E as shown in the figure. Now switch on S1 to provide power supply and slide polarity-changer switch S2 towards npn or pnp transistor position accordingly. Adjust potmeter VR until oscillations are achieved, i.e. LED1 or LED2 glows. If green LED glows, it means the npn transistor is okay. On the other hand, if red LED glows, it means the pnp transistor is okay. An open or shorted transistor will not light any of the LEDs.

A single 1.5V cell lasts for several hundred tests, as the tester requires very little current. Keep power switch S1 in 'off' state when the circuit is not in use.

VOLTAGE-BASED CONTROLLER FOR SWITCHES

RAJ K. GORKHALI

Here's a simple circuit for controlling four switches from a distance through just a pair of wires.

In the circuit, the inverting inputs (pin 2) of operational amplifiers IC1 through IC4 are set to refrence voltages of +12V, +9V, +6V, and +3V, respectively, through a chain of four 1k resistors (R1 through R4).

The reference voltage (V_{REF}) can be simply calculated by the following relationship:

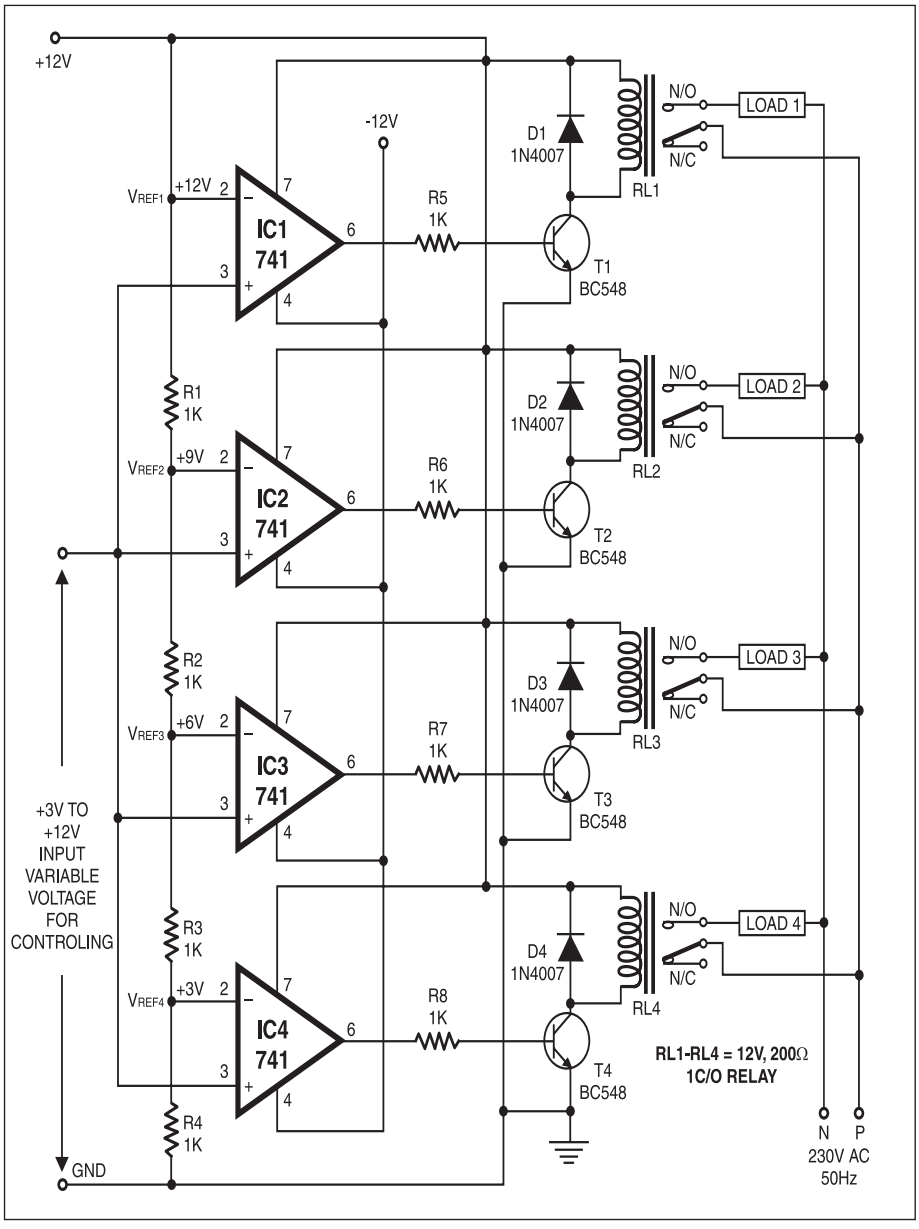
$$V_{REF} = \frac{\text{Total applied voltage} \times \text{Resistance across reference voltage}}{\text{Total resistance}}$$

For example, reference voltage V_{REF3} is calculated as follows:

$$V_{REF3} = \frac{12V \times (R3+R4)}{R1+R2+R3+R4} = \frac{12V \times 2k}{4k} = 6V$$

The non-inverting inputs (pin 3) of the four op-amps (IC1 through IC4) are tied together and connected to a pair of wires that provide +3V to +12V input voltage for controlling the switches.

Four 12V, 200-ohm, single-changeover relays are connected to four BC548 relay driver transistors (T1 through T4) via resistors R5 through R8, respectively. These relays energise depending on the voltage present at the controlling voltage input terminal; for example, relay RL4 energises when controlling voltage input of +3V is available at non-inverting pin 3 of IC4. Four electrical equipment can be connected to the terminals of the relays through the 220V AC, 50Hz mains.



BURGLAR ALARM SYSTEM

T.A. BABU

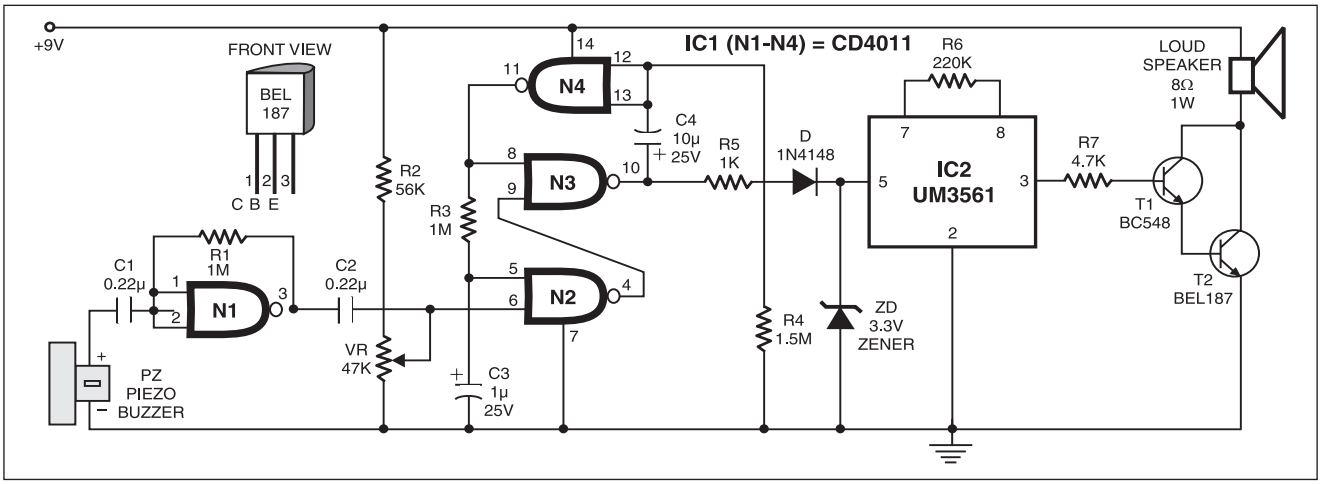
This inexpensive, compact alarm system scares away the thief and draws the attention of your neighbours if the thief attempts to break your protected door or window to enter your house. It can also be installed in offices and garages to prevent unauthorised entry.

In the alarm circuit, N1 acts as a low-frequency amplifier. When piezoelectric diaphragm within piezobuzzer PZ vibrates, the developed voltage is superimposed with an adjustable DC threshold level. The amplified signal triggers the monostable oscillator formed by N3 and N4.

The high output voltage from pin 10 of N3 is applied via diode D to pin 5 of sound generator IC2 (UM3561). (Resistor R6 connected across pins 7 and 8 of IC2 determines the frequency of the inbuilt oscillator of IC2.) As a result, IC2 starts generating the audio signal output at pin 3.

The output voltage from IC2 is further amplified by the Darlington pair of transistors T1 and T2. The amplified output of the Darlington pair drives the loudspeaker. The alarm lapses after a time period determined by the time constant of R4 and C4.

Retriggering of the monostable oscillator is prevented for a brief period by R3 and C3. Piezobuzzer PZ is used as a vibration sensor. PZ should be mounted such that it can sense vibration due to tampering.



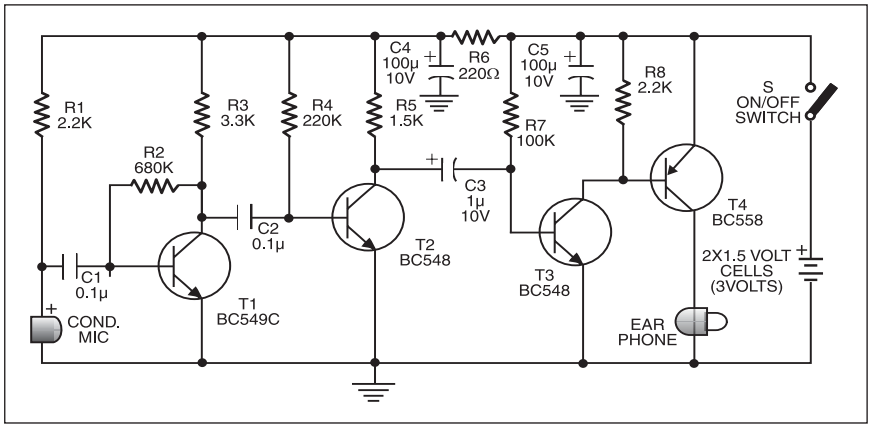
LOW-COST HEARING AID

PRADEEP G.

Commercially available hearing aids are quite costly. Here is an inexpen-sive hearing aid circuit that uses just four transistors and a few passive components.

On moving power switch S to 'on' position, the condenser microphone detects the sound signal, which is amplified by transistors T1 and T2. Now the amplified signal passes through coupling capacitor C3 to the base of transistor T3. The signal is further amplified by pnp transistor T4 to drive a low-impedance earphone. Capacitors C4 and C5 are the power supply decoupling capacitors.

The circuit can be easily assembled on a small, general-purpose PCB or a Vero board. It operates off a 3V DC supply. For this, you may use two small 1.5V cells. Keep switch S to 'off' state when the circuit is not in use. To increase the sensitivity of the condenser microphone, house it inside a small tube.



OVER-/UNDER-VOLTAGE PROTECTION OF ELECTRICAL APPLIANCES

C.H. VITHALANI

This circuit protects refrigerators as well as other appliances from over- and under-voltage. Operational amplifier IC LM324 (IC2) is used here as a comparator. IC LM324 consists of four operational amplifiers, of which only two operational amplifiers (N1 and N2) are used in the circuit.

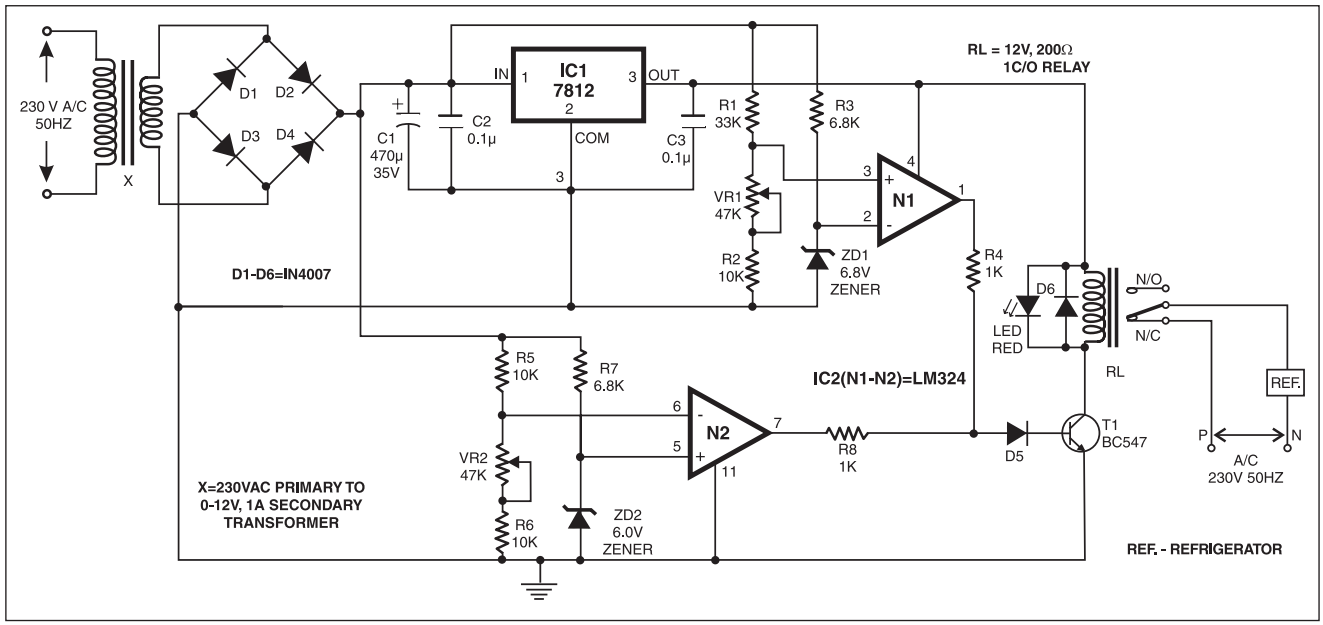
The unregulated power supply is connected to the series combination of resistors R1 and R2 and potmeter VR1. The same supply is also connected to a 6.8V zener diode (ZD1) through resistor R3.

Preset VR1 is adjusted such that for the normal supply of 180V to 240V, the voltage at the non-inverting terminal (pin 3) of operational amplifier N1 is less than 6.8V. Hence the output of the operational amplifier is zero and transistor T1 remains off. The relay, which is connected to the collector of transistor T1, also remains de-energised. As the AC supply to the electrical appliances is given through the normally closed (N/C) terminal of the relay, the supply is not disconnected during normal operation.

When the AC voltage increases beyond 240V, the voltage at the non-inverting terminal (pin 3) of operational amplifier N1 increases. The voltage at the inverting terminal is still 6.8V because of the zener diode. Thus now if the voltage at pin 3 of the operational amplifier is higher than 6.8V, the output of the operational amplifier goes high to drive transistor T1 and hence energise relay RL. Consequently, the AC supply is disconnected and electrical appliances turn off. Thus the appliances are protected against over-voltage.

Now let's consider the under-voltage condition. When the line voltage is below 180V, the voltage at the inverting terminal (pin 6) of operational amplifier N2 is less than the voltage at the non-inverting terminal (6V). Thus the output of operational amplifier N2 goes high and it energises the relay through transistor T1. The AC supply is disconnected and electrical appliances turn off. Thus the appliances are protected against under-voltage. IC1 is wired for a regulated 12V supply.

Thus the relay energises in two conditions: first, if the voltage at pin 3 of IC2 is above 6.8V, and second, if the voltage at pin 6 of IC2 is below 6V. Over-voltage and under-voltage levels can be adjusted using presets VR1 and VR2, respectively.



Readers Comments:

In the 'Over-/Under-Voltage Protection of Electrical Appliances' article:

Q1. 1. How can we introduce a variable power-on time delay?

Q2. What are the wattages of zener diodes ZD1 and ZD2?

Q3. How can we calculate the values to which VR1 and VR2 are to be set? Also give some test-point voltages to troubleshoot the circuit easily.

V. Ramesh
Through e-mail

The author C.H. Vithalani replies:

A1. You can connect the output of opamp N1 to the input of monostable multivibrator (which may be constructed around timer IC 555). This monostable multivibrator will de-energise the relay for preset duration.

A2. Both the zener diodes are of 1/4W.

A3. The values for variable resistors VR1 and VR2 can be calculated as follows:

VR1 = (5604.33–10V_H) / (V_H–130.33) kilo-ohms

VR2 = (2300–10V_L) / (V_L–115) kilo-ohms

where VH and VL are over- and under-voltage, respectively. ❑

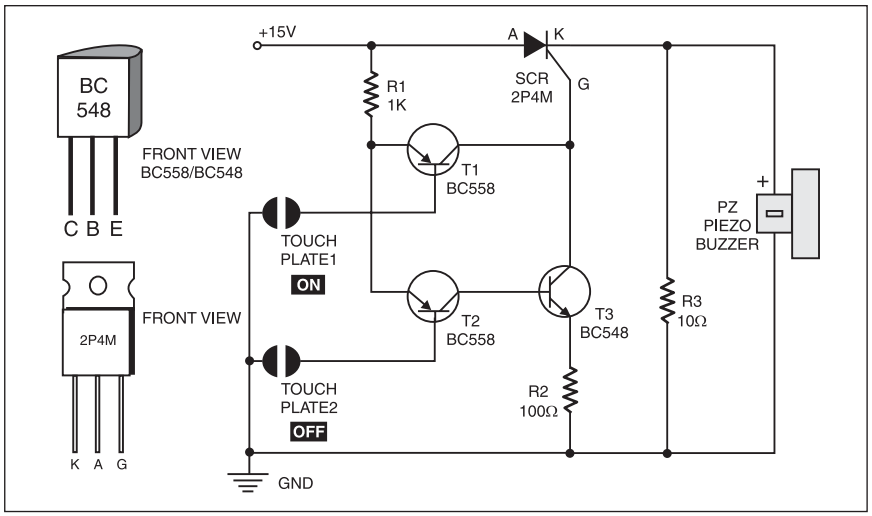
TOUCH BELL

K. RAJESH

This touch bell is built around three transistors and one silicon-controlled rectifier (SCR). The SCR (2P4M) is used to turn on/off the buzzer. Of the three transistors, two are pnp type (BC558) and one is npn type (BC548). The maximum biasing voltage for the SCR is around 15V.

When someone touches plate 1 (which is connected between the base of pnp transistor T1 and ground), transistor T1 gets forward biased and it triggers the SCR by giving a positive voltage to the gate of the SCR. The SCR, in turn, activates the buzzer. The buzzer now sounds continuously.

On the other hand, when someone touches plate 2 (which is connected between the base of pnp transistor T2 and ground), transistor T2 gets forward biased and it provides the necessary voltage for conduction of npn transistor T3. The ground voltage applied to the gate of the SCR turns off the SCR, thus stopping the buzzer.



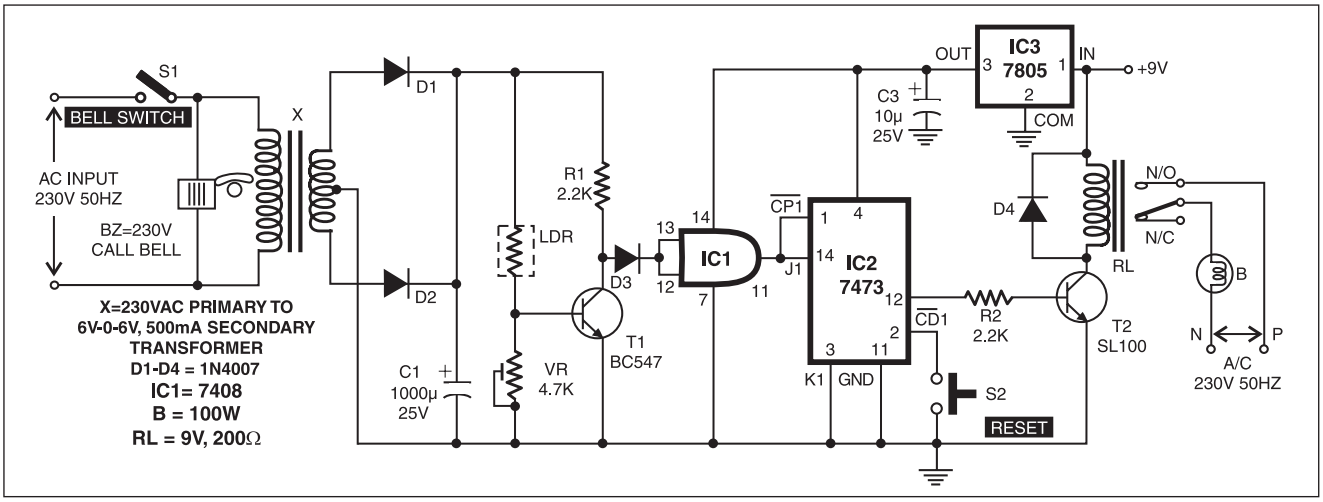
The pin configuration of transistors BC558 and BC548 and SCR 2P4M is shown in the figure. Use a 15V supply to power the circuit.

DOORBELL WITH SECURITY FEATURE

PRAVEEN KUMAR

This doorbell system works such that when someone presses your calling bell switch during the night, not only the bell rings but the bulb connected to it also glows. In order to turn the bulb off, just press the reset pushbutton switch provided in the circuit. Place the bulb near the calling bell switch so that you can see the person pressing the calling bell before opening the door. So you can choose not to open the door to doubtful persons. During day time, the bulb doesn't glow and only the calling bell sounds.

When calling bell switch S1 is closed, the bell rings and simultaneously transformer X gets AC supply. The output of this 6V-0V-6V/500mA step-down transformer is rectified by diodes D1 and D2. The rectified output is filtered by 1000μF, 25V capacitor C1



and fed to the collector of transistor BC547 (T1) via 2.2k resistor R1. A light-dependent resistor (LDR) is connected to the base of this transistor.

During the day, the LDR has a very low resistance as it receives continuous light. So when the calling bell switch is pressed, the transistor conducts and its collector is pulled to ground. Thus the next section of the circuit remains inactive and we hear the calling bell only.

The next section consisting of IC 7408 (IC1) and IC 7473 (IC2) gets a separate supply voltage of 5V from regulator IC 7805 (IC3) as shown in the figure.

During the night, as no light falls on the LDR, it has a very high resistance. So when calling bell switch S1 is pressed, transistor T1 doesn't conduct. As a result, diode D3 is forward biased to make input pins 12 and 13 of IC1 high. Since IC1 is an AND gate, its output at pin 11 will be high. This output is fed to pins 1 and 14 of JK flip-flop IC 7473 (IC2).

For a given high input to the latch made up of IC2, its output pin 12 will be high. Thus transistor SL100 (T2) receives base current and conducts. This energises a 9V, 200-ohm relay (RL) and the 100W bulb connected to the relay glows. The bulb will glow until you press reset pushbutton switch S2.

RESISTANCE MEASUREMENT USING SAMPLE AND HOLD METHOD

SOMNATH CHAKRABARTI

A sample and hold circuit, as the name implies, samples an analogue input signal and holds its value until the input is again sampled.

Fig. 1 shows the basic principle of the sample and hold circuit. The analogue signal V_i is applied to a non-inverting unity-gain amplifier built around an op-amp. This amplifier acts as a buffer. The control terminal of analogue switch S_w decides whether the switch will be closed or open.

When the control terminal is held at logic 1, the switch is closed (on). The time for which the switch remains 'on' is called the sampling period (T_S). During the sampling period, the hold capacitor (C_h) charges up very quickly through the low 'on' resistance of the analogue switch (typically, a few hundred ohms) and follows the analogue voltage at every instant (refer Fig. 2).

When the control terminal is held at logic 0, the switch becomes open (off) and the sampling period ends. The period for which the switch remains 'off' is called the hold period (T_h). The voltage across hold capacitor C_h is fed to a buffer built around another op-amp.

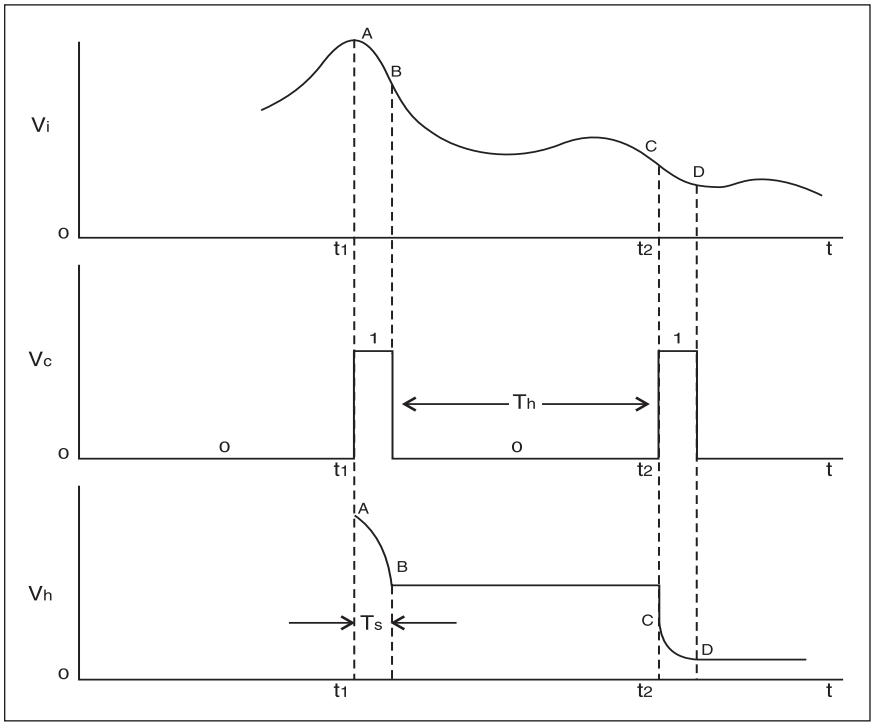


Fig. 2: Waveforms of input signal, control signal, and sample and hold signal

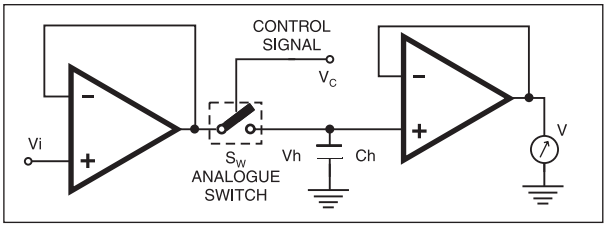


Fig. 1: A typical sample and hold circuit

During period T_h, hold capacitor C_h holds the latest value of the analogue voltage (the value just before the switch is turned off) because it does not find any path to discharge. On the left side it finds the exceedingly large resistance of the 'off' CMOS switch (several hundreds of mega-ohms), and on the right side it finds the high input resistance of the buffer (about 10^5 mega-ohms). So the only path to discharge is its own natural leakage resistance. Using a mylar or teflon capacitor, we can realise a very high leakage resistance in mega-ohms. Thus the voltmeter holds the most recent value of the sampled analogue voltage until we go

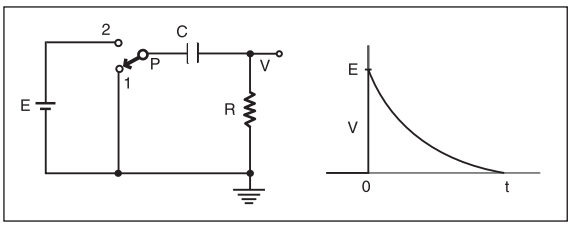


Fig. 3: Capacitor charging circuit and its exponential curve

TABLE

S. No.	Time t, in seconds	Voltage across R in Volt
1	0	4.95
2	5	3.88
3	10	3.06
4	15	2.41
5	20	1.89
6	25	1.49
7	30	1.16
8	35	0.90
9	40	0.69
10	45	0.53
11	50	0.40
12	55	0.30
13	60	0.22
14	65	0.15
15	70	0.10
16	75	0.06
17	80	0.03
18	85	0.01
19	90	0.00

for the next sampling.

The sample and hold circuit can be used to measure a high resistance by charging a capacitor (refer Fig. 3). Initially, the timing capacitor (C) is uncharged. As the pole (P) of the DPDT switch leaves contact 1 and touches contact 2, the output voltage (V) immediately attains the battery voltage (E). (Recall that the voltage across a capacitor cannot be changed instantaneously.) The capacitor now begins to charge up and the voltage V begins to decrease exponentially, viz,

$V = E\, e^{-\tau/\tau}$

where τ is the time constant of the R-C circuit, i.e $\tau = RC$.

Fig. 4 shows the necessary circuit. Here we've used National Semiconductor's IC LF398 (IC2) for the sample and hold circuit. IC NE555 (IC1) is used as an astable multivibrator generating a square wave. Its 'on' time is given by:

$T_{on} = R_A\, C_T \ln 2$

where 'ln' stands for natural log.

This is the previously introduced sampling time (T_s) and is about 0.5 second.

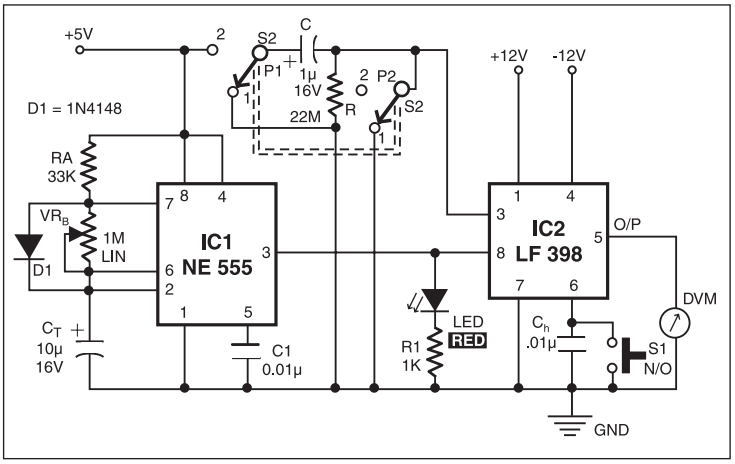


Fig. 4: The sample and hold circuit

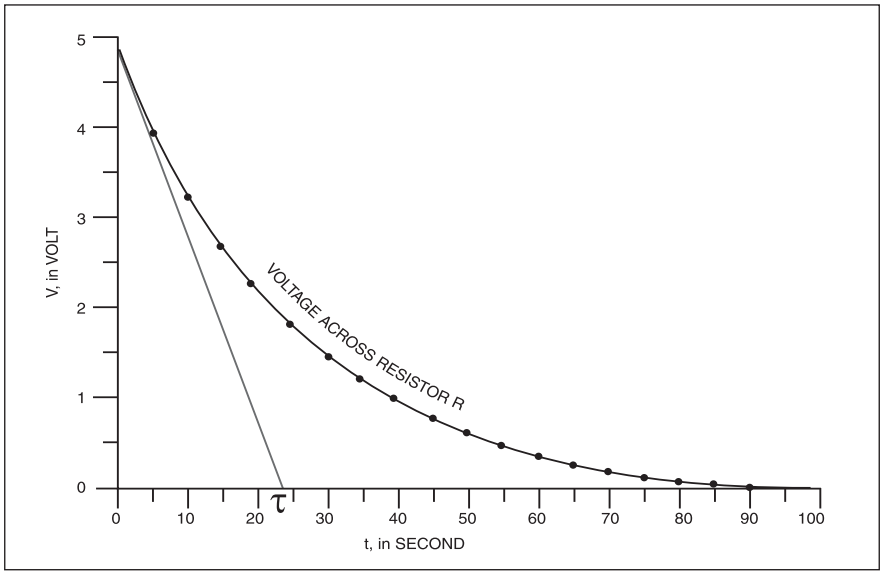


Fig. 5: The voltage measured at different times

The 'off' time of the multivibrator is:

$T_{off} = VR_B\, C_T \ln 2$

This is the hold period (T_h). Using a 1-mega-ohm linear potentiometer, VR_B is adjusted to make T_h close to 5 seconds.

First, momentarily press normally-open (N/O) switch S1 so as to discharge hold capacitor C_h and then release it. Keep DPDT switch S2 in Down position, so both the poles are in touch with their respective contact 1. This makes the capacitor C fully discharged and it is ready for placement in the charging circuit.

Carefully observe the blinking of the LED for at least three cycles. Its flashing rate will give you an idea of the sample and hold period. When the LED flashes up for a split second, toggle the DPDT switch to Up position, so both the poles will touch their respective contact 2.

You can now notice the accurate status of the capacitor. The voltage across resistor R is the analogue voltage you want to measure. Measure this voltage regularly at an interval of 5 seconds until the capacitor is fully charged and the voltage across the resistor drops to zero. With C=1 μF and R=22 mega-ohms, it takes about four time constants, which is about 90 seconds for the capacitor to fully charge up.

Draw the measured voltage values (across R) against their respective times (refer Fig. 5 and Table). The tangent drawn to the curve at the origin (t=0) intersects the time axis at t=22 seconds, i.e. τ=22 seconds. Since C is 1 μF, the measured value of resistor R is close to 22 mega-ohms. This is an excellent agreement.

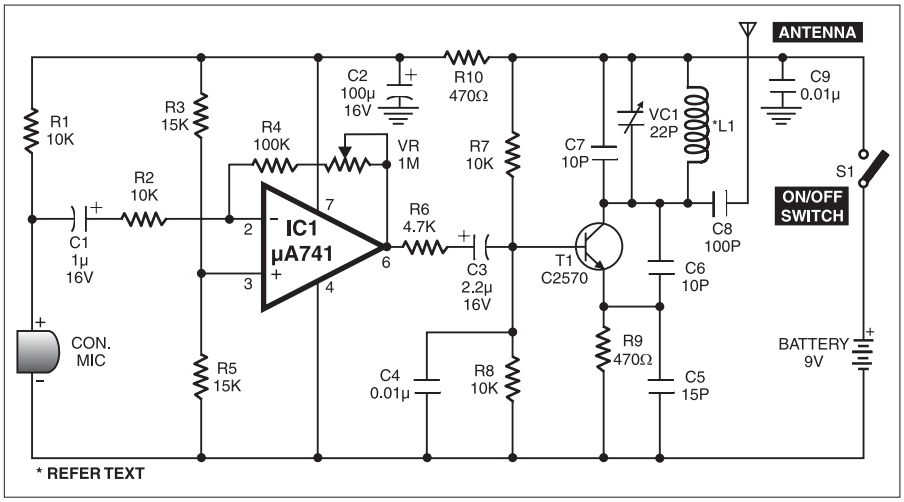
The natural leakage resistance of the mylar capacitor C is very large and doesn't contribute in the measurement.

SENSITIVE FM TRANSMITTER

PRADEEP G.

Here's a highly sensitive FM transmitter whose condenser microphone can receive modulating audio signals from several metres away. It then amplifies the audio signals and transmits the same in the FM frequency range. Thus the circuit can be used as a spy transmitter to overhear the conversations at a remote place using an FM receiver. By using an FM radio with tape recorder, you can record the conversations wirelessly.

The circuit comprises two stages, namely, a high-gain audio preamplifier and a VHF oscillator. The audio preamplifier is based on op-amp IC 741 (IC1). By adjusting 1-mega-ohm preset VR, the gain of op-amp can be varied. The VHF oscillator is wired around RF transistor C2570. The amplified audio signals modulate the RF carrier frequency. The modulated signals are transmitted from the aerial, which is a 75cm long wire.



Coil L1 has four turns of 20 SWG wire on a 4mm diameter air core wound with a slight spacing to a length of 1.5 cm. After construction, adjust the spacing between the coil turns to generate frequency in the range of 88-108 MHz. Also adjust 22pF trimmer VC1 to vary the frequency.

Keep the unit away from FM radio to avoid howling through feedback. House the condenser microphone inside a small tube to increase its audio sensitivity.

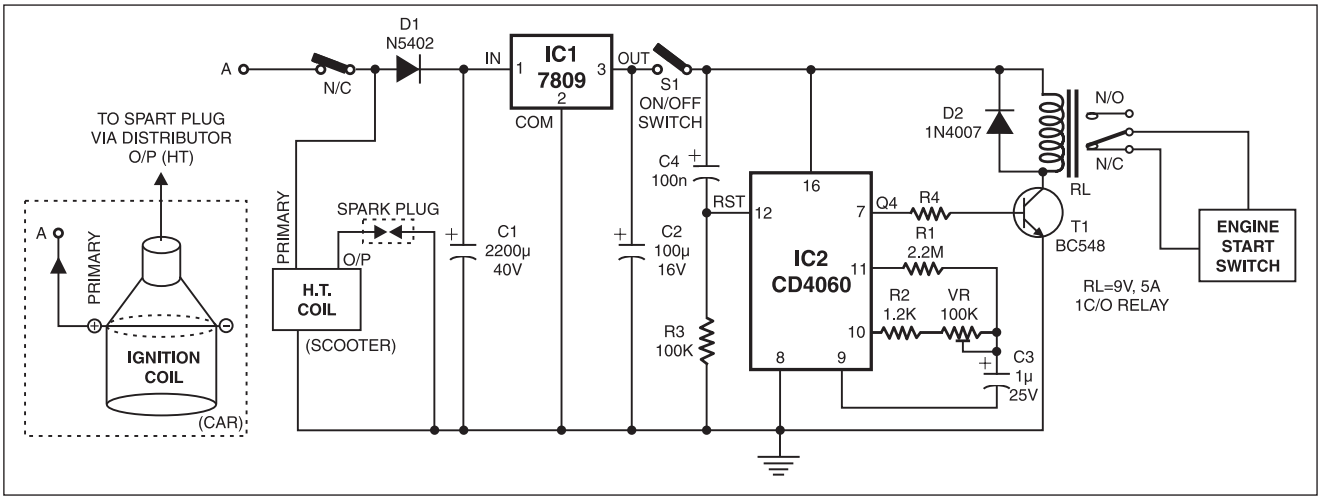
VEHICLE SECURITY SYSTEM

ASHOK K. DOCTOR

This security system guards your vehicle against theft. When the system is switched on, any unauthorised person won't be able to start the vehicle engine, thus foiling the theft attempt.

The circuit is built around IC CD4060 (IC2) that has an in-built oscillator and ten selectable binary outputs. The output Q4 (divided by 2^4) at pin 7 of IC2 is selected to activate the relay. The power-on reset is provided by capacitor C4 and resistor R3. In the absence of external clock, the oscillator frequency is determined by R1, R2, VR, and C3. Adjust 100-kilo-ohm VR to select the time delay.

When the car engine is started, an



AC voltage generated by the alternator (magneto) is rectified by diode D1 and filtered by capacitors C1 and C2. Regulator IC 7809 (IC1) regulates the voltage to 9V DC. IC2 is reset and the timing cycle immediately begins. After the predetermined time period, the output Q4 (pin 7) goes high and transistor T1 (BC548) conducts to energise the relay and switch off the engine.

Every time the engine is started, the same operation takes place to switch off the engine, thus foiling the theft attempt.

On/off switch S1 is an arming switch. You can also use a miniature toggle switch, which can be cleverly concealed in your vehicle. Alternatively, a key-operated switch available at auto component shops can be used. Before you leave the vehicle, close switch S1. When you return, open it before you start the engine.

MOBILE PHONE MULTIPOWER UNIT

HAMZA ANJUMUKKIL

Most of the mobile phone complaints are related to the power supply. This is mainly because of improper charging and use of non-recommended chargers and low-quality batteries. Depending upon the make and model, the charging time of mobile phones varies from 1 hour to 3 hours. The charging current is also different for different models. So it is better to use the charger specified by the company only.

Equipment for measuring the capacity or the backup time of a battery are not readily available in the market. But by measuring the charging and discharging currents, the approximate backup time of a battery can be found out. For example, charge a battery of 4.8V, 400mA rating for 1 minute and check whether it can discharge for 1 minute through a 400mA torch bulb.

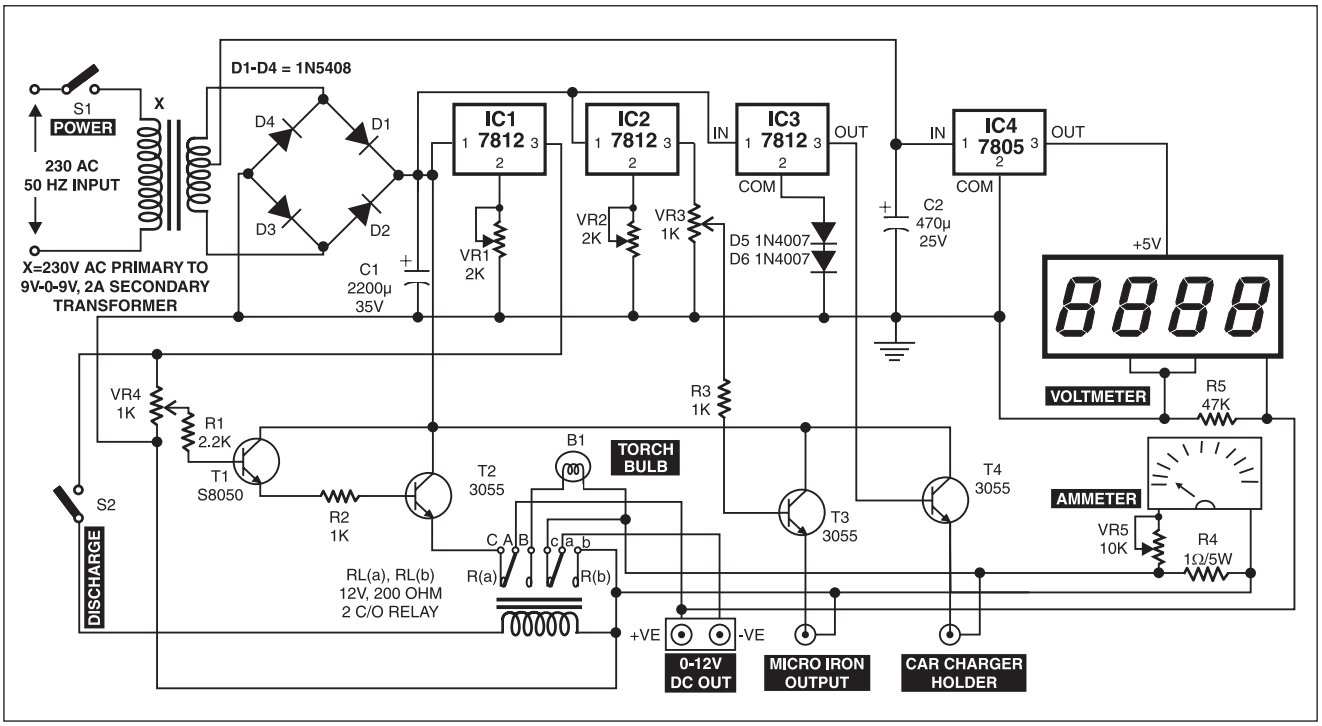
If the battery is discharged fully, it would not get charged again through a normal charger. The battery would require an initial charge or boosting.

Here's a multipurpose circuit (see figure) for battery boosting as well as normal cellphone battery charging. You can boost the battery at 400 mA for two minutes and then charge using a normal charger or this car charger. Other features of this circuit are a variable regulated DC voltage output (0-12V), voltage display panel meter, provision to measure charging/discharging current, ammeter, and micro soldering iron.

The supply voltage for the whole unit is given by a 230V/18V, 2A transformer. This is rectified by a bridge rectifier (1N5408 x 4), filtered, and given to the ICs (IC1, IC2, and IC3) of regulator 7812. Another regulator IC (7805) gives regulated 5V to the voltage display panel meter. The center tapping of the transformer is connected to LM7805. The panel meter displays the variable output voltage (0-12V). It is a 3½-digit LED display module, which is readily available in the market.

Since pin 2 of IC1 (IC 7812) is grounded through a 2-kilo-ohm preset (VR1), it produces an output voltage of 13V (12V+voltage drop across the preset). You can increase the output of IC1 up to 18V by varying the preset.

The output voltage of IC1 is given to transistor T1 (S8050) through 1kilo-ohm



potentiometer VR4 and 2.2-kilo-ohm resistor R1. Potentiometer VR4 acts as the boosting voltage controller. The function of 2.2-kilo-ohm resistor is to limit the boosting current. Transistor T1 acts as the pre-current amplifier.

Power transistor T2 (3055) works as the current amplifier, while 1-kilo-ohm resistor R2 acts as a current limiter to transistor T2. The emitter of T2 is connected to point C of a 12V, 200-ohm relay.

In normal condition, discharge switch S2 is opened and points 'A' and 'a' of the relay are closed to 'C' and 'c', respectively. Hence the boosting out terminals get a supply of 12V maximum. This voltage can be varied from 0 to 12V by using boosting voltage controller VR4. The mobile phone battery is boosted from this variable DC output. The boosting voltage is also given to the digital voltmeter or panel meter for display of the variable DC output.

A volume unit (VU) meter is used for measuring the charging and discharging current. It works from 0.1V to 1V (max.). Within this voltage range, it reads a load current of maximum 1 amp. The maximum current reading can be set with the help of 10-kilo-ohm preset VR5 connected to the VU meter.

The VU meter, boosting terminal, and car charger are connected to ground through 1-ohm, 5W resistor R4. So the VU meter displays the current taken while charging and discharging according to the voltage drop across this resistor.

When discharge switch S2 is switched on, relay RL energises and points 'A' and 'a' come in contact with points 'B' and 'b', respectively. Now if a battery is connected to the boosting terminals, it discharges through the discharge bulb and the V-U meter reads the discharging current.

On adjusting 2-kilo-ohm preset VR2, IC2 (IC 7812) gives an output of 16.5V. This voltage is given to transistor T3 (3055), which works as a current amplifier, through 1-kilo-ohm potentiometer VR3 and 1-kilo-ohm current-limiting resistor R3. Potentiometer VR3 works as the voltage control for micro-tip soldering iron. A standard micro-tip iron needs 16V DC maximum to heat up to 300°C. The micro iron current amplifier drives a micro iron of 1W to 25W.

Regulator IC3 (IC 7812) produces an output of 13V by making use of diodes D5 and D6 connected in series at pin 2 towards ground for dropping 1V. This output is given to power transistor T4 (3055), which works as a current amplifier. An output of 12.5V is obtained at the collector of T4, which is given to the car charger.

The car charger works on DC and it has an inbuilt voltage regulator and current limiter. The input of car charger varies from 4V to 12V. The outputs of different car chargers depend on the make and model. Each charger has its own connector for connection to the mobile phone. The charger holder given here can be used to connect any model of car charger for charging a mobile phone battery.

Normally, mobile phones have a voltage rating of 2.4V to 4.8V. Be careful while connecting a substitute power supply, as even a slight increase in the applied voltage can damage the phone.

Some phones go dead due to a shorted RF power amplifier. If a battery is connected to such a handset, it may suddenly get fully discharged and become dead.

Thus it is advantageous to verify the overall loading of the handset before connecting an external power supply or battery. For the purpose, you can use an ohmmeter. The battery terminal of the handset reads 5 to 50 ohms in one direction and 1 kilo-ohm to 150 kilo-ohms in the other direction. If a wide difference is noted, the circuit is either open or shorted.

Readers Comments:

The 'Mobile Phone Multipower Unit' is very useful to me. I have the following queries regarding this circuit:

1. Can I use a portable TV transformer in place of 9V-0-9V, 2A transformer?

2. To power the circuit from a 12V car battery, what modifications are required?

3. Can you modify the circuit to use such transistor as BD115 in place of large-size power transistor 3055?

4. Can I use it to charge the battery of a cordless phone?

5. As 3½-digit LED display is not available in the local market, please suggest its substitute.

6. Give the lead identification of transistor S8050.

A.S.J. Krishna
Rajahmundry, Andhra Pradesh

The author Hamza Anjumukkil replies:

I am glad to know that my article is useful to Mr Krishna. As regards his queries:

1. One can't use any transformer other than 9-0-9 centre-tapped, as 5V is needed for digital display.

2. The car battery can be used as the power source for the unit. But the 12V car battery cannot provide 14.5V DC required to heat the micro soldering iron to the maximum temperature. Also, the regulator section for the digital display unit needs to be modified. One should provide a very good heat-sink for the 7805 regulator since it drops 12V car battery voltage down to 5V.

3. One has to use transistor 3055 itself for driving the heavy current required during the operation. Hence no substitute is advisable for this transistor.

4. This charging unit can be used to charge any type of rechargable battery used in cordless phones.

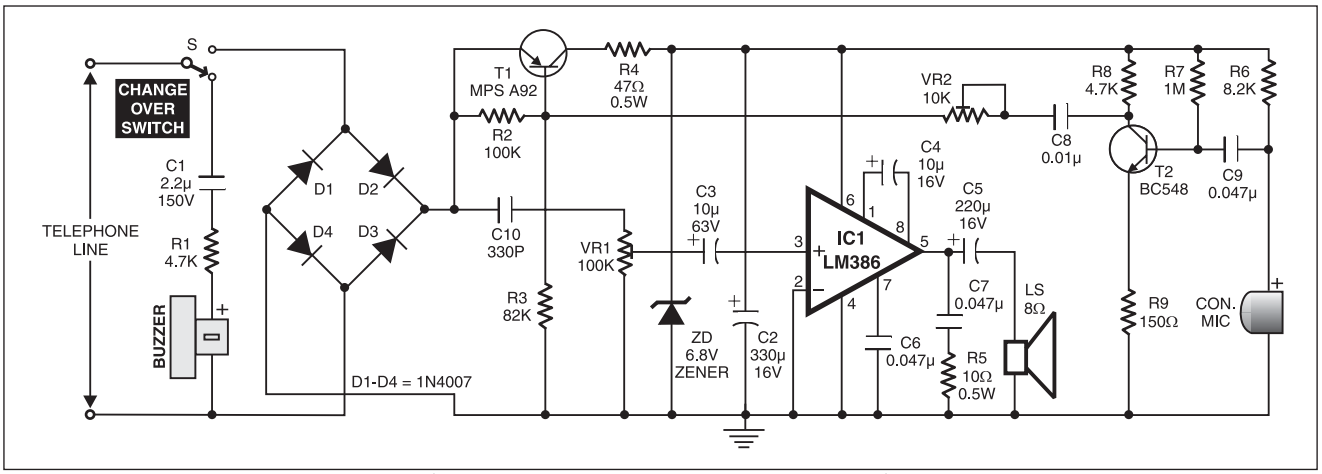
5. The 3½-digit LED display is very much available in the market. You can purchase it from Orbit Electronics, Chunam Lane, Lamington Road, Mumbai. It is also available in the Lajpat Rai Market of New Delhi.

6. The pin configuration of S8050 is the same as that of common BC series transistors, i.e. in the sequence of E, B, and C from left, when viewed from the bottom.

TELEPHONE RECEIVER

S.K. ROUSHON

This simple telephone receiver without a dialling section can be connected in parallel to a telephone line. It can be easily assembled on a small vero board or a PCB. A geometry box made in the shape of a telephone receiver will be an excellent cabinet for it. No external power supply is needed, which makes the circuit handy.



The ringer section comprises R1, C1, and a buzzer. If your telephone has a loud ringer, this circuit can be avoided. A bridge rectifier consisting of diodes D1 through D4 protects the circuit from any polarity change in the telephone line. PNP transistor MPS-A92 (T1) is the main interface transistor. The output of T1 is regulated by zener diode ZD and capacitor C2 to get 6.8V for powering the amplifier section. This power is also used to bias the transmitter section.

The transmitter section comprises transistor BC548 (T2) together with a few discrete components and a condenser microphone. The transmit signal is fed to the base of interface transistor T1. The voice input for the amplifier comes directly from the positive end of the bridge rectifier.

The amplifier section is built around high-performance, low-wattage power amplifier IC LM386. This circuit is designed as a high-gain amplifier. A small 8-ohm speaker is good enough for the output.

After all soldering is done, adjust presets VR1 and VR2 to their middle position and connect the circuit to the telephone line in parallel. Adjust VR1 and VR2 for optimum reception as well as transmission.

Readers' Comments:

Q1. The telephone receiver circuit is a nice, easy-to-construct circuit. But it lacks suppression of side tone, which leads to an annoying high volume. Please publish an add-on circuit for the receiver for reducing the side tone.

Omkar Thakur
Through e-mail

Q2. I want to make the telephone receiver. Before proceeding, I would like to know whether the telephone receiver will connect the call as soon as we change the position of the changeover switch (S), or do we have to take the call using a normal telephone before using the receiver.

I had this doubt because I wondered how the receiver sends the signal to the exchange asking to stop the ringing signal and start the conversation. So clarify whether we need a normal telephone to first change from the ringing signal to the talk mode when a call comes in.

Vineeth Abraham
Thiruvananthapuram

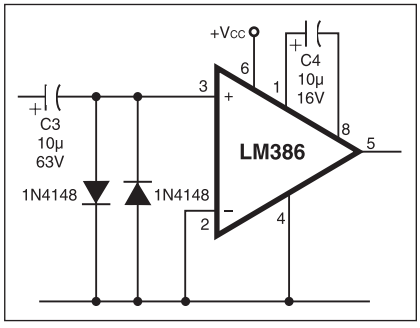


Fig. 1: Add-on circuit for the telephone receiver

The author S.K. Roushon replies:

A1. I thank Mr Thakur for showing interest in my circuit. The circuit is built to offer medium performance. And to keep it simple, I had to compromise on certain aspects. For better performance of the circuit:

1. Replace VR2 by a 100k preset.
2. Use two 1N4148 ICs with reverse polarity in parallel between pin 3 of IC LM386 and the ground (refer Fig. 1 here).
3. In my prototype R3 was 4.7k. You can also try with a 4.7k resistor.

A2. You do not need any other telephone receiver. As soon as you change the position of switch S, it will connect. But remember to switch it back after you finish taking a call, so it is kept ready for the next call.

WASHING MACHINE MOTOR CONTROLLER

SANTHOSH VASUDEVAN

Washing machines usually em ploy a single-phase motor. In semi-automatic washing machines, a purely mechanical switch controls the timing and direction of the motor. These switches are costly and wear out easily.

Here's a controller for single-phase motors of washing machines (Fig. 1) that efficiently replaces its mechanical equivalent. Basically, a single-phase motor requires a master timer, which decides the time for which the motor should keep rotating (washing time), and a spin direction controller, which stops the motor for 3 seconds after every 10 seconds and then resumes rotation in opposite direction.

The direction of rotation can be controlled as shown in Fig. 2. When switch S1 is in position A, coil L1 of the motor receives the current directly, whereas coil L2 receives the current with a phase shift due to capacitor C. So the rotor rotates in clockwise direction (see Fig. 2(a)). When switch S1 is in position B, the reverse happens and the rotor rotates in anti-clockwise direction (see Fig. 2(b)). Thus switch S1 can change the rotation direction.

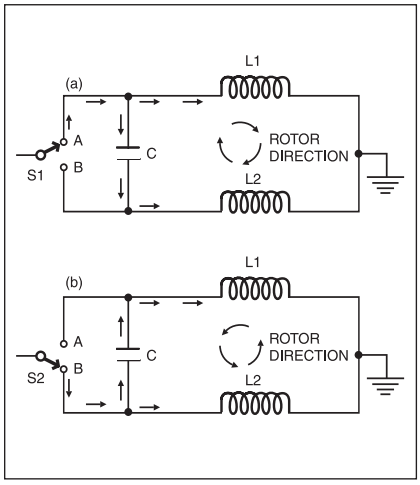


Fig. 2: Direction of motor

The motor cannot be reversed instantly. It needs a brief pause between switching directions, or else it may get damaged. For this purpose, another spin direction control timer (IC2) is employed. It is realised with an IC 555. This timer gives an alternate 'on' and 'off' time duration of 10 seconds and 3 seconds, respectively. So after every l0 seconds of running (either in clockwise or anticlock-wise direction), the motor stops for a brief duration of 3 seconds. The values of R3 and R4 are calculated accordingly.

The master timer is realised with monostable IC 555 (IC1) and its 'on' time is decided by the resistance of 1-mega-ohm potmeter VR. A 47-kilo-ohm resistor is added in series so that even when the VR knob is in zero resistance position, the net series resistance is not zero.

The on-off cycle in the master timer should go on only for the set time (here it is 18 minutes). Once the master timer goes off, the cycle should stop. To achieve this, the outputs of both the timers are connected to NAND gate N1 (IC3), which gives a low output only when both the timers are giving high outputs.The output pin 2 of N1 is connected to relay RL1 via pnp transistor T1, so the relay energises only when the output from NAND gate N1

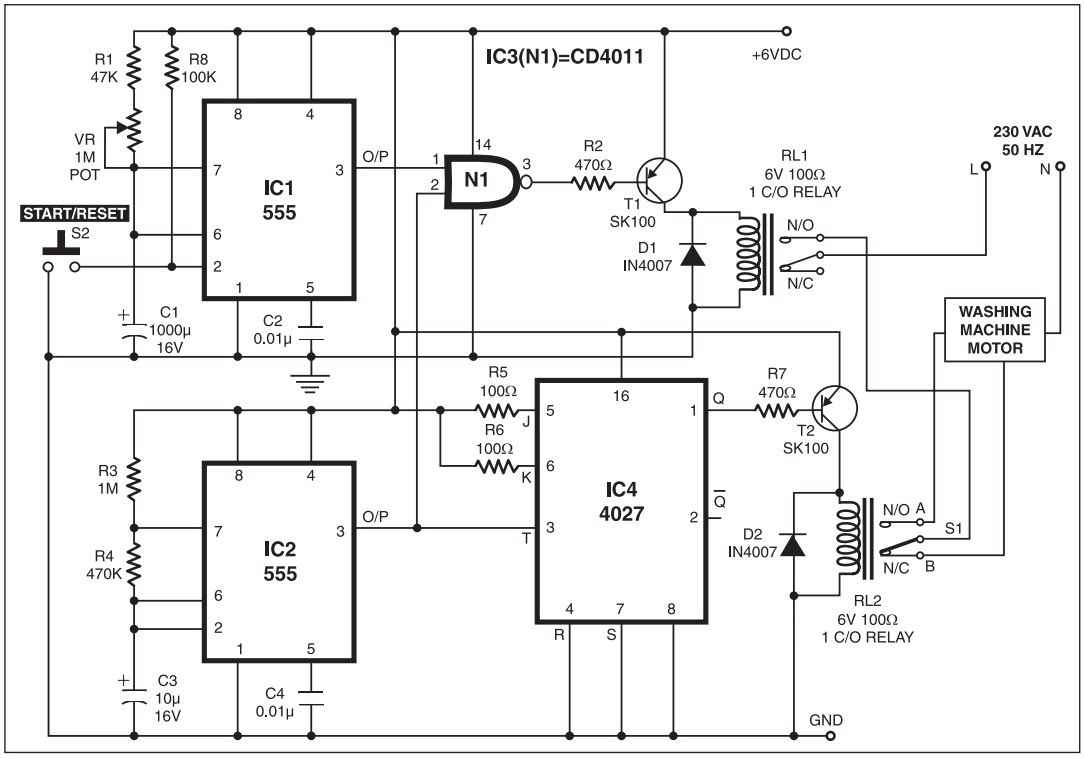


Fig. 1: Circuit diagram of washing machine motor controller

is low. As the mains 220V line is taken through relay RL1, the motor turns off during the 3-second off period after the set time of 10 seconds is over. The graph is shown in Fig. 3.

During 'on' time of spin direction timer IC2, the output of negative-edge triggerd JK flip-flop at pin 2 goes low to energise relay RL2 and washing machine motor rotates in one direction. During the off time of IC2, the output of N1 goes high again to de-energise relay RL1, which cuts off the mains supply to RL2 and the motor stops rotating.

Floating point trouble may occur at trigger pin 2 of IC1. Resistor R8 overcomes this problem by holding pin 2 high.

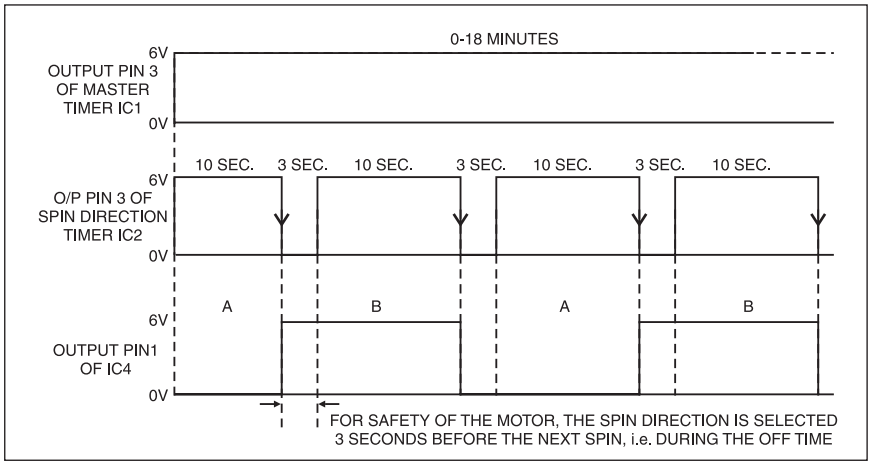


Fig. 3: Timing diagram for rotation of motor

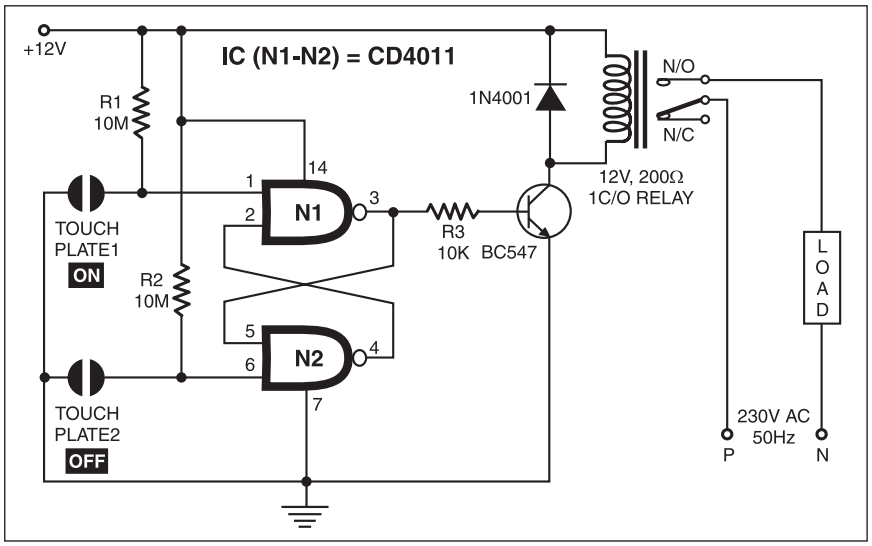
SIMPLE TOUCH-SENSITIVE SWITCH

RAJ K. GORKHALI

This touch-sensitive switch is built around NAND gate IC CD4011 and transistor BC547.

When someone touches plate 1 (which is connected between pin 1 of gate N1 and ground), the RS flip-flop comprising gates N1 and N2 is set. The resulting high output at pin 3 of gate N1 energises the relay via relay driver transistor. The relay, in turn, switches on the load operating on mains.

On the other hand, when someone touches plate 2 (which is connected between pin 6 of gate N2 and ground), the RS flip-flop is reset. The resulting low output at pin 3 of gate N1 de-energises the relay via relay driver transistor. The relay, in turn, switches off the load operating on mains. The diode across the relay coil protects the transistor from back e.m.f induced in the relay when it de-energises.



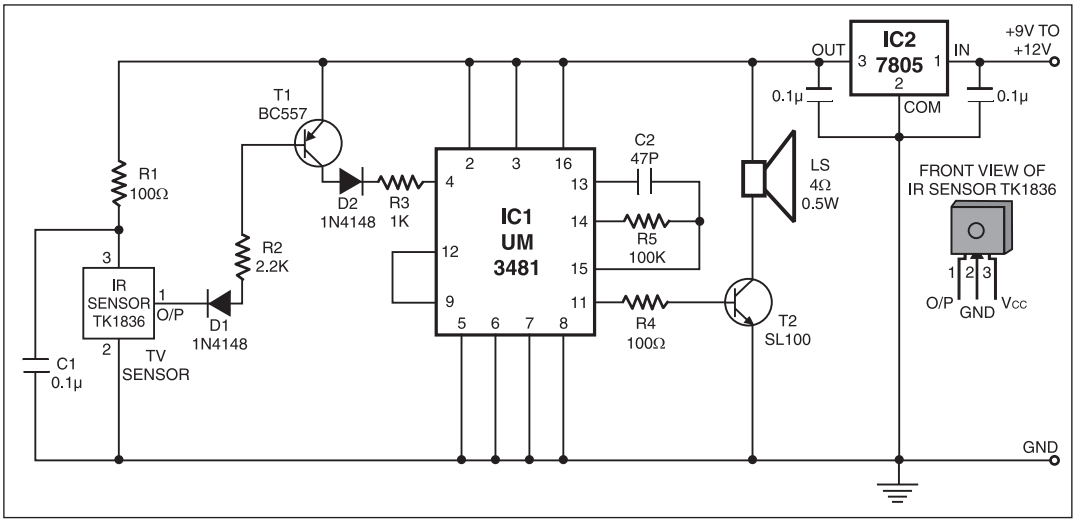
The circuit works off a 12V power supply. If you want to control heavier loads, the current rating of the relay should be increased accordingly.

REMOTE-OPERATED MUSICAL BELL

PRADEEP G.

This infrared light-controlled 12-tone musical bell can be operated using any TV remote control. It can be operated from up to 10 metres, provided the remote control is directed towards the sensor.

The circuit uses the popular 3-lead IR sensor TK1836 to trigger musical bell built around IC UM3481(IC1). (You can also



When any button on the TV remote control is pressed, the sensor's output pulses low. Transistor T1 conducts to apply a triggering pulse to IC1 at its pin 4. After playing one musical tone, the circuit automatically resets. If you again press any of the remote's buttons, another music is heard. This way, twelve different musical tones can be generated.

The circuit works off a 5V power supply. Regulator IC 7805, powered from a 9-12V DC source, provides regulated 5V.

use IC UM3482, UM3483, or UM3484 in place of IC UM3481.) The sensor responds only to 36 kHz. Most TV remote controls transmit this frequency.

SMOKE EXTRACTOR

D. MOHAN KUMAR

Here is a low-cost circuit that auto-matically extracts solder fumes while you assemble a circuit, thereby saving you from inhaling the same. The solder paste gives off toxic fumes, which could pose a potential health hazard if not controlled. Those working closely with electronic components are at a higher risk of fumes being inhaled.

Place this compact unit close to the circuit board and it will extract all the fumes from the immediate working area, leaving smoke-free surroundings. It is fully automatic and turns on a fan when the soldering iron is kept in the holder unit after making a solder joint. The fan automatically turns off after a few seconds. The automatic working of the unit is convenient as the soldering work is intermittent.

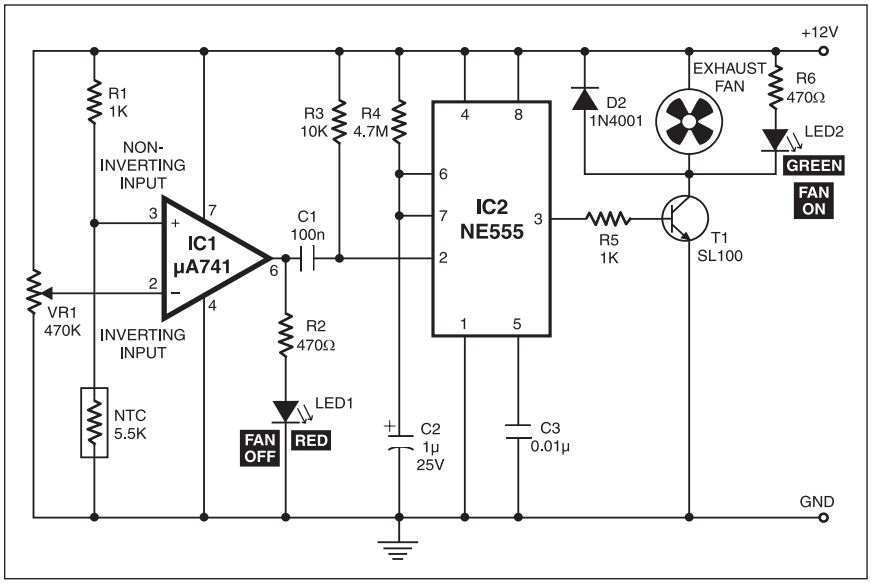


Fig. 1: Circuit of smoke extractor

The circuit (Fig. 1) is designed around popular op-amp µA741 (IC1), which is configured as a comparator. Its non-inverting input (pin 3) is held high by a potential divider comprising resistor R1 and an NTC thermistor; NTC refers to negative temperature coefficient. The inverting input (pin 2) of IC1 is supplied with a reference voltage by preset VR1.

Initially output pin 6 of IC1 is high. Set the voltage at inverting input pin 2 below the voltage at non-inverting input pin 3. When the hot soldering iron is placed close to the thermistor, the resistance of the thermistor falls, dropping the voltage at the non-inverting input of IC1 below the voltage at pin 2. The resulting output at pin 6 goes from high to low and triggers the monostable for a prefixed time (here approximately 30 seconds) to switch on the exhaust fan. This condition is maintained as long as the soldering iron is placed close to the thermistor.

The output of IC1 is fed to trigger pin 2 of monostable IC NE555 (IC2) via capacitor C1. IC2 is designed as a standard monostable and its trigger pin 2 is held high by resistor R3. When the output of IC2 goes low, it cuts off transistor T1 and the fan is switched off.

When the soldering iron is removed from the unit, the thermistor cools and its resistance increases again and subsequently the output of IC1 goes low. This sends a low pulse to the trigger of IC2 and its output goes high. As a result T1 is biased into conduction and the fan is supplied virtually the full 12V supply and it turns on.

Resistor R5 prevents the excessive base current from flowing into the transistor. The timing components of the monostable are resistor R4 and capacitor C2. With the specified values, the 'on' time for the fan is around 30 seconds. Diode D2 protects the transistor from back e.m.f. LED1 and LED2 indicate the exhaust fan's 'off' and 'on' conditions, respectively.

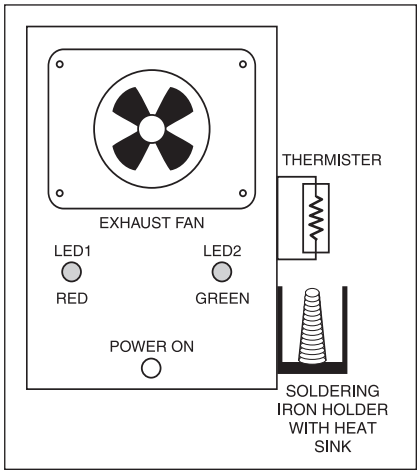


Fig. 2: Suggested arrangement for the metal cabinet (housing the circuit) and heat-sink

The circuit can be constructed on a simple Vero/perforated board. A 12V DC, 500mA mains adaptor is used to power the circuit. House the circuit in a metal case. You can use a small heat-sink (as used in power transistors) to make the set-up for keeping the soldering iron.

Fix the thermistor close to the heat-sink. The thermistor leads should be well insulated with heat-resistant sleeves. Fix the fan in the same fashion as an exhaust fan to pull away fumes from the working area.

After assembling the circuit, adjust VR1 until green LED2 turns off. When you take the hot tip of the soldering iron near the thermistor, green LED2 turns on. The set-up is now ready for use.

AUTO SNOOZE FOR DIGITAL ALARM CLOCKS

S.K. ROUSHON

Alarm clocks are made to wake you up. However, it is quite bothersome to hear the alarm continuously until you get up and put it off. Also, you might fall asleep again. The snooze facility solves this problem to some extent by allowing you to turn the alarm off for a specific period of time by pressing a switch. But here again, you need to look for the snooze switch!

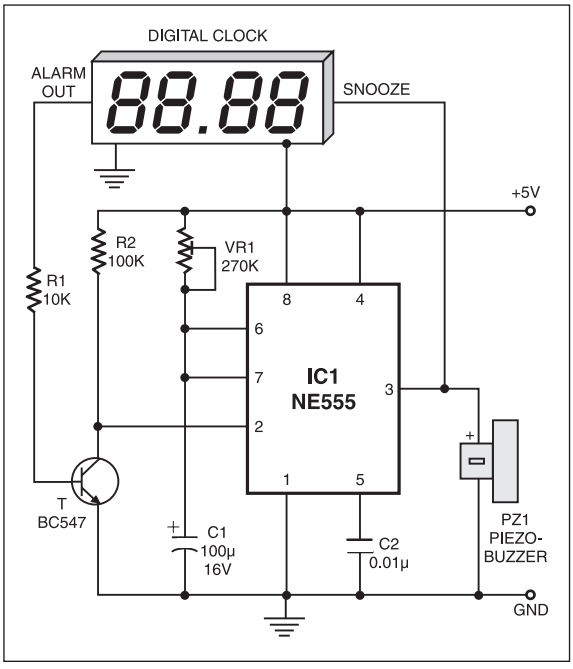


Fig. 1: Auto-snooze circuit for digital alarm clock with snooze facility

Here is a low-cost solution to these problems. This circuit wakes you up gently. It puts off the alarm after a predetermined time and makes it sound again after some time. And this process is automatic. You don't need to press any switch, thanks to timer IC NE555, CMOS NAND gate CD4011, and a few discrete components.

There are two types of auto-snooze circuits: one for a digital alarm clock with snooze facility (Fig. 1) and the other for a digital alarm clock without snooze facility (Fig. 2).

In the auto-snooze circuit for a digital alarm clock with snooze facility, the alarm output as well as the snooze facility of the clock are used. When the alarm's output goes high at the predetermined time, transistor BC547 conducts to trigger pin 2 of IC1. As a result, output pin 3 of IC1 goes high and it makes the alarm sound as well as triggers the snooze. This results in sounding of the alarm for =1.1×VR1×C1 seconds and turning the alarm output off so that pin 2 gets triggered only for a moment. After the snooze time (9 minutes for clock chip MM5387) programmed in the clock IC is over, the alarm output goes high and triggers IC1 again, and this process continues until the alarm-off switch of the clock is pressed.

On the other hand, in the auto-snooze circuit for alarm clocks without snooze facility, first set presets VR2 and VR3, using the formula t=1.1RC, such that the time period of IC3 is less than the time period of IC2. For example, let the time constant of IC3 be 10 seconds and that of IC2 be 7 minutes. In such a setting the snooze time is 6 minutes 50 seconds and alarm-on time is 10 seconds.

In normal situation, the input of

NAND gate N2 is low and thus its output at pin 4 is high. Consequently, the input at pin 2 of N1 is high. On the other hand, the input at pin 1 of N1 is kept low via resistor R3. At the predetermined time, when the alarm output from the clock goes high (also making pin 1 of N1 high), it makes output pin 3 of N1 low, which triggers both the monostables (IC2 and IC3) at pin 2. As a result, the input of gate N2 goes high and hence its output pin 4 as also pin 2 of gate N1 go low. Thus the output of N1 goes high immediately.

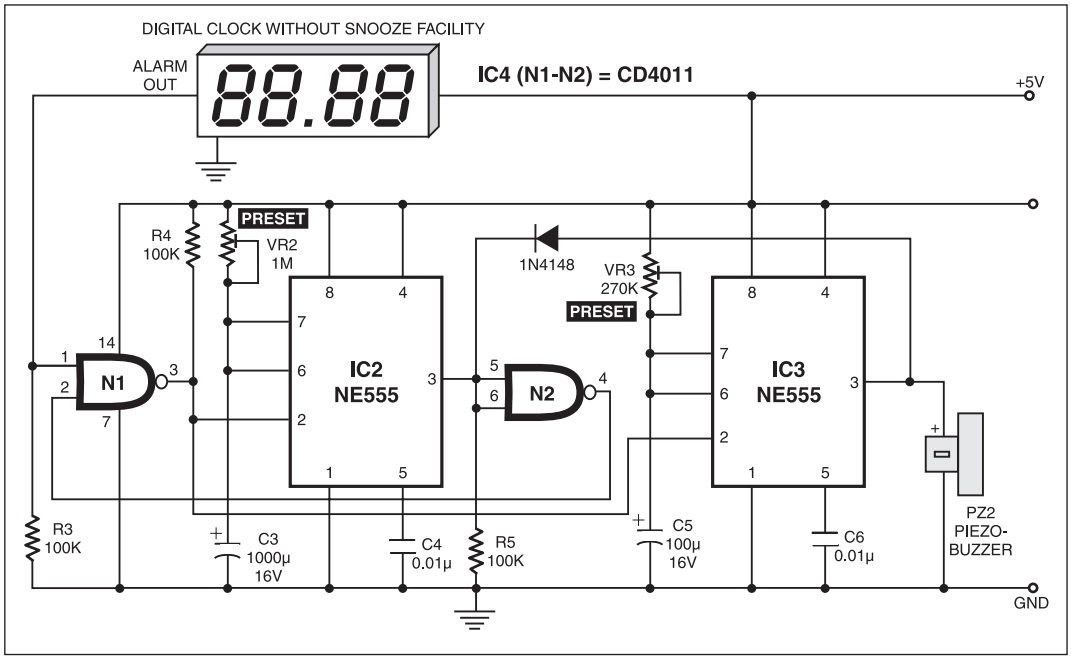


Fig. 2: Auto-snooze circuit for digital alarm clock without snooze facility

The output of IC3 switches on the alarm circuit and it sounds for 10 seconds. The output of IC2 remains high for 7 minutes. Diode 1N4148 prevents the alarm from staying 'on' after 10 seconds. After 7 minutes, the inputs of N2 go low and thus monostable IC2 gets triggered once again. This process continues as long as the alarm output of the clock is high.

The entire circuit can be powered from the 5V DC power supply of the clock. For louder sound you can use any alarm circuit with a suitable power supply in place of the piezobuzzer.

ALARM USING YOUR OWN VOICE

NAGA BABU ARAVA

This alarm plays your prerecorded voice message. It is built around the readily available quartz clock. Take the buzzer out of the quartz clock and connect its positive terminal to pin 1 and negative terminal to pin 2 of optocoupler IC MCT2E (IC2). Pin 4 of IC2 is grounded and pin 5 is connected to trigger pin 2 of monostable multivibrator IC 555 (IC3) as shown in Fig. 2.

Fig. 1 shows the circuit for recording your voice message. When you press switch S2, it plays the meassage. The control circuit shown in Fig. 2 avoids the need for pressing switch S2 and thereby sounding the voice alarm automatically at the preset time.

Connect points A and B of the recording circuit to the corresponding points A and B of the control circuit. After making the connections, press record

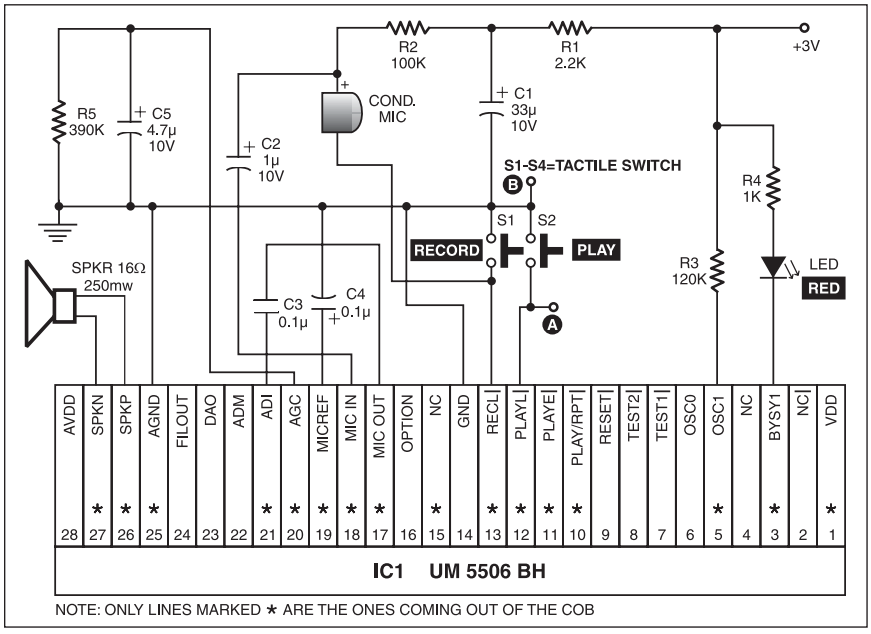


Fig. 1: Voice recording circuit

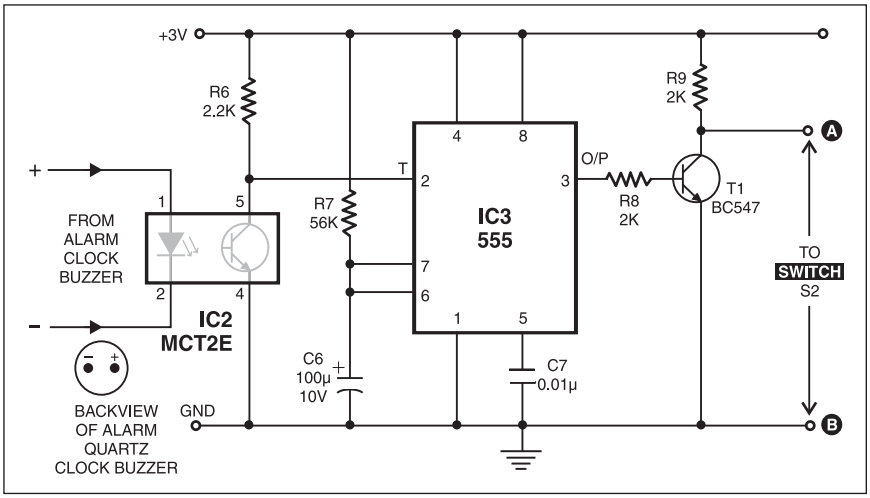


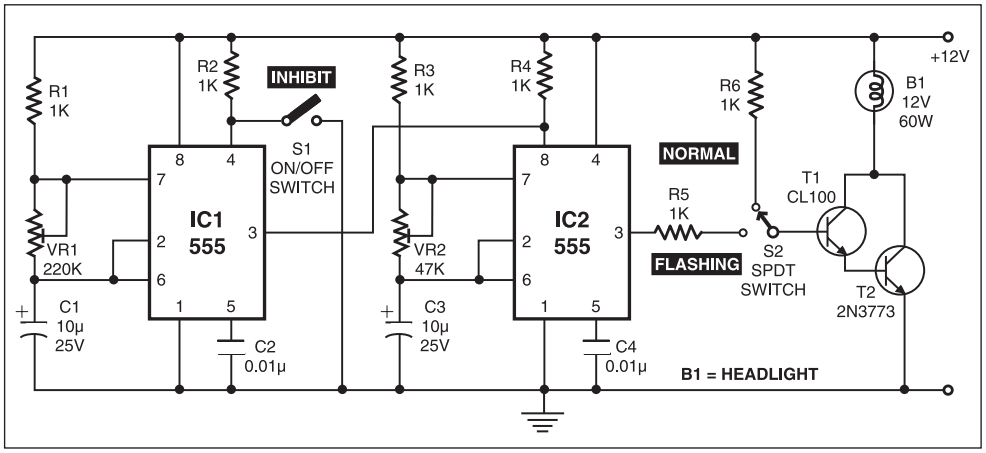
Fig. 2: Voice control circuit

switch S1 to record your 6-second voice message through condenser microphone. Set the desired alarm time in quartz clock. At the time of alarm, buzzer terminals provide voltage to the internal LED of optocoupler IC2. This results in conduction of internal transistor of IC2, and its collector voltage at pin 5 drops to trigger IC3. The output of IC3 goes high for approximately 6 seconds. During this period, the prerecorded message is heard continuously. The message repeats every 6 seconds. The sound is loud enough in a room.

The circuit operates off 3 volts and it can be easily fitted in a small box and fixed on the back side of the alarm quartz clock.

FLASHING HEADLIGHT FOR MOBIKES

PRAVEEN SHANKER



The dipper for a vehicle's headlight changes the light from high beam to low beam or vice versa given a particular set of road conditions. The shifting of light from high to low beam reduces the dazzle for those approaching head-on towards the vehicle. But the dip beam (or low beam) sometimes fails to attract attention of the riders coming from opposite direction. The same is particularly true for blind, steep turns of the roads. A solution for this problem is a flashing bright light that will attract attention immediately.

Here is a simple flashing headlight circuit for motorbikes. It is built around two 555 timer ICs.

The first astable multi-vibrator wired around IC1 introduces a pause interval between the flashes generated by the second multivibrator wired around IC2. The flashes pause for the 'off' duration of IC1, which makes pin 8 grounded to prevent the flashes if switch S2 is towards flashing mode. Pause duration can be adjusted using preset VR1.

The second astable multivibrator works only during the 'on' time of the first astable multivibrator. The flash rate can be varied with the help of preset VR2.

The circuit produces 3 to 6 flashes per second at an interval of 2 seconds. It makes the headlights to flash and then pause, again flash and then pause, and so on.

As headlight bulbs draw a large current, a Darlington pair of transistors T1 and T2 is used to conduct the required current without damaging the transistors. Use heat-sinks with transistors.

Switch S1 is used to inhibit the flashes. With switch S2 you can choose between normal and flashing modes of headlights.

ELECTRONIC MOTOR STARTER

T.A. BABU

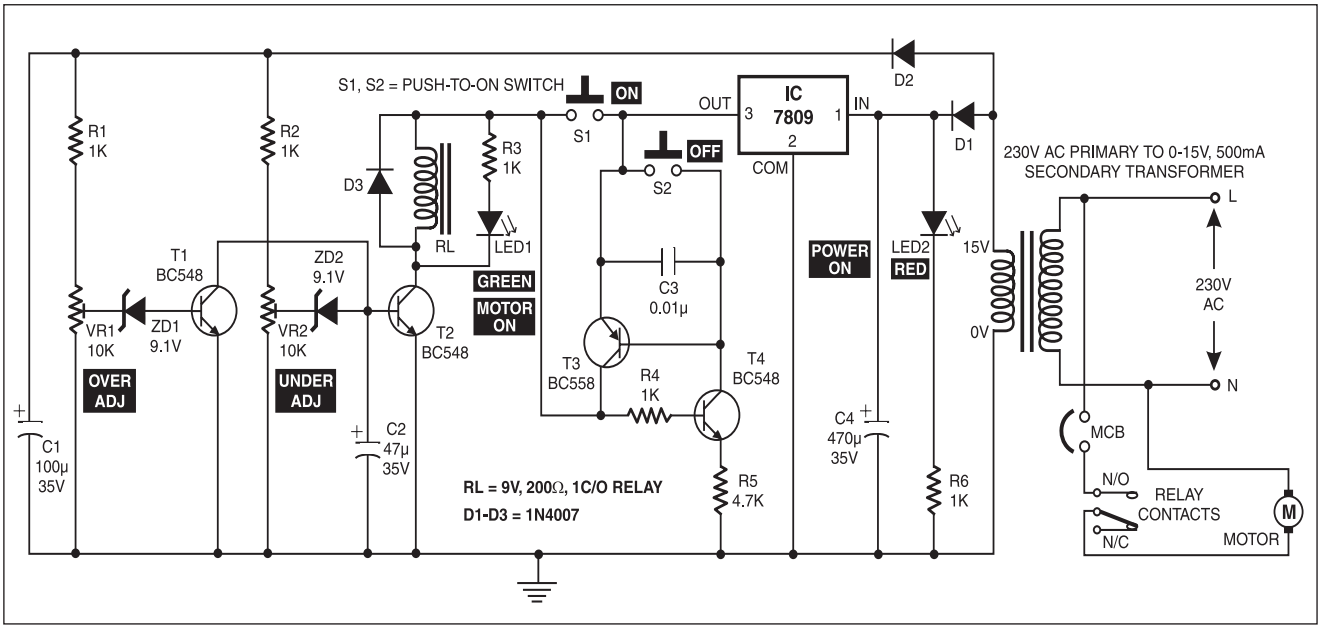
This motor starter protects single-phase motors against voltage fluctuations and overloading. Its salient feature is a soft on/off electronic switch for easy operation.

The transformer steps down the AC voltage from 230V to 15V. Diodes D1 and D2 rectify the AC voltage to DC. The unregulated power supply is given to the protection circuit.

In the protection circuit, transistor T1 is used to protect the motor from over-voltage. The over-voltage setting is done using preset VR1 such that T1 conducts when voltages goes beyond upper limit (say, 260V). When T1 conducts, it switches off T2. Transistor T2 works as the under-voltage protector. The under-voltage setting is done with the help of preset VR2 such that T2 stops conducting when voltage is below lower limit (say, 180V). Zener diodes ZD1 and ZD2 provide base bias to transistors T1 and T2, respectively. Transistors T3 and T4 are connected back to back to form an SCR configuration, which behaves as an 'on'/'off' control. Switch S1 is used to turn on the pump, while switch S2 is used to turn off the pump.

While making over-/under-voltage setting, disconnect C2 temporarily. Capacitor C2 prevents relay chattering due to rapid voltage fluctuations.

Regulator IC 7809 gives the 9V regulated supply to soft switch as well as the relay after filtering by capacitor C4. A suitable miniature circuit breaker is used for automatic over-current protection. Green LED (LED1) indicates that the motor is 'on' and red LED (LED2) indicates that the power is 'on'. The motor is connected to the normally-open contact of the relay. When the relay energises, the motor turns on.



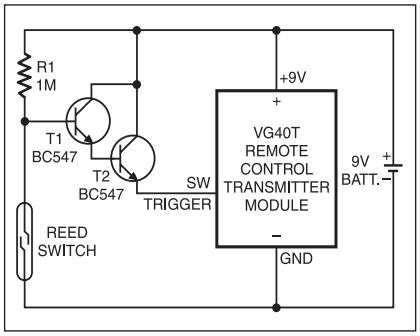
WIRELESS DOOR OPENING ALARM

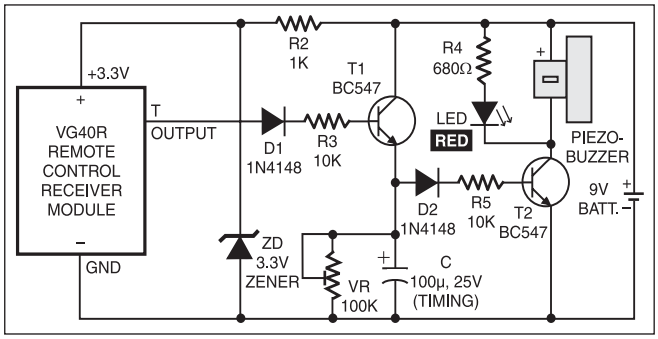
PRADEEP G.

This door opening alarm does not re-quire any wire connection be tween the door unit and the alarm unit. It uses compact VG40T transmitter and VG40R receiver modules, each operating on a very high frequency of 300-315 MHz.

The transmitter unit, which is to be mounted on the door, is shown in Fig. 1. The VG40T transmitter module has three leads. The positive lead is connected to the positive terminal of the battery and the negative lead is connected to the negative terminal of the battery. The central SW lead is connected to the emitter of transistor T2 for triggering the transmitter.

A simple magnet-operated, 2-lead reed switch is used as the sensor. The reed switch is fixed on the door frame using an adhesive such as Feviquick. The magnet (not shown in Fig. 1) is fitted on the moving part of the door such that when the door is closed, the magnet comes close to





reed switch and thus closes the reed switch terminals.

When the door is opened, the magnet moves away from reed switch and the reed switch terminals open. As a result, Darlington pair transistors T1 and T2 conduct and apply a positive voltage to trigger the VG40T transmitter module. The transmitter module then sends a modulated signal to the receiver module.

Fig. 2 shows the alarm unit. The compact receiver module, shown in the figure, also has three leads (positive, negative, and output T). Across the positive and negative leads, a 3.3V zener diode is connected. The output lead is connected to the anode of diode D1, which is the input of the alarm unit.

When the receiver module receives the transmitted signal, its output goes high for a moment and the npn transistor connected at its output terminal conducts to charge the electrolytic capacitor. Then piezobuzzer sounds and the LED glows for a few seconds depending upon the value of the timing capacitor. After that, the piezobuzzer automatically resets. By changing the value of the timing capacitor and adjusting preset VR, you can vary the alarm duration.

The transmitter unit and the alarm unit can communicate wirelessly up to 15 metres. These are powered from separate 9V batteries.

NON-CONTACT POWER MONITOR

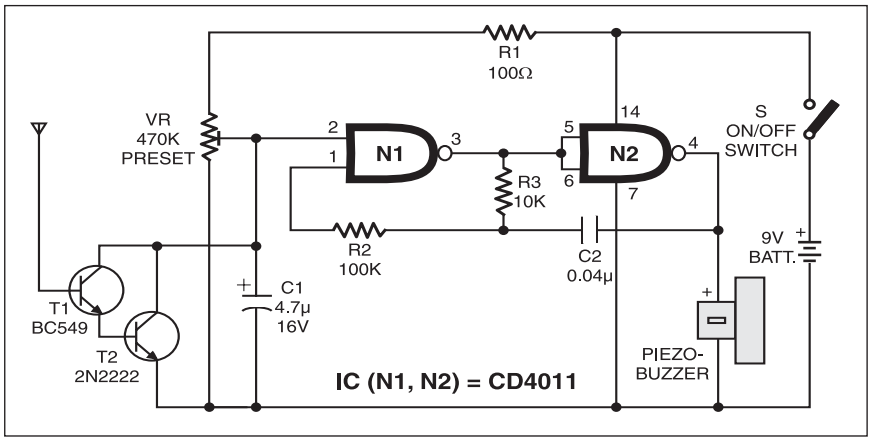
D. MOHAN KUMAR

Here is a simple non-contact AC power monitor for home appliances and laboratory equipment that should remain continuously switched-on. A fuse failure or power breakdown in the equipment going unnoticed may cause irreparable loss. The monitor sounds an alarm on detecting power failure to the equipment.

The circuit is built around CMOS IC CD4011 utilising only a few components. NAND gates N1 and N2 of the IC are wired as an oscillator that drives a piezobuzzer directly. Resistors R2 and R3 and capacitor C2 are the oscillator components. The amplifier comprising transistors T1 and T2 disables the oscillator when mains power is available.

In the standby mode, the base of T1 picks up 50Hz mains hum during the positive half cycles of AC and T1 conducts. This provides base current to T2 and it also conducts, pulling the collector to ground potential.

As the collectors of T1 and T2 are connected to pin 2 of NAND gate N1 of the oscillator, the oscillator gets disabled when the transistors conduct. Capacitor C1 prevents rise of the collector voltage of T2 again during the negative half cycles.



When the power fails, the electrical field around the equipment's wiring ceases and T1 and T2 turn off. Capacitor C1 starts charging via R1 and preset VR and when it gets sufficiently charged, the oscillator is enabled and the piezobuzzer produces a shrill tone. Resistor R1 protects T2 from short circuit if VR is adjusted to zero resistance.

The circuit can be easily assembled on a perforated/breadboard. Use a small plastic case to enclose the circuit and a telescopic antenna as aerial. A 9V battery can be used to power the circuit. Since the circuit draws only a few microamperes current in the standby mode, the battery will last several months. After assembling the circuit, take the aerial near the mains cable and adjust VR until the alarm stops to indicate the standby mode. The circuit can be placed on the equipment to be monitored close to the mains cable.

MOSFET-BASED PREAMPLIFIER FOR FM RADIO DXing

N.S. HARISANKAR,VU3NSH

FM transmissions can be received within a range of 40 km. If you are in fringe areas, you may get a very weak signal. FM DXing refers to hearing distant stations (1500 km or more) on the FM band (88-108 MHz). The term 'DX' is borrowed from amateur radio operators. It means 'distance unknown'; 'D' stands for 'distance' and 'X' stands for 'unknown.' For an FM receiver lacking gain, or having a poor signal-to-noise ratio, using an external preamplifier improves the signal level.

The dual-gate MOSFET preamplifier circuit shown in Fig. 1 gives an excellent gain of about 18 dB. It costs less and is simple to design.

Field-effect transistors (FETs) are superior to bipolar transistors in many applications as these have a much higher gain—approaching that of a vacuum tube. These are classified into junction FETs and MOSFETs. On comparing the FETs with a vacuum tube, the gate implies the grid, the source implies the cathode, and the drain implies the plate. In a transistor, the base implies the grid, the emitter implies the source, and the collector implies the drain.

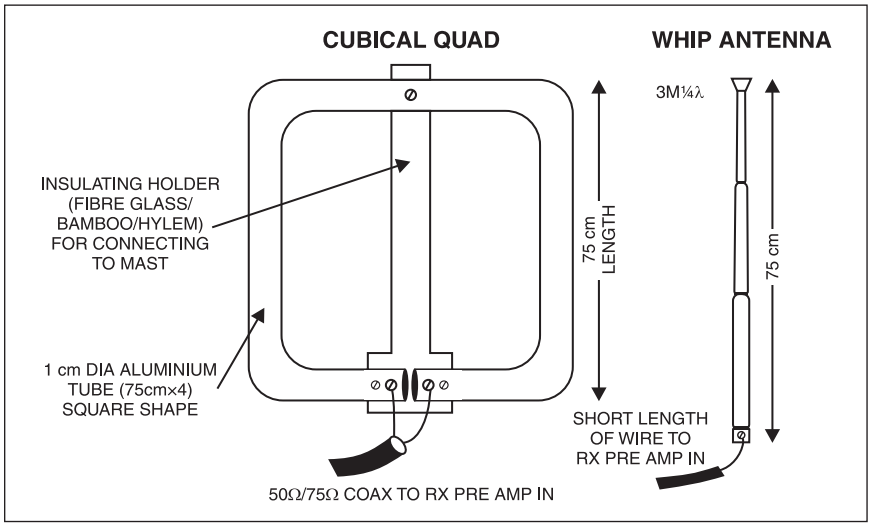


Fig. 2: Different antennae used for FM DXing

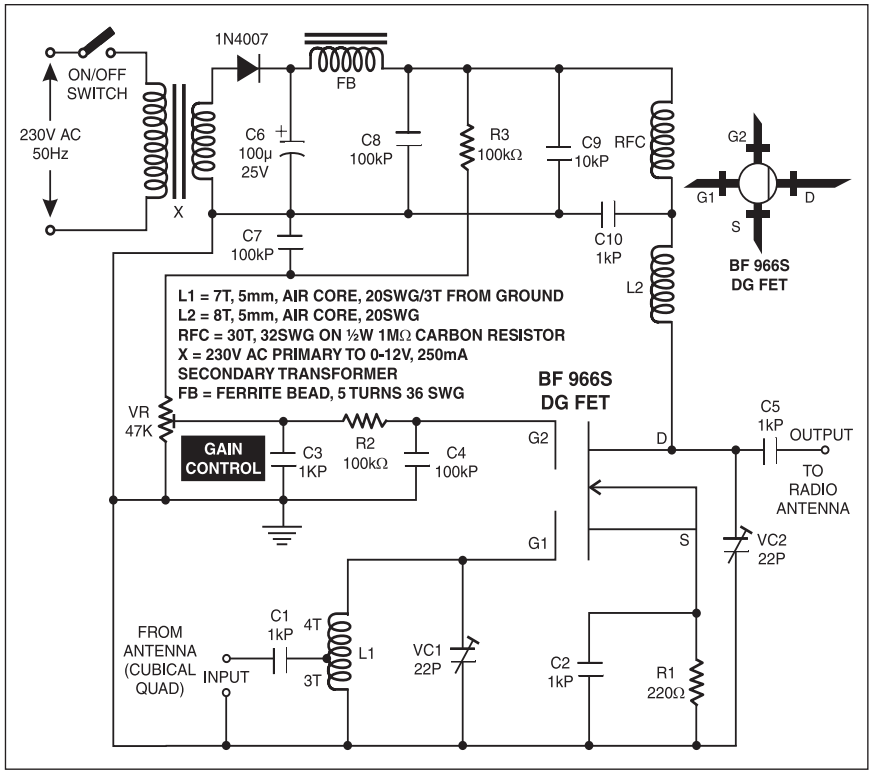


Fig. 1: Circuit of MOSFET-based preamplifier for FM DXing

In dual-gate FETs, gate 1 is the signal gate and gate 2 is the control gate. The gates are effectively in series, making it easy to control the dynamic range of the device by varying the bias on gate 2.

The MOSFET is more flexible because it can be controlled by a positive or negative voltage at gate 2. The resistance between the gate and rest of the device is extremely high because these are separated by a thin dielectric layer. Thus the MOSFET has an extremely high input impedance.

Dual-gate MOSFETs (DG MOSFETs) are very popular among radio amateurs. These are being used in IF amplifiers, mixers, and preamplifiers in HF-VHF transceivers. The isolation between the gates (G1 and G2) is relatively high in mixer applications. This reduces oscillator pulling and radiation. The oscillator pulling is troublesome particularly in shortwave communications. It is a characteristic in many unsophisticated frequency-changer stages, where the incoming signal, if large, pulls the oscillator frequency

Coil & Capacitor Details for DXing of Various Frequency Bands

	10m band (28 MHz) amateur radio	6m band (50 MHz) amateur radio	3m band (98 MHz) FM radio	2m band (144 MHz) FM radio
L1 core	17T, 28 SWG Amindoncore (T-50-6), Tap at 6T from GND	12T, 26 SWG OnT-37-10,Tap at 5T from GND	7T, 5mm dia., 20SWG Aircore, 1cm Length, Tap 3T from GND	5T, 20SWG, ½-inch ID, ½-inch L Tap 2 from GND
L2 core	17T, 28 SWG Amindoncore T-50-6, Without tap	12T, 26 SWG On T-37-10, without tap	8T, close winding, 5mm dia. Air-core, No tap	4T, 20 SWG, No tap
VC1 & VC2	60 pF	22 pF	22 pF	15 pF

slightly off the frequency set by the tuning knob and towards a frequency favourable to the (large) incoming signal. A DG MOSFET can also be used for automatic gain control in RF amplifiers.

DG MOSFET BF966S is an n-channel depletion-type MOSFET that is used for general-purpose FM and VHF applications. In this configuration, it is used for FM radio band. The quadratic input characteristic of the FET input stage gives better results than the exponential characteristic of a bipolar transistor.

Gate 1 is meant for input and gate 2 is for gain control. The input from the antenna is fed to gate G1 via C1 and L1. Trimmer VC1 is used to tune and select the input frequencies. Capacitor C4 (100 kpF) at the gain control electrode (gate 2) decouples any variation in G2 voltage at radio frequencies to maintain constant gain. Set preset VR (47k) to adjust the gain or connect a fixed resistor for fixed gain. The output of the circuit is obtained via capacitor C5 and fed to the FM receiver amplifier.

For indoor use, connect a ¼-wavelength whip antenna, ½-wavelength 1.5m wire antenna, or any other indoor antenna set-up with this circuit. You may use a 9V battery without the transformer and diode 1N4007, or any 6V-12V power supply to power the circuit (refer Fig. 1). The RF output can be taken directly through capacitor C5. For an improved input and output impedance, change C1 from 1 kpF to 22 pF and C5 from 1 kpF to 100 kpF.

For outdoor use at top mast, like a TV booster, connect the C5 output to the power supply unit (PSU) line. Use RG58U/RG11 or RG174 cable for feeding the power supply to the receiver amplifier. The PSU for the circuit is the same as that of a TV booster. For TV boosters, two types of mountings are employed: The fixed tuned booster is mounted on the mast of the antenna. The tunable booster consisting of the PSU is placed near the TV set for gain control of various TV channels. (For details, refer 'High-Gain 4-Stage TV Booster' on page 72 of *Electronics Projects Vol. 8*.)

Mount the DG MOSFET BF966S at the solder side of the PCB to keep parasitic capacitance as small as possible. Use an epoxy PCB. After soldering, clean the PCB with isopropyl alcohol. Use a suitable enclosure for the circuit. All component leads must be small. Avoid shambled wiring to prevent poor gain or self oscillations. Connecting a single-element cubical quad antenna to the circuit results in 'Open Sesam' for DXing.

You can use a folded dipole or any other antenna. However, an excellent performance is obtained with a cubical quad antenna (refer Fig. 2) and Sangean ATS-803 world-band receiver.

In an amplifier, FET is immune to strong signal overloading. It produces less cross-modulation than a conventional transistor having negative temperature coefficient, doesn't succumb to thermal runaway at high frequencies, and decreases noise. In VHF and UHF, the MOSFET produces less noise and is comparable with JFETs. DG FETs reduce the feedback capacitance as well as the noise power coupled to the gate from the channel, giving stable unneutralised power gain for wide-band applications.

This circuit can be used for other frequency bands by changing the input and the output LC networks. The table here gives details of the network components for DXing of stations at various frequency bands.

Readers' Comments:

In the 'MOSFET-based Preamplifier for FM DXing' circuit, I request the author to clarify the following points:

Q1. Is MOSFET BF966 internally protected from statics? Please suggest a low-cost alternative for the same.

Q2. When I used a dipole rooftop antenna for FM radio, an unknown channel overlapped the entire FM band, even the local broadcast band. How can I cure this problem?

Q3. Can I listen to the broadcasts from FM stations located 500 km or more away by simply adding this circuit?

Q4. For my 28MHz project, can I use MC1350P IC for the front-end preamplifier. What is the sensitivity of this IC?

Q5. What is the power supply unit (PSU) and how it is to be connected for outdoor use at top-mast?

Q6. Where can I get Segean ATS-803 world-band receiver?

Himanshu Kulasrestha
Nagpur

The author N.S. Harisankar, VU3NSH, replies:

A1. The input gates of BF966 FET are protected by a back-to-back zener arrangement. More data is available at www.vishay.com. The cost of the FET varies from supplier to supplier. On an average, it is Rs 12 to Rs 20. In place of BF966, you can use 956.

A2. Radio DXing depends on a number of factors including set-up, location and polarisation of your antenna; sensitivity, selectivity and the input RF termination of your receiver; direct or reflected signal; and weather conditions. The interference throughout the band is due to cable TV leakage or any other strong transmitter nearby your location.

A3. The DXing distance cannot be predicted as 500 km or 3000 km. It depends on many factors. You may refer to wave propagation and allied topics at the Website 'www.arrl.org.' One of my friends at Ambasamudram is receiving FM Radio Vijayawada very clearly and all FM transmissions

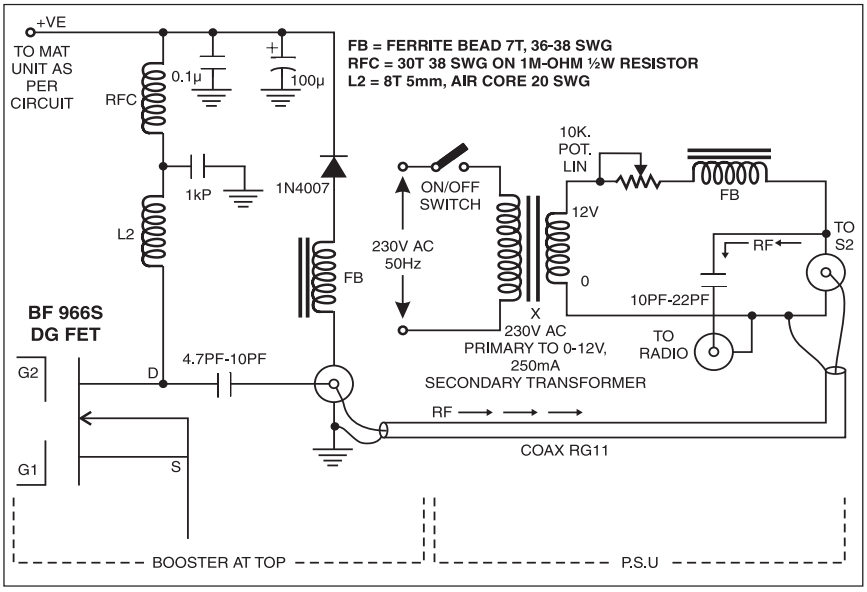


Fig.1: Power supply unit for preamplifier for FM DXing

from Colombo with this 3-element cubical quad antenna.

A4. The MC1350 from Motorola can be used below 70 MHz. It is meant for video, IF, TV and radio reception.

A5. The power supply unit is shown in the Fig.1. The values of capacitors in power supply and RF amplifier sections (decoupling capacitor to radio and coupling capacitor from RF amplifier to cable) vary depending upon the radio's input impedance. So select different values or use trimmers.

The power supply circuit in the figure is meant only for the RF preamplifier stage and not for the FM receiver. Regarding rectifiers and filters, diode 1N4007 and 100μF ferrite bead in the RF amplifier constitute the rectifier system for providing DC supply to the circuit (usually found in the TV booster system). The S2 connector at the power supply is connected to the preamplifier stage via a sufficient length of coaxial cable. The cable gives low AC voltage to the booster stage. The boosted RF signal from the preamplifier stage is fed to S2 point and the RF signal couples through 10-22pF capacitor for the RF receiver input.

A6. For Segean worldband receiver, Mr Kulasrestha may contact the HAM radio operator or shortwave listeners' club in his locality. He can also find the related information on Websites 'www.universal-radio.com' or 'www.hamradio.com.' The receivers are professional grade type with a high gain.

ELECTROLYTIC CAPACITOR TESTER

P. VENKATA RATNAM

Using this electrolytic capacitor tester you can detect leaky and dead (open) electrolytic capacitors. It operates based on the time constant (T) of the capacitor when it is charged up to 63 per cent of the applied voltage via a known resistor. The time constant is calulated as follows:

$T = C \times R$

where 'T' is in seconds, 'C' is in microfarads, and 'R' is in mega-ohms.

Two NE555 timer ICs are used. IC1 is wired in the monostable mode. Initially, when the power is applied, the low output of IC1 causes LED1 to glow. When IC1 is triggered by pressing switch S3, the capacitor under test starts charging via the selected resistor (R1, R2, R3, or R4) and its output jumps to high state, causing LED1 to go off. It remains high for a time duration (in seconds) depending on the RC time constant and then returns to the original low state, which causes LED1 to glow again.

The monostable time period (=1.1×R×C) can be measured by a stop-watch. By comparing this time period (delay time) with that of a good capacitor, we can find the value of the capacitor.

IC2 is connected in the astable mode. Two red LEDs (LED2 and LED3) are connected to its output pin 3. When the output of IC1 jumps to high state, LED1 goes off and the power is applied to pins 4 and 8 of IC2, causing LED2 and LED3 (connected to IC2) to start flashing. Using VR1 adjust the flashing rate of LED2 and LED3 to one flash per second. After the monostable time period is over, LED2 and LED3 stop flashing and LED1 glows again. The number of flashes counted is the time period in seconds.

TABLE I

Resistance range	Arbitrary factor
10 mega-ohms	0.09
1 mega-ohm	0.9
100 kilo-ohms	9
10 kilo-ohms	90

Connect the capacitor under test at the indicated position with polarity as shown in the figure. Close switch S1 to apply power to the tester. LED1 glows immediately to indicate that power is applied to the tester. Set selection switch S2 to low-resistance range position. On pressing switch S3, LED1 goes off and LED2 and LED3 start flashing. Count the flashes carefully until the LED stops flashing.

Now connect a good capacitor of the same value to the tester and note its delay

period. If the delay period of the capacitor under test is almost equal to that of the good capacitor, it is in good condition. In case LED2 and LED3 flash indefinitely without stopping or no flashing, the capacitor under test is leaking or dead short.

To calculate the approximate value of the capacitor under test, multiply the delay time by an arbitrary factor. The arbitrary factor is different for different resistance ranges (refer Table I).

Example 1: For a 10μF capacitor, delay time is 126 seconds in the 10-mega-ohm range. On multiplying 126 by 0.09, we get 11.34 μF as the measured value of the capacitor.

Example 2: For a 1000μF capacitor, delay time is 130 seconds in the 100-kilo-ohm range. On multiplying 130 by 9.0, we get 1170 μF as the measured value.

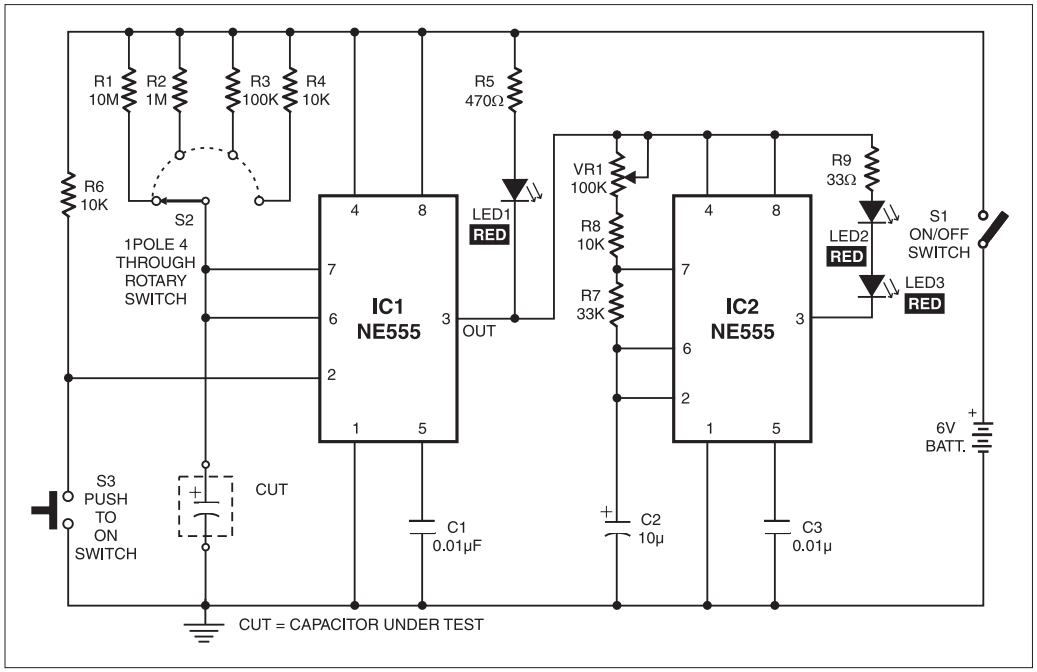


TABLE II

Capacitor value	Delay time in seconds				
	Range 10 mega-ohms	Range 1 mega-ohm	Range 100 kilo-ohms	Range 10 kilo-ohms	Measured value of capacitor
10 μF	126		–	–	0.09×126 = 11.34 μF
100 μF	–	128		–	0.9×128 = 115.2 μF
1000 μF	–	–	130		9.0×130 = 1170 μF
4700 μF	–	–	–	45	90.0×45= 4050 μF

The delay times and measured values of the capacitor are given in Table II. As the tolerance of electrolytic capacitor is very high, the near value of it can easily be infered from the measured values which is enough for all practical purposes.

KEY CHAIN LIGHT

T.A. BABU

A key chain with a built-in white LED comes in handy to help you at your front door or search your valuables in the dark. The intensity of white LED is 4000 to 5600 mcd (millicandela) at forward voltage of 3.6V and forward current of 20 mA.

Here's such an LED light circuit for key chains. It comprises a toroidal transformer and two complementary transistors, and is powered by a single AAA cell. Transistors T1 (BC547) and T2 (BC558) form a relaxation oscillator with capacitor C2 (0.01 μF) in the feedback loop. The feedback is controlled by

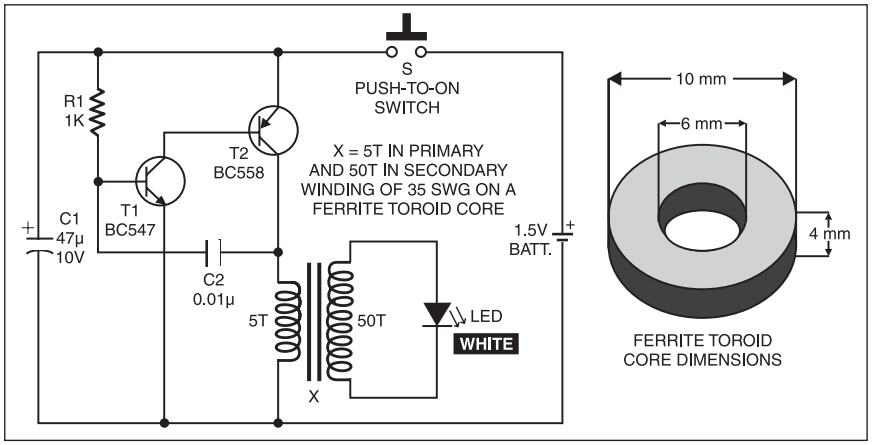


Fig. 1: Circuit for key chain light

the time constant of timing components R1 and C2, which controls the frequency of operation.

The toroidal transformer steps up the oscillator output to a sufficient value to flash the white LED. The values of R1 and C1 need not be precise. Use of surface mount devices will make the unit more compact.

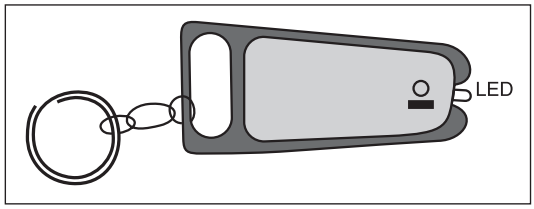


Fig. 2: Suggested enclosure for key chain light

A single 1.5V AAA cell gives enough brightness. For more brightness, connect two such cells in series. A good-quality white LED from a reputed manufacturer is highly recommended.

Caution. The white LED beam, when viewed directly, can harm the eyes.

AUTOMATIC NIGHT LAMP WITH MORNING ALARM

D. MOHAN KUMAR

This circuit automatically turns on a night lamp when bedroom light is switched off. The lamp remains 'on' until the light sensor senses daylight in the morning. A super-bright white LED is used as the night lamp. It gives bright and cool light in the room. When the sensor detects the daylight in the morning, a melodious morning alarm sounds.

The circuit is powered from a standard 0-9V transformer. Diodes D1 through D4 rectify the AC voltage and the resulting DC voltage is smoothed by C1. Regulator IC 7806 gives regulated 6V DC to the circuit. A battery backup is provided to power the circuit when mains fails. When mains supply is available, the 9V rechargeable battery charges via diode D5 and resistor R1 with a reasonably constant current. In the event of mains failure, the battery automatically takes up the load without any delay. Diode D5 prevents the battery from discharging backwards following the mains failure and diode D6 provides current path from the battery.

The circuit utilises light-dependant resistors (LDRs) for sensing darkness and light in the room. The resistance of LDR is very high in darkness, which reduces to minimum when LDR is fully illuminated. LDR1 detects darkness, while LDR2 detects light in the morning.

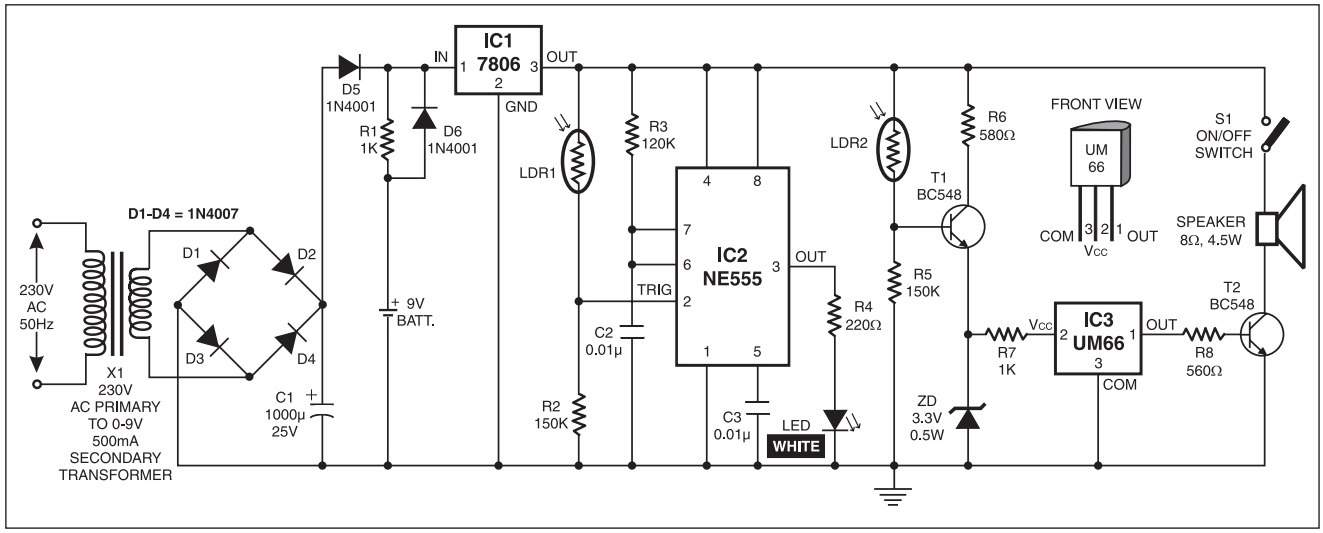
The circuit is designed around the popular timer IC NE555 (IC2), which is configured as a monostable. IC2 is activated by a low pulse applied to its trigger pin 2. Once triggered, output pin 3 of IC2 goes high and remains in that position until IC2 is triggered again at its pin 2.

When LDR1 is illuminated with ambient light in the room, its resistance remains low, which keeps trigger pin 2 of IC2 at a positive potential. As a result, output pin 3 of IC2 goes low and the white LED remains off. As the illumination of LDR1's sensitive window reduces, the resistance of the device increases.

In total darkness, the specified LDR has a resistance in excess of 280 kiloohms. When the resistance of LDR1 increases, a short pulse is applied to trigger pin 2 of IC2 via resistor R2 (150 kiloohms). This activates the monostable and its output goes high, causing the white LED to glow.

Low-value capacitor C2 maintains the monostable for continuous operation, eliminating the timer effect. By increasing the value of C2, the 'on' time of the white LED can be adjusted to a predetermined time.

LDR2 and associated components generate the morning alarm at dawn. LDR2 detects the ambient light in the room at



sunrise and its resistance gradually falls and transistor T1 starts conducting. When T1 conducts, melody-generator IC UM66 (IC3) gets supply voltage from the emitter of T1 and it starts producing the melody. The musical tone generated by IC3 is amplified by single-transistor amplifier T2. Resistor R7 limits the current to IC3 and zener diode ZD limits the voltage to a safer level of 3.3 volts.

The circuit can be easily assembled on a general-purpose PCB. Enclose it in a good-quality plastic case with provisions for LDR and LED. Use a reflective holder for white LED to get a spotlight effect for reading. Place LDRs away from the white LED, preferably on the backside of the case, to avoid unnecessary illumination. The speaker should be small so as to make the gadget compact.

Readers' Comments:

In the 'Automatic Night Lamp with Morning Alarm' circuit, please clarify:

Q1. Why the circuit did not specify the type of LED to be used in it?

Q2. Can I use a bulb instead of LED so that I can use it as an outdoor lamp? Can I use a relay with it for operating bigger bulbs? If so, how?

Q3. I find no preset in the circuit. At which intensity of light thus the LED glows? I want to attach a preset along with it so that I can specify the intensity of light at which the bulb should glow.

Please indicate the changes to be made in the circuit for adding the above.

Aloysius Tany Fernandez
Coimbatore

The author D. Mohan Kumar replies:

I thank Mr Fernandez for showing keen interest in my circuit. Replies to his queries are given below:

A1. The circuit is a simple switch with a different application. One can use any type of LED rated 3.6V available in the market.

A2. A 3V torch bulb cannot be used at the output of IC1 since the IC cannot tolerate much current drain. IC 555 has only 200mA current sinking capacity. One can modify the circuit for outdoor applications by incorporating a relay driver circuit from the output of IC1. Any medium-power transistor such as CL100 or BD139 can be used to switch on the relay when the output of IC1 goes high. A 60W bulb can be connected to the relay contacts, if required. If such a modification is done, a 100k preset should be connected in series with the LDR from positive rail to adjust sensitivity of the LDR.

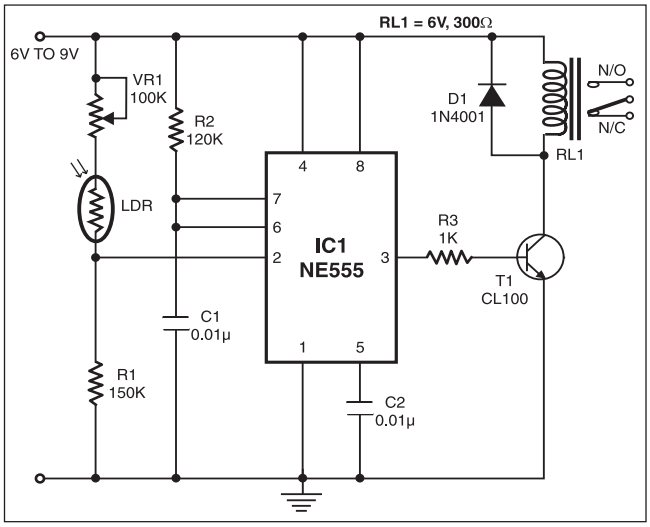


Fig. 1: Modified circuit of automatic night lamp for outdoor lighting

A3. In the given design, a preset is not needed since the LED turns on at the moment when the room light is switched off. It is not necessary to adjust the light of LED since it gives a cool light sufficient for reading purpose also.

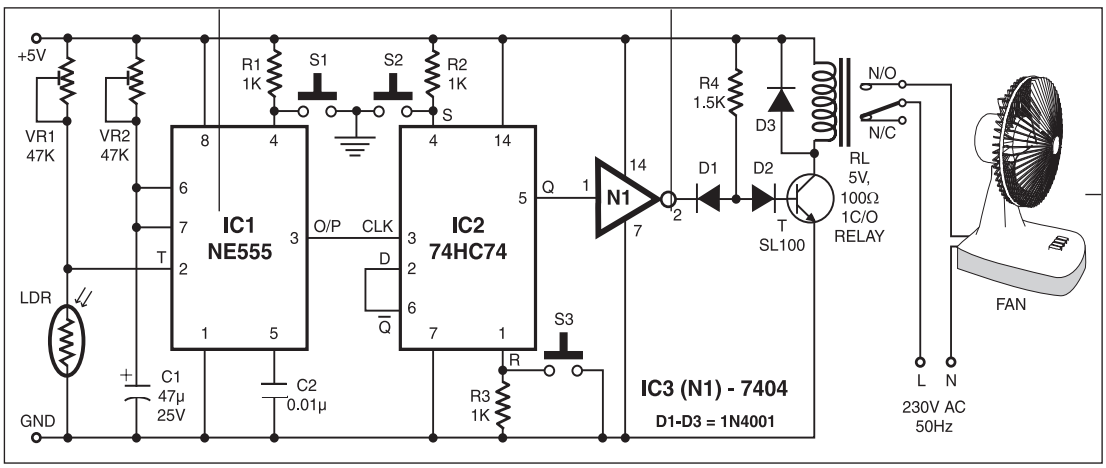
The modified circuit for outdoor operation is given in Fig. 1 here.

FAN ON/OFF CONTROL BY LIGHT

V. GOPALAKRISHNAN

This circuit lets you turn on/off a fan by just directing torchlight or other light toward its light-dependent resistor (LDR). The circuit is powered from a 5V power supply.

Preset VR1 and a light-dependent resistor (LDR) work as the potential divider. Normally,



the LDR's resistance is high (20 kilo-ohms) in darkness and low (2 kilo-ohms) in light. This value of high and low resistances varies for other LDRs. Preset VR1 is used for setting the intensity of light, while preset VR2 is used for setting the output time period of IC1.

When light falls on the LDR, the monostable (IC1) triggers at pin 2, making its output at pin 3 from low to high. This low-to-high transition forms a clock for D flip-flop. The D flip-flop is operated in toggle mode by connecting its Q output to D point. The flip-flop output goes to an inverter (N1). The inverter output is fed to the relay driver transistor.

When the inverter output is low, diode D1 conducts and the current is diverted into the inverter. Hence the relay does not energise. When the inverter output is high, diode D2 conducts and the current is diverted into transistor T. Hence the relay energises.

One terminal of the fan is connected to the normally-open (N/O) contact of the relay, while another terminal is connected to the neutral (N) of mains. The mains live (L) is connected to the pole of the relay. When the relay energises, the fan turns on. Otherwise, the fan remains off.

Switches S1 and S3 are for initial resetting of the monostable (IC1) and D flip-flop (IC2), respectively, and switch S2 is used for setting the D flip-flop. Paste a piece of paper on the face of the LDR so that it doesn't get activated by ambient light. Use a torch to light the LDR.

After initial resetting of the monostable and D flip-flop, the inverter output goes high and the fan turns on via the relay. When light falls on the LDR, the fan goes off. If torchlight is again directed toward the LDR, the fan turns on. The sequence repeats.

Initially if switch S2 is used to set the D flip-flop, the fan is held 'off'. The relay does not energise as the Q output of D flip-flop goes high to make the inverter output low. Directing the light towards the LDR at this moment turns the fan 'on.'

INTERCOM WITH MUSICAL RINGTONE

PRADEEP G.

Here's a low-cost two-way intercom circuit with musical ringtone that can be made using readily available components (see Fig. 1).

Melody generator IC UM66 (IC1) generates ringtone for the intercom circuit. The supply voltage to IC1 is limited to 3.3 volts by the zener diode. IC LM386 (IC2) amplifies the signal received from either the condenser mic or the output of IC1. The speaker volume is adjusted with the help of VR1.

For two-way intercom operation, make two identical units as shown in Fig. 2. The output of one unit goes to the speaker of the remote unit, and vice versa. If you use separate power supplies for the two units, a 3-core cable (two wires for outputs and one wire for common ground) is needed for intercom connection. The wiring for intercom is shown in Fig. 2.

The two units can be housed inside separate handy phone cabinets or toy cell phone cabinets available in the market.

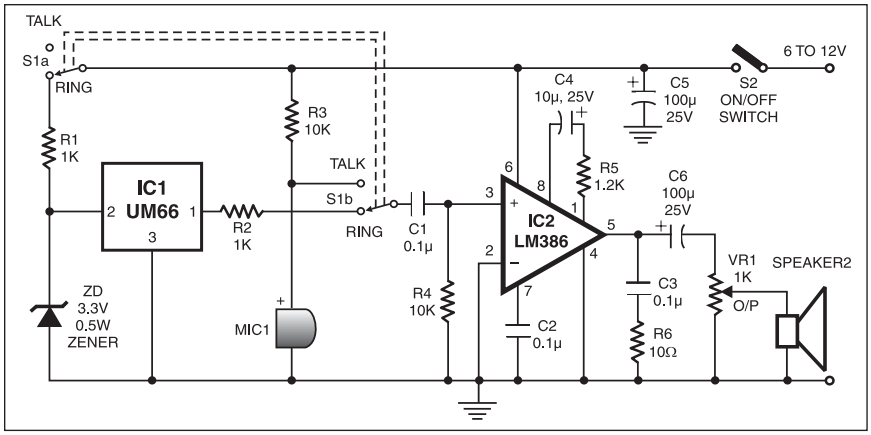


Fig. 1: Circuit for two-way intercom with musical ringtone

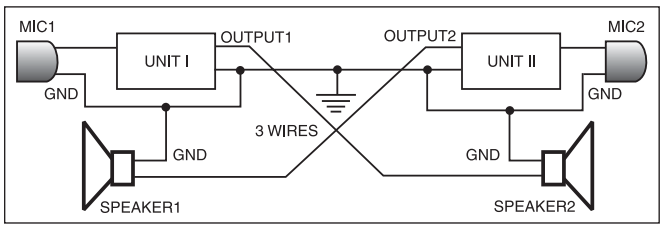


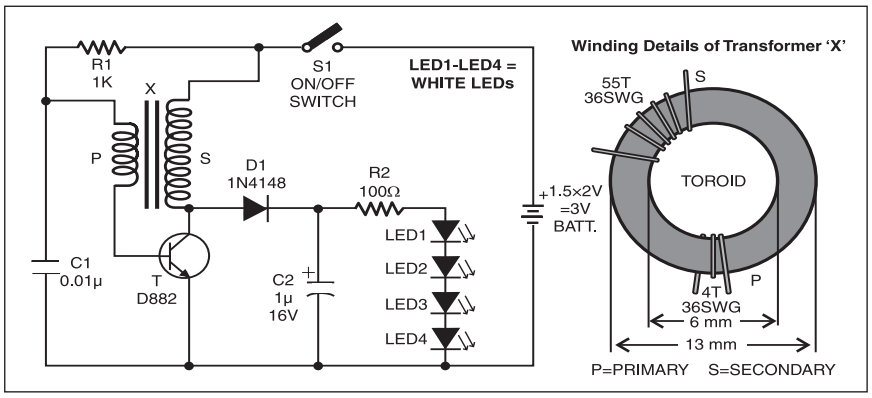
Fig. 2: Wiring of two intercom units

LED TORCH

T.A. BABU

LEDs are becoming increasingly popu-lar in many lighting appli cations. White LEDs are now common in torches.

Here's a simple and economical LED torch that operates off two 1.5V cells. For a white LED, the forward conduction voltage and the forward current are 3.6V and 20 mA, respectively. Capacitor C1, the transistor, and the transformer form a self-oscillating DC-DC converter. The transformer boosts the input battery voltage and supplies a high voltage to the white LEDs. Diode D1 and smoothing capacitor C2 supply the high voltage to the LED chain via resistor R2.



The light intensity is affected by the input battery voltage. As the trans-former is not readily available in the market, you may need to build it yourself. The transformer windings also affect the light intensity. The transformer coil is wound on a 4mm thick ferrite toroid having 13mm outer diameter and 6mm inner di-ameter.

Primary windings comprise 4 turns and secondary windings 55 turns of 36SWG copper-enamelled wire on toroid transformer core as shown on the right side in the figure. For the long life of 1.5V cells, keep switch S1 in 'on' position only when the torch is in use.

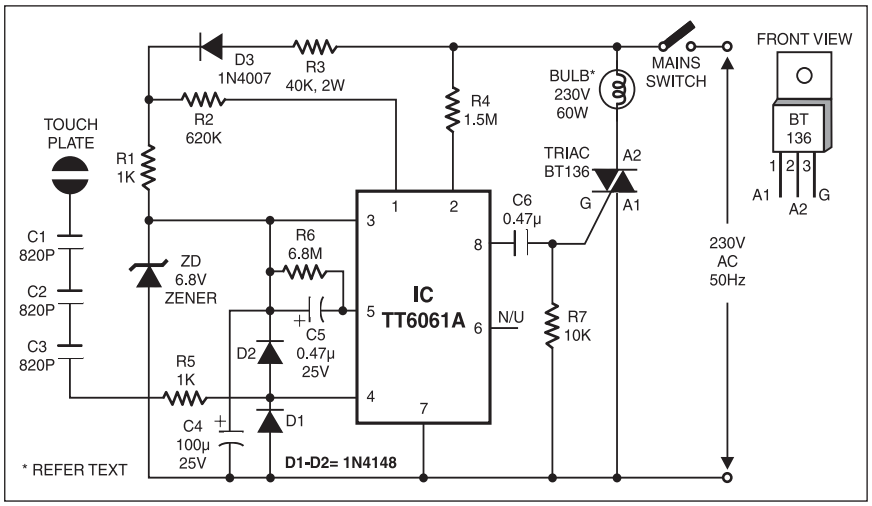
TOUCH DIMMER

K. KRISHNA MURTY

By simply touching this touch dim-mer you can increase the light in-tensity of incandescent lamps in three steps. The touch dimmer is built around 8-pin CMOS IC TT8486A/ TT6061A specifically manufactured for touch dimmer applications.

Initially, when mains switch is 'on,' the bulb is 'off'. Now, if you touch the touch plate, the bulb glows dimly. On second touch, the bulb gives medium light. At the third touch, the bulb is driven fully. An-other touch puts off the light.

Since the IC is highly sensitive, use a long wire to connect the IC to the touch sensor. The circuit uses minimum external components. For touch plate, you can use a simple copper plate of 1cm×1cm or even the end of the lead wire. Touch plate is coupled to the touch detector through 820pF, 2kV capacitors C1, C2, and C3 connected in series. Internally IC TT6061A's touch signal is connected to the counter/decoder via a resistor and clock input CK is connected to the counter/de-coder via a frequency generator.



Line frequency signal is taken through R4 at pin 2 of IC TT6061A. At zero cross-ing, the triac (BT136) triggers to drive a 200W bulb.

The 6.8V power supply is taken directly from mains through resistors R1 and R3, diode D3, capacitor C4, and zener diode and fed to power-input pin 3 of the IC. Capacitors C1, C2, and C3 connected between touch input pin 4 and touch plate remove the shock potential from the touch plate, so do not replace these capacitors with a single capacitor or with a capacitor of a lower voltage rating. Mains potential exists in the circuit. Needless to say, it is dangerous to touch the circuit when mains is 'on.'

Pin Assignments of IC TT6061A

Pin No.	Pin name	Function description
1	CK	System clock input
2	FI	50Hz line frequency
3	V_{DD}	Power input pin for V_{DD}
4	TI	Touch input
5	CI	Sensor control input
6	NC	Not connected
7	V_{SS}	Power input pin for V_{SS}
8	AT	Angle-trigger output

EXPERIMENTAL STUDY OF SWITCHED-CAPACITOR CIRCUIT

PROF. SOMNATH CHAKRABARTI

In recent years switched-capacitor cir-cuits have gained popularity due to their suitability in integrated circuits.

The two major problems faced by IC designers are:

1. Implementing a large-value resistor requires a large chip area.
2. Difficulty arises in designing RC active circuits having precise time constants.

In switched-capacitor circuits both these difficulties are circumvented by simulating a resistor with a capacitor that occupies smaller chip area. Also time constant is made to depend on a highly stable crystal oscillator frequency and ratioed capacitors.

The basic principle of simulating a resistor with a switched capacitor is shown in Fig. 1.

Suppose initially the SPDT switch (S) connects the capacitor (C) to the voltage (V) at contact 1. The charge (Q) acquired by the capacitor is:

Q = C .V ... ...(1)

If the switch is now shifted to contact 2, the capacitor discharges fully and the charge of the capacitor passes to the ground.

If the switch is moved back and forth very rapidly with frequency f or time period T (=1/f), the average current flowing through the current meter will be:

I=C .V/ T ... ...(2) or,

V/I = 1/C.f ... (3)

From Eq.(3), it's clear that the switched capacitor simulates a resistor (R) given by:

R= 1/ C.f ... ...(4)

As switching cannot be done manually, you require an electronic switch. IC CD4066 (IC1) includes four such CMOS bidirectional analogue switches in a single package. An electronic switch has a control terminal apart from two input/ output (I/O) terminals. The enable input pins (pins 5 and 13) are driven by a unipolar square wave (clock) having two discrete voltage levels. When control is at logic 0 (0V) the switch is open, and when the control is at logic 1 (12V) the switch is closed. Using two such switches you can build the required SPDT switch as shown in Fig. 2. When one switch is closed, the other has to be kept open, and vice versa. That means you must drive the two control terminals by complementary clocks.

Timer IC NE555 (IC2) is operated in astable mode to generate the square wave. The 50% duty cycle is achieved by equal-value timing resistors (R4=R5) and diode D1. The inverting (NOT) operation is performed traditionally by transistor T1 (2N3904) to generate the complementary clock. A frequency meter can read out the actual frequency.

You should perform the experiment on a breadboard. Connect a suitable capacitor (C2) for generating the required test frequency between pin 2 of IC2 (NE555) and ground. Connect test capacitor C1 (preferably mylar or teflon capacitor) between pin 2 (joined to pin 3) of IC1 and ground.

Adjust trimpot VR1 to 3 volts. This voltage is applied to pin 1 of IC1. Connect the test capacitor (C1) and note the frequency and current readings on the frequency meter and the microammeter, respectively. Calculate resistance (R) from the relationship:

R = V/I

Now check whether this value satisfies Eq.(4).

Record observations for various combinations of f and C in the form of a table.

Finite values of the 'on' resistance of analogue switches (approx. 80 ohms) and microammeter resistance (typically 1-2 kilo-ohms) will be all swamped out if you simulate a high-value resistor.

Observation Table for Simulation of Resistor with Switched Capacitor
Constant voltage (V) = 3.04 volts

Timing capacitor C2	Measured frequency f (kHz)	Test capacitor C1	1/C.f (kΩ)	Measured current (μA)	R=V/I (kΩ)
0.01 μF	1.66	2.00 nF	301.2	10	304.0
0.01 μF	1.66	4.78 nF	125.4	24.5	124.1
0.01 μF	1.66	0.01 μF	59.70	48.0	62.91
4.7 nF	3.94	3.41 nF	74.37	40.0	75.75

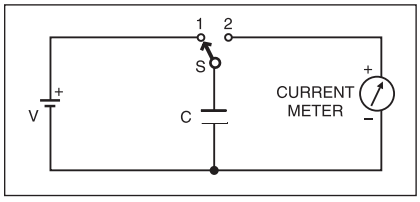


Fig. 1: The principle of switched capacitor

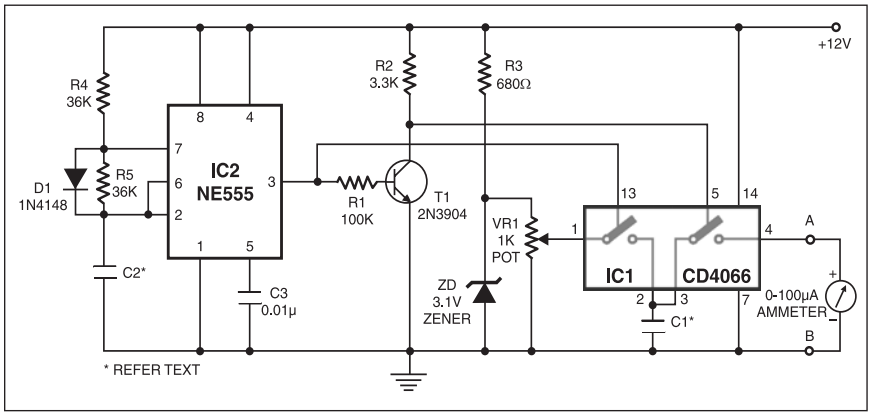


Fig. 2: Circuit for simulation of resistor with switched capacitor

Top 20 Projects (Out of 91)

- Car Security System With Remote Control
- DTMF Remote Control System
- Programmable Logic Controller
- Temperature Measurement using Transistor as Sensor
- Microcontroller-Driven Data Display
- Microprocessor-Controlled Thermometer
- DTMF 8-Channel Switching Via Powerline
- Multi-feature Emergency Light
- Proportional Load Control Using PC
- Number Guessing Game
- Lead-Acid Battery Charger With Voltage Analyser
- Wireless TV Headphone Circuit
- Electric Shock Gun
- 0-100°C Temperature Detector
- Earth Fault Protector
- Burglar Alarm System
- Low-cost Hearing Aid
- Vehicle Security System
- Smoke Extractor
- Mosfet-Based Preamplifier FM Radio Dxing
- Automatic Night Lamp With Morning Alarm
- Intercom With Musical Ringtone

EFY SINCERELY YOURS

EFY Enterprises Pvt Ltd

D-87/1, Okhla Industrial Area, Phase I, New Delhi 110 020; Ph: +91-11-26810601-03

Fax: +91-11-26817563; E-mail: info@efyindia.com; Website: www.efyindia.com