

REMOTE-CONTROLLED REAL-TIME CLOCK WITH DEVICE CONTROLLER



■ D.S. OBEROI AND HARINDER DHINGRA

This project makes use of a TV remote control. Using RC5 coding, a real-time clock chip is set to control five different alarm settings. These settings can also be used to switch on an external device. Up to eight devices can be controlled with this project. The circuit is based on ATMEL ATmega16 microcontroller and Maxim's DS1307 real-time clock chip. An LCD module allows for user interface.

Circuit description

Fig. 1 shows the block diagram of the remote-controlled real-time clock with device controller. It comprises six sections, namely, IR detector remote control sensing and decoding unit, real-time clock, LCD interface, device switching unit, microcontroller and power supply unit

The microcontroller unit integrates all the sub-systems and system software operates the system. Fig. 2 shows the circuit of the device controller.

Remote control section. This circuit makes use of a Philips TV remote for device switching and RTC parameter setting. It uses RC5 coding format, which is also known as 'biphase coding.' In RC5-coded signal, each bit has a uniform duration.

Table I shows how all the commands of RC5 remote control are encoded. The first two bits designated 'S' are 'start' bits, which are used to adjust and synchronise the receiver. These bits are used to calculate and analyse the bit length of other bits.

The third bit is a 'toggle' bit (T), which toggles every time a button is pressed at the remote control. This bit

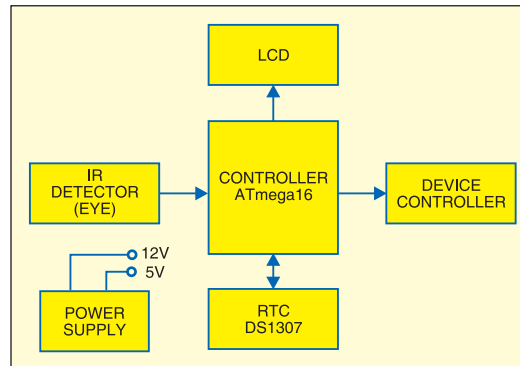


Fig. 1: Block diagram of remote-controlled real-time clock with device controller

TABLE I
RC5 Coding Format

S	S	T	A4	A3	A2	A1	A0	C5	C4	C3	C2	C1	C0
---	---	---	----	----	----	----	----	----	----	----	----	----	----

TABLE II
Remote Command Codes

Remote key	Numeric value (C5-C0)	Function
0	0	Press this key to start the setting of the RTC.
Volume +	16	Increases the value (of a given field). Used in 'Alarm Delete Mode' to change the selection to Yes or No
Volume -	17	Decreases the value (of a given field). Used in 'Alarm Delete Mode' to change the selection to Yes or No
Mute	13	Shifts to next field (in RTC setting mode)
TV/AV	56	Starts the RTC functioning. This remote control key acts as 'Enter' key (confirmation key). Used in alarm mode, to confirm the setting
Power	12	Returns to Display Mode (from alarm mode). This key lets the user to return to the previous menu; when used in any of the Alarm Menu, acts as 'Escape' key and cancels the action in Alarm mode
Timer/sleep	38	Enters Alarm menu for setting/deletion/viewing alarm data.
Channel +	32	Scrolls forward alarm setting 'See' of Alarm menu
Channel -	33	Scrolls backward alarm setting 'See' of Alarm menu

is used to identify whether the button is really pressed or whether an obstacle came in between the path of the IR remote and IR receiver.

Bits A4 down to A0 are used to identify the device. So a maximum of

32 devices can be interrogated to respond individually to the same type of coding without any disturbance.

Bits C5 down to C0 are control/command bits. Therefore a maximum of 64 commands can be equipped in an RC5 type remote. Decimal equivalent of a few command codes used in this project are listed at Table II.

Pressing any command/control button on the remote generates code signal, which is received by IR receiver-demodulator (TSOP1738). The output of the IR de-

modulator is normally high, which changes to low when any of the buttons on the remote is pressed. It is fed to PD3 through the I/O interface line of the microcontroller. The microcontroller decodes the incoming

RC5 data stream and subsequent actions are taken based on the this information.

LCD unit. The LCD module (16-character×2-line) is interfaced with the microcontroller. Data pins 7 through 14 of the LCD module are connected to port A (PA0 through PA7) of the

microcontroller. Register-select (RS) pin 4 and enable pin (pin 6) of the LCD are interfaced with PC1 and PC2 of the microcontroller, respectively. R/W pin of the LCD (pin 5) is pulled low permanently and thus is always in writing mode. Back light of the LCD is controlled by PD0 line of the

microcontroller with the help of transistor T1. All the information is displayed on the LCD, which forms the basic user interface unit.

Real-time clock. IC DS1307 (IC2) from Maxim (Dallas Semiconductor) is a serial RTC chip with calendar function. This chip also incorporates 56 bytes of NV RAM. Data and address are transferred serially through I²C bidirectional bus, which obviates the need for a large number of interface lines. The bidirectional data is read and written with the help of just two I/O interface lines.

In this chip, the clock operates in either 24- or 12-hour format with AM/PM indicator. In calendar mode, end of the month is automatically adjusted for the months with less than 31 days and leap year compensation is valid up to year 2100.

The memory map of DS1307 (also referred to as 'time keeper register map') is shown in Table III. For setting the clock and calendar at power-'on,' the data is first written to the designated memory location of the RTC chip and, during the normal operation. It is read back from each specific memory location during the clock and calendar display functions.

The alarm settings are stored in NV RAM; these can be deleted and altered at any given point of the operation. All this is achieved with the help of the system software. In the project, only five alarms have been allowed but the same can be changed as per requirement; only the

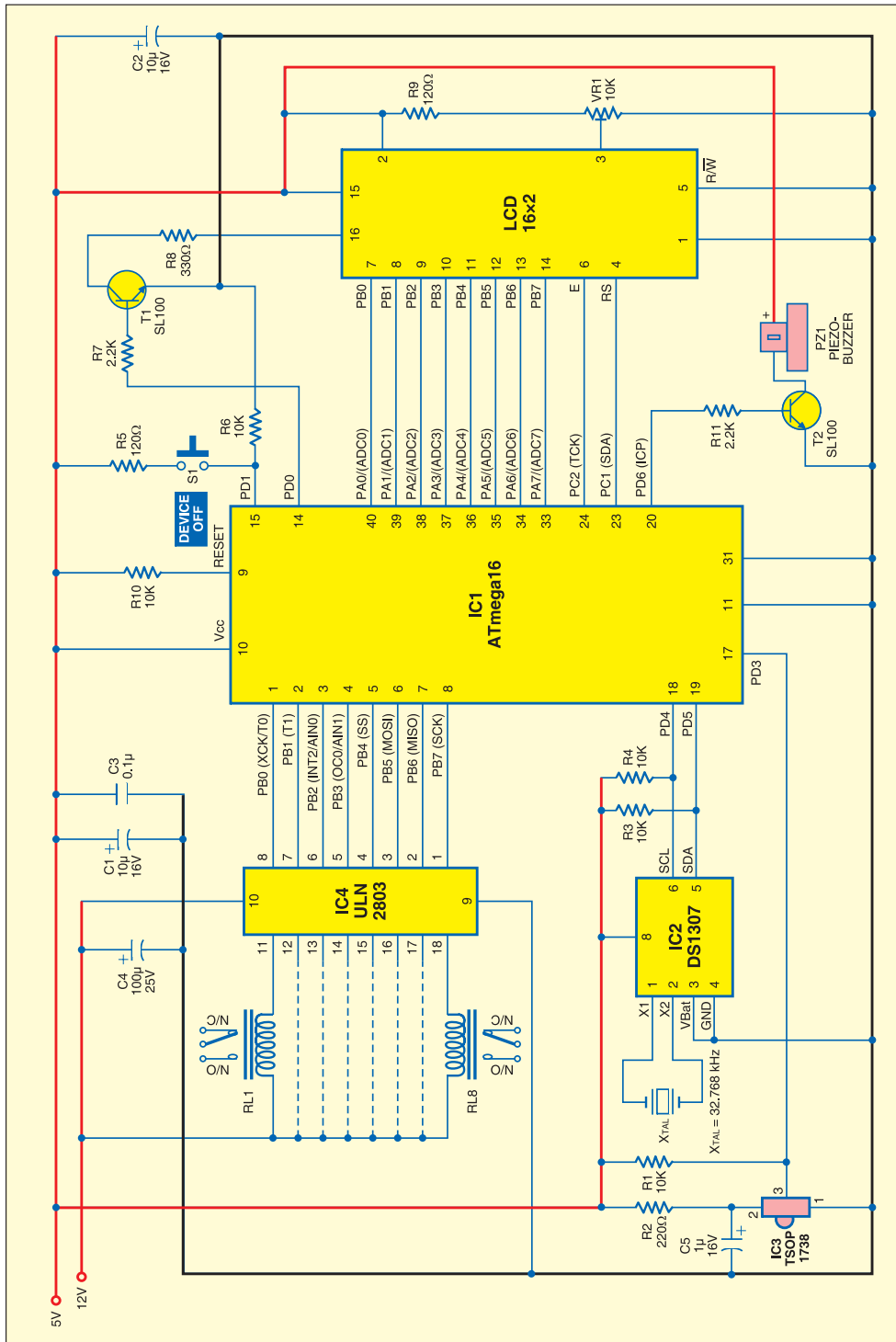


Fig. 2: Circuit of remote-controlled real-time clock with device controller

alarm limits have to be changed.

Clock pulse to the RTC is provided by a 32.768kHz crystal. As per the convention of I²C interface communication, the device address while writing into DS1307 chip is 0xD0 (in binary format 11010000) and while reading the data from DS1307 the device address is 0xD1 (in binary format 11010001). Due to this unique device addressing technique, a number of I²C-interfaced chips can be attached on SCL and SDA interface lines of RTC. Thus, at a given time, only one I²C device will respond to the data on the I/O interface lines.

Device control. Devices are connected to contacts of the relay and relays are controlled through the outputs of IC4. Port B of IC1 (PB0 through PB7) is interfaced with pin 8 down to pin 1 of IC4 (ULN2803) to control relays RL1 through RL8, respectively. Five out of eight devices (device 1 through 5) switch on with their respective alarm settings and the remaining three devices switch on directly.

Whenever an alarm is activated for a particular setting, the respective de-

Address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Function	Range	
00H	CH	10 seconds			Seconds			Seconds	Seconds	00-59	
01H	0	10 minutes			Minutes			Minutes	Minutes	00-59	
02H	0	12 24	10 hour PM/AM	10 hour	Hours			Hours	Hours	1-12 +AM/PM 00-23	
03H	0	0	0	0	0	Day		Day	Day	01 07	
04H	0	0	10 date		Date			Date	Date	01 31	
05H	0	0	0	10 month		Month			Month	01-12	
06H	10 year			Year			Year	Year	Year	00-99	
07H	Out	0	0	SQWE	0	0	RS1	RS0	Control	-	
08H									RAM	RAM	00H-FFH
									56×8		

0 – Always reads back as 0

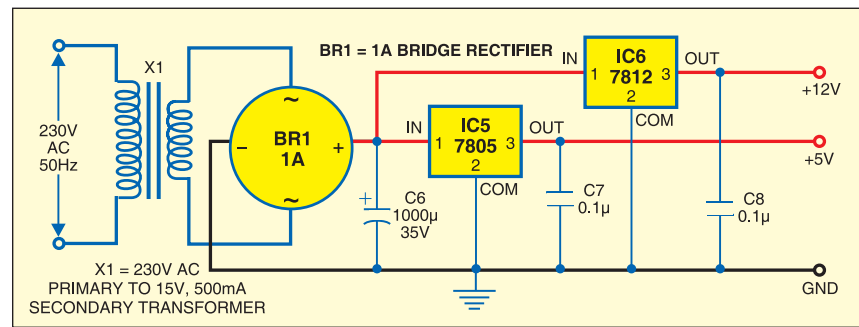


Fig. 3: Power supply

vice is switched on. The device can be switched off only by pressing the respective key number on the remote control.

Controller. The controller unit is based on ATmega16 low-power, 8-bit CMOS microcontroller with AVR enhanced RISC architecture. This microcontroller has 16 kB of in-system programmable flash memory, 512 bytes on EEPROM, 1 kB of SRAM, two 8-bit timers, one 16-bit timer, 32 general-purpose I/O lines and 32 general-purpose working registers. It integrates all the subsystems to form a complete unit.

Although this microcontroller already has an RTC counter and I²C lines, we have used a separate RTC chip and special I²C software to show the functionality with the dedicated RTC chip, which is much easier to control and manage in terms of time keeping as well as achieving calendar functions. Regarding I²C, we wanted to show how the same can be achieved with the help of the software without

the use of any specialised hardware functionality of the microcontroller; and also help the user to implement the same on any other microcontroller that does not have the dedicated facility of I²C interface.

This controller unit first uses serial data received from the IR detector to get the information regarding the pressed key and then takes the appropriate action, and also updates the information on the LCD side-by-side. When in operational mode, the microcontroller reads the data from the RTC chip every one second (achieved through timer interrupts-compare mode). During this 1-second interrupt, it checks for the alarm setting, controls the snooze, and also updates the time/date information on the LCD. In between, it checks the data from the IR detector and takes action accordingly.

Power supply. Fig. 3 shows the power supply circuit. The 230V, 50Hz AC mains power supply is stepped down by transformer X1 to deliver a

PARTS LIST

Semiconductors:

IC1	- Atmega16 AVR microcontroller
IC2	- DS1307 real time clock
IC3	- TSOP1738 IR receiver module
IC4	- ULN2803 darlington array
IC5	- 7805 5V regulator
IC6	- 78012 12V regulator
T1, T2	- SL100 npn transistor
BR1	- 1A bridge rectifier
	- LCD module (16×2)

Resistors (all ¼-watt, ±5% carbon):

R1, R3, R4,	- 10 kilo-ohm
R6, R10	- 220 ohm
R2	- 120 ohm
R5, R9	- 2.2 kilo-ohm
R7, R11	- 330 ohm
R8	- 10 kilo-ohm preset

Capacitors:

C1, C2	- 10µF, 16V electrolytic
C3, C7, C8	- 0.1µF ceramic disk
C4	- 100µF, 25V electrolytic
C5	- 1µF, 16V electrolytic
C6	- 1000µF, 35V electrolytic

Miscellaneous:

X1	- 230V AC primary to 15V, 500mA secondary transformer
S1	- Push-to-on switch
PZ1	- Piezo buzzer
X _{CRY}	- 32.768 KHz crystal
RL1-RL8	- 12V, 1C/O relay

secondary output of 15V at 500 mA. The transformer output is rectified by a full-wave bridge rectifier BR1, filtered by capacitor C6, and regulated by IC5 and IC6. IC5 and IC6 output is 5V and 12V, respectively. The regulated 12V is used for relay-driver IC4 and the rest of the circuit operates with 5V. Capacitors C7 and C8 bypass any ripple present in the regulated power supply.

An actual-size, single-side PCB pattern for the remote-controlled real-time clock with device controller (Fig. 2 and Fig. 3) is shown in Fig. 8 and its component layout in Fig. 9.

Software

The system software is used to achieve integration and functionality. The software for this project is written in 'C' language and compiled using WINAVR. WINAVR is available free of cost for Windows and Linux operating systems. It is capable of handling all the AVR functionalities like UART, timer, ADC, interrupts, etc and offers the facility of writing the program in 'C.' The finally obtained Intel hex code file was burnt into AVR's flash memory using a suitable programmer. The microcontroller uses a 4MHz internally generated clock. To activate, fuse bytes have to be programmed as follows:

Fuse low byte = D3

Fuse high byte = 99

The software for the entire project was written in modules in accordance with the functionality of each subsystem. The files used in this project are shown in Table IV.

The ds1307.c file contains the code for controlling the functions related to serial DS1307 RTC. The chip uses I²C interface, which essentially uses only two I/O interface lines—SDA and SCL—for bidirectional address and data communication. This file essentially uses sub-routines, which have been indicated in the I2Cmaster.h header file. Subroutines i2c_write() and i2c_read() are used for writing and reading the data to and from the RTC chip through DS1307_write() and DS1307_read(), respectively.

The Assembly code for the complete I²C operations and communication is written in the I2Cmaster.S file,

TABLE IV
Software Files and Their Function

File Name	Function
.rclock.c	Main file which integrates all the functions of the sub-units of the project
ds1307.c	Contains the code for the controlling the functions of DS1307 RTC
.lcd.c	Contains the code for controlling the LCD
.lcd.h	Contains the global declarations and sub-routines, which are used by other files
.i2cmaster.h	Contains sub-routine information required for I2C communication
.i2cmaster.S	Contains code for controlling the I2C communication (optimised for 4.00 MHz operation)

which has been written for 4.00MHz operation, and for this reason, the microcontroller's fuses have been set to use the internal 4.00MHz clock.

The data stored in the RTC chip is in BCD format. The BCD data is changed into binary format for normal processing and displaying. For this purpose, the BCD2bin() subroutine is used. Similarly, the user data is first converted into BCD format with bin2BCD() sub-routine and then written into the RTC chip. This file contains all the sub-routines that are used in rclock.c file modules.

I²C-interfaced chips are initiated in a particular sequence for their proper functioning, and this has been taken care of during initialisation and reading/writing data from/to DS1307.

The user alarm settings are stored in the NV RAM of DS1307 in two fields, namely, hours and minutes. In this project, a maximum of five alarm settings have been allowed but this number can be increased by changing the variable MAX_ALARM value (in lcd.h file).

Whenever the alarm setting data is to be viewed, altered, deleted or added, the program checks the user SRAM area, and based on the value stored in the ALARM_COUNT_RGST memory location, further action is permitted. While clock is functioning, the data from this SRAM area, which has

now been allocated to an array, is checked and the desired action is taken when the alarm setting data matches the current time.

i2cmaster.S and *i2cmaster.h*. These files are related to control of the I²C interface communication. The details of the SCL and SDA pins and I/O interface line are defined in the i2cmaster.s file and the Assembly code file is compiled along with other code files; the parameter to be used in 'makefile' for this file is ASRC.

lcd.c This file contains the code for control of functioning of the attached LCD module. The code controls the initialisation of the LCD, data writing on the LCD, and also the movement, characteristics and location of the cursor. It offers the facility to write data on the LCD character-by-character or string-wise. The command set used in the software is based on the command set used in the LCD based on Hitachi HD44780 ICs.

lcd.h This header file contains all the constant variable values and names of the subroutines used by various files used in the software. It clearly indicates which variable can be used as a global variable and which of the sub-routines can be used across the software files.

rclock.c This file contains the code that integrates all the subunits together. It contains the code that will call the initialisation routines to initialise I²C chip, LCD, microcontroller ports, set timer and its IRQ, check data from the IR detector, extract information regarding the key pressed, etc.

Depending upon the key pressed, the file initiates the change in RTC data like year, month, day, date, hours and minutes and also in viewing, deleting and adding new alarm settings. It takes control of the devices attached to the unit and controls their status depending upon the alarm settings.

All the above-mentioned functions are implemented in this file by means of specific implementation of sub-routines and global variables wherever required.

All the sub-routines used in the various files are clearly marked and commented for their functionalities

with codes explained.

The entire project software was compiled and debugged with WINAVR development environment, which is based on avr-gcc 3.3.1 (WinAVR-20030913).

Remote control operation

The entire device operation is controlled through keys of the Philips remote control. The remote keys are used as shown in Table II. The functions of these keys are given below.

Key 0 is used to change the setting of the RTC. When the RTC is functioning normally, it can be switched to setting mode by pressing the '0' key and immediately the cursor will blink on the hour field, indicating that this value can be changed through 'volume up' and 'volume down' keys to a desired value.

Keys 1 through 8 are used, during normal operation, to control the state of the external devices. Key 1 is used to control Device 1 and so on. Devices 1 through 5 are switched on according to the alarm timing also; however, these can be switched off by pressing the respective key on the remote control itself.

Keys 1 through 3 are also used in the Alarm menu. Key 1 is used to add a new alarm value to the alarm table. Key 2 is used to delete the alarm setting for the table (by changing Yes/No field with volume up/down key). Key 3 is used to scroll through the alarm settings by use of channel up and down keys.

Power key is used for deactivating the alarm and also returning one setup back in the Alarm menu. Whenever you want to return from the Alarm's sub-menu (like New, Del and See), press this key. Also, this key is used to return from the Alarm menu to display screen.

Mute key is used to change the field (from hour to minute to day of the week to day of the month to month of the year and, finally, the year and back to hour and so on and so forth).

Volume Up and Down keys are used to change the value of a given field when setting the clock data or the alarm data.

TV/AV key starts the RTC func-



Fig. 4: Screen display after switch on



Fig. 5: Screen display after pressing 'Timer' key



Fig. 6: Delete an alarm setting



Fig. 7: View the alarm setting

tioning once the desired settings have been made. This remote control key acts as 'Enter' key (confirmation key).

Timer key takes the user to alarm setting menu.

Clock mode setting

When the RTC is switched on, the screen display is as shown in Fig. 4. The second field value shows 40 seconds. These indicated initial values have been set in subroutine `i2c_initialisation()`. If some other values are required, the desired values can be written.

The cursor blinking on the hour field indicates that value can be changed through volume up and down key—which will now change the hour value from '0' to '23' hours and back to '0' again.

Once the desired value of the hour has been achieved, you can change to minute field by pressing 'Mute' key. The change of field is indicated by cursor blinking on the appropriate field (here minute field). Value of minutes can be changed from '0' to '59' and again to '0.' Again, after the desired value has been achieved, the field can be changed to week day by pressing 'Mute' key. Now week days can be changed by pressing volume up and down keys.

Next field-change data is related to day of the month (the date can be changed from '1' to '31') and after this, the month of the year can be changed (from January to December). The year can be changed from '00' to '99'. Take care when setting the month and the days in that particular month.

The process remains the same when setting the alarm value also, where only hour and minute fields are available for setting. Once the setting has been made as per the requirement, TV/AV key can be set and now the clock will start functioning with the desired settings. The functioning of the clock can be observed by change in the second field.

Alarm setting

As explained earlier, the given design allows five alarm settings with snooze facility and control of five devices only. You can enter the alarm menu by pressing 'Timer' key. The screen at this point of operation is shown in Fig. 5.

Through this menu, you can set new alarm (1New), delete alarm setting (2Del) and view alarm settings (3See). The respective menu can be activated by pressing key 1, 2 or 3. In new setting mode, if five alarm settings have been made, the same is indicated by a 'Full' sign. In case new setting has to be made, an already existing alarm setting has to be deleted and then only new alarm setting will be allowed.

To return from 'New' mode to 'Alarm' menu, press 'Power' key. If space is available for new alarm setting, you can change hour-field data and minute-field data by using 'Volume Up,' 'Volume Down' and 'Field' keys, i.e., 'Mute' key. Once the desired

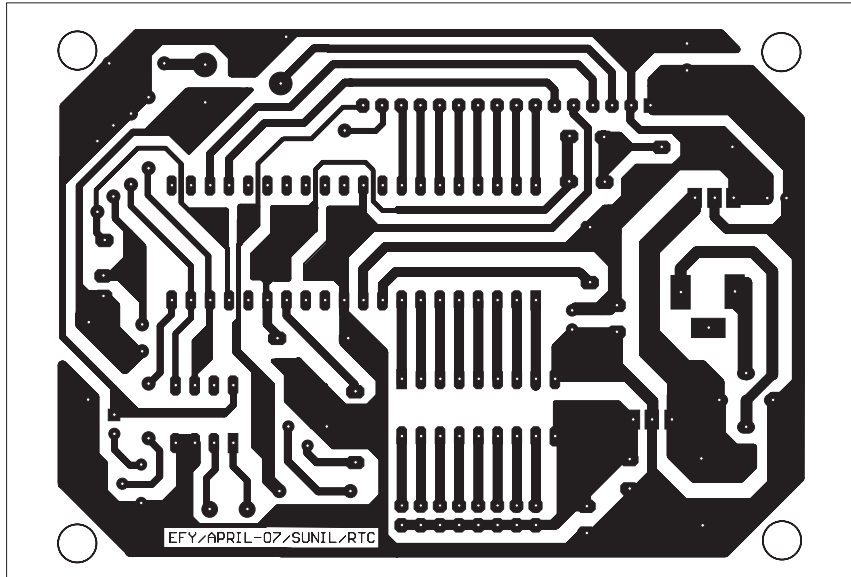


Fig. 8: Actual-size, single-side PCB for the remote-controlled real-time clock with device controller

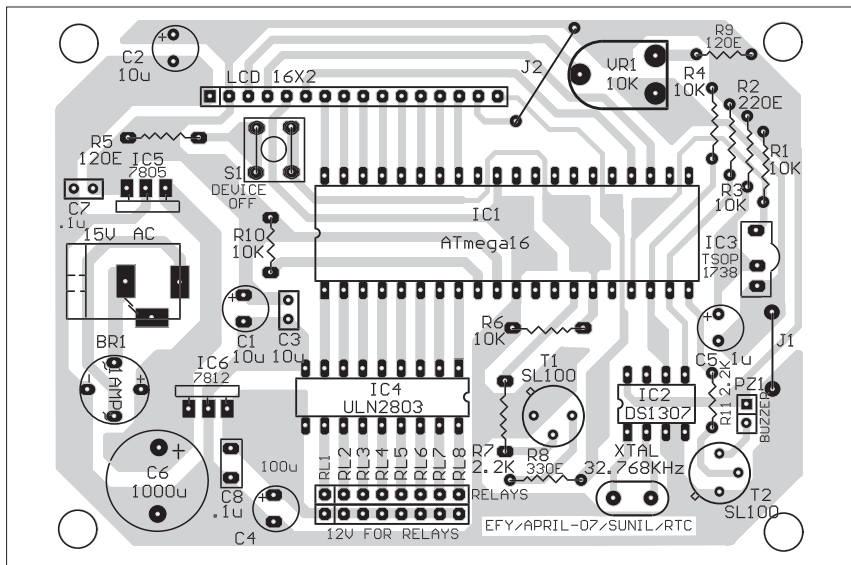


Fig. 9: Component layout for the PCB

settings have been made, pressing the 'TV/AV' key stores the alarm settings in the NV RAM of the DS1307 chip.

To delete an alarm setting, first press key '2'. The display will be as shown in Fig. 6.

The alarm settings/values can be scrolled with the help of 'Channel Up' and 'Channel Down' keys. For a setting that is to be deleted, change the status of the indicator from 'No' to 'Yes' by 'Volume Up' and 'Volume Down' keys. Then delete data by pressing 'TV/AV' key and immediately the message 'Deleting Data' is displayed, indicating

that one alarm setting has been deleted.

To view the alarm setting data, you can use key '3' in 'Alarm' menu. The screen at this point will be as shown in Fig. 7, which indicates the total number of alarm settings in the memory (Tot:) and data being viewed. You can scroll through the data with the help of Channel Up/Down key.

As you scroll through the data, the field 'Data:' indicates the alarm setting as per the programmed sequence. The device-control field number will be the same as 'Data:' field number. (If 'Data:5,' the device to be controlled

will be '5,' and if 'Data:1' device 1 is attached with this data field.) If no alarm data is found in the memory, 'No Alarm Data' is shown on the LCD.

In 'New' and 'Del' sub-menus of 'Alarm' menu, 'TV/AV' key is used to confirm the desired setting/action. To cancel settings/actions, press 'Power' key at any time of the operation.

By default, all the five alarm settings have been made initially through the software. These values can be changed in sub-routine dummy() found in the ds1307.c file.

As soon as the alarm is activated, the LCD backlight glows periodically along with the buzzer sound. Deactivate the alarm with 'Power' key; however, the respective-controlled device will continue to be 'on.' The alarm can be put into snooze mode by pressing any key on the remote control. In snooze mode, the buzzer will not sound but LCD backlight will glow. The snooze delay can be changed by setting the constant snooze_delay variable (in the lcd.h file) to a desired value.

No doubt the alarm time can be set in any sequence; the external device will switch on in a sorted time sequence. For example, if the alarm time is '01:21' at location-3 and '01:10' at location-5, device-5 will switch on only at '01:10' and device-3 will switch on only at '01:21,' even though in the alarm sequence the location of device-3 is before device-5. When the clock is switched on, all the controlled external devices are in switched-off condition.

In snooze mode, the alarm can be de-activated by pressing 'Power' key.

All the external devices can be switched off simultaneously by pressing S1. This will work only in normal functioning of the device and not during the alarm mode.

The project was assembled on a general-purpose PCB.

EFY notes. All the software files related to this article have been included in this month's EFY-CD. For testing the project, you can use the RC5-format remote published in EFY March 2007. ●

D.S. Oberoi is principal design engineer at DOEACC Centre, Srinagar/Jammu, and Harinder Dhingra is a lecturer at GCET, Jammu