

TEMPERATURE INDICATOR USING AT89C52



ADITYA RANE

Here's a microcontroller-based temperature indicator that displays the temperature in the range of -55°C to 125°C . Besides AT89C52 microcontroller, it uses a temperature sensor chip and an LCD module. The indicator outputs the calibrated data in digital form. The program for the microcontroller is written in C and not in Assembly language. Since C program has well-defined syntax, it far outweighs

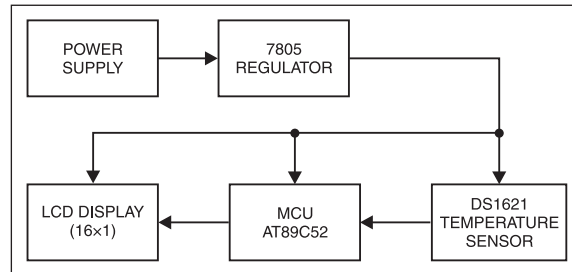


Fig. 1: Block diagram of temperature indicator using AT89C52

the merits of the Assembly language program.

The circuit

Fig. 1 shows the block diagram of the temperature indicator using microcontroller AT89C52. The power supply for the circuit is regulated by IC 7805 and supplied to different parts of the

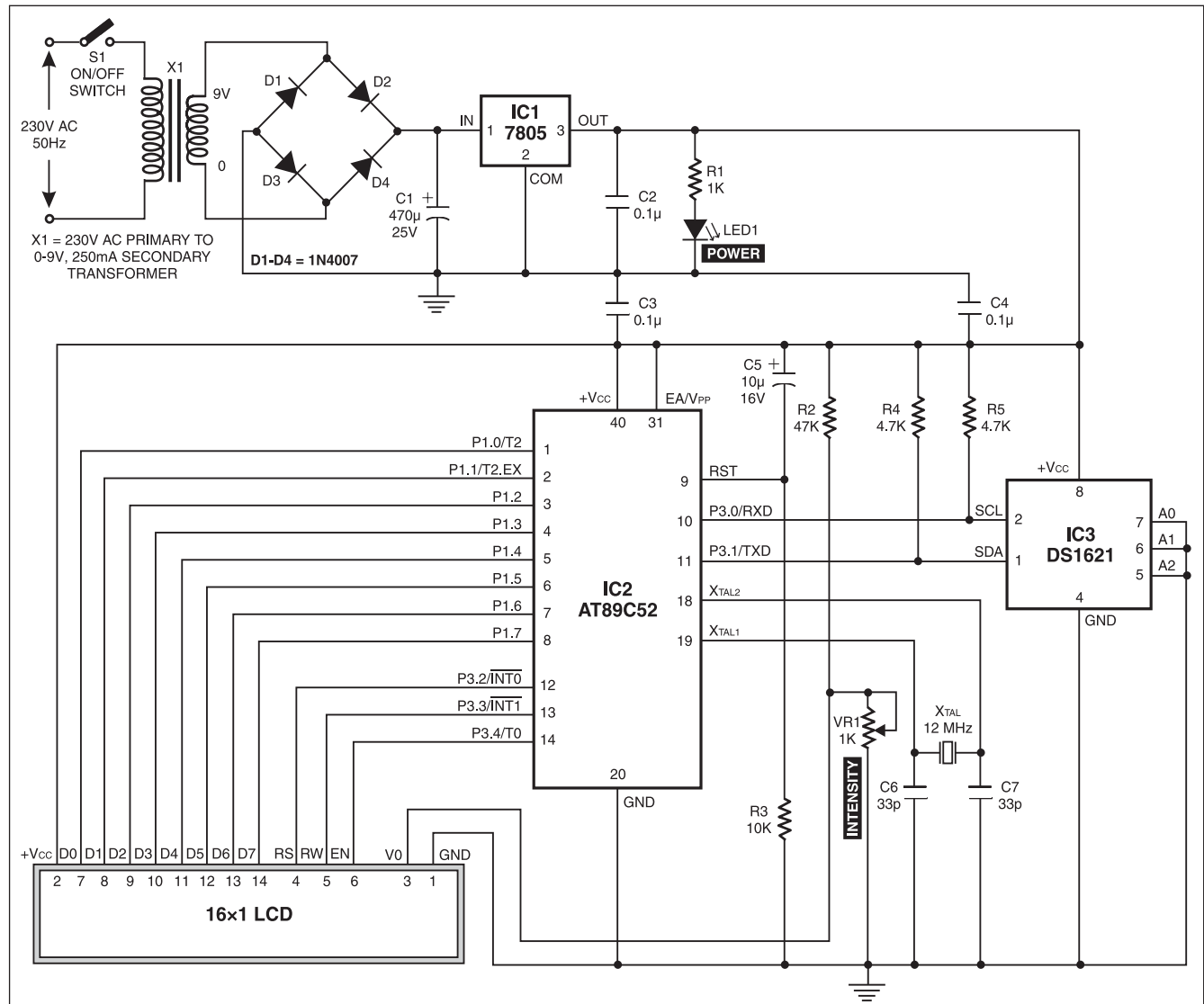


Fig. 2: Circuit diagram of temperature indicator using AT89C52

PARTS LIST

Semiconductors:

- IC1 - 7805 regulator IC
- IC2 - AT89C52 microcontroller
- IC3 - DS1621 temperature sensor
- D1-D4 - 1N4007 rectifier diodes
- LED1 - Red LED

Resistors (all 1/4-watt, ±5% carbon, unless stated otherwise):

- R1 - 1-kilo-ohm
- R2 - 47-kilo-ohm
- R3 - 10-kilo-ohm
- R4, R5 - 4.7-kilo-ohm
- VR1 - 1-kilo-ohm preset

Capacitors:

- C1 - 470µF, 25V electrolytic capacitor
- C2, C3, C4 - 0.1µF ceramic disk
- C5 - 10µF, 16V electrolytic capacitor
- C6, C7 - 33pF ceramic capacitor

Miscellaneous:

- Transformer - 230V AC primary to 0-9V, 250mA secondary
- Crystal - 12 MHz
- LCD - 16 × 1 LCD module
- S1 - On/Off SPST switch

unit. DS1621 is the temperature sensor chip. The microcontroller unit (MCU) reads the temperature from the sensor. The temperature data is compared with certain user-defined temperature values and processed inside the MCU as per the program and then sent to the LCD for display.

Fig. 2 shows the circuit of temperature indicator using microcontroller AT89C52. Working of each section of the circuit is covered in the following paragraphs.

Power supply. The power supply unit consists of a step-down transformer (230V AC primary to 0-9V, 250mA secondary), bridge rectifier and voltage regulator. The output of the transformer is fed to bridge rectifier diodes D1 through D4 (each 1N4007). The ripple from the output bridge rectifier is filtered by capacitor C1 and fed to regulator IC 7805. The regulated output is given to the temperature sensor, microcontroller unit and LCD module, respectively.

When switch S1 is closed, LED1 glows to indicate the presence of power in the system.

Temperature sensor. Temperature sensor chip DS1621 (IC3) is an 8-pin DIP IC. Its pin details are shown in Fig. 3 and the internal block diagram in Fig. 4. The chip can measure temperatures from -55°C to +125°C in 0.5°C increments, which are read as 9-bit values. It can operate off 2.7V to 5.5V. Data is read/written via a 2-wire

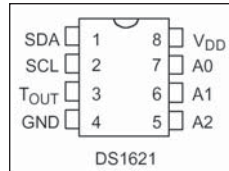


Fig. 3: Pin details of IC DS1621

serial interface. Pins 1 and 2 of the temperature IC are connected to pins 11 and 10 of the microcontroller, respectively.

The thermal

below user-defined low temperature TL. User-defined temperature settings are stored in the non-volatile memory. Temperature settings and temperature readings are all communicated to/from IC DS1621 over a 2-wire serial cable. The most significant bit (MSB) of the data is transmitted first and the last significant bit (LSB) is transmitted last.

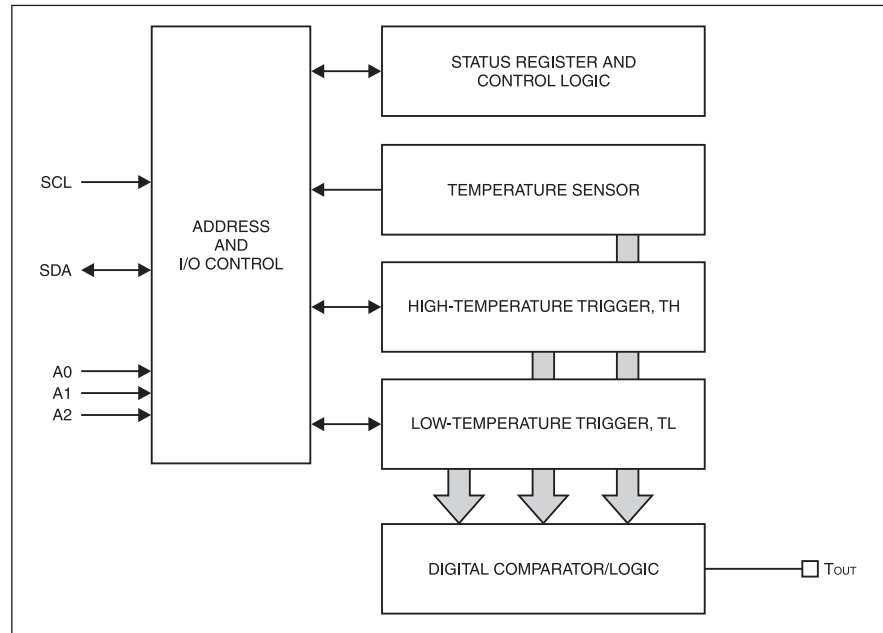


Fig. 4: Internal block diagram of IC DS1621

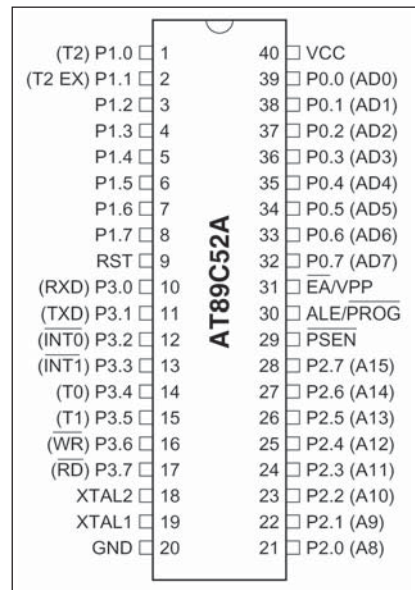


Fig. 5: Pin details of IC AT89C52

alarm output (T_{out}) of IC DS1621 activates when the temperature exceeds user-defined high temperature TH. The output remains active until the temperature drops

Addressing. The chip address of DS1621 comprises internal preset code nibble '1001' (binary) followed by externally configurable address pins/bits A2, A1 and A0. The eighth bit of the address byte is determined by the type of operation (either read or write) that is to be performed. For writing to the device the eighth bit is '0' and for reading from the device the eighth bit is '1'. In our case, A2, A1 and A0 pins are grounded and hence the device address for writing is '1001000b' or 90(hex) and for reading the device address is '10010001b' or 91(hex).

Configuration/status register. This register can be accessed for reading or writing by issuing command byte AC(hex) from the master (82C52). This register is particularly required if DS1621 is used for thermostat control, since it contains flag bits THF (high-temperature flag) and TLF (low-temperature flag) which are set to '1' when temperature crosses the respective limits earlier written into TH and TL registers. It also contains the flag bit (Done), which is set to '1' when results of conversion are available after issuing of

start conversion command EE(hex). The other bits of configuration register are defined below:

| | | | | | | | |
|------|-----|-----|-----|-----|---|-----|-------|
| DONE | THF | TLF | NVB | 1 | 0 | POL | 1SHOT |
| MSB | | | | LSB | | | |

'NVB' is the non-volatile memory busy flag, '1' is write to an E² memory cell in progress, '0' indicates that non-volatile memory is not busy, 'POL' is non-volatile output polarity bit ('1' = active-high and '0' = active-low) and '1SHOT' is one-shot

mode. A copy to E² may take up to 10 ms. If 1SHOT is '1,' DS1621 will perform one temperature conversion upon reception of the Start Convert T protocol. If 1SHOT is '0,' DS1621 will continuously perform temperature conversions. This bit is non-volatile.

Command Set. Complete command instruction set for accessing various internal registers as well as for starting and stopping of conversion process are given in Table I. For understanding the exact sequence in which Start bit, address byte,

acknowledgement bit, command byte(s) and data byte(s) are to be sent along the I²C bus, please refer to the datasheet of DS1621, wherein these aspects have been explained in proper detail. This will help in understanding the contents of the main program.

Microcontroller unit. Microcontroller AT89C52 (IC2) is a 40-pin IC from Atmel. Its pin details are shown in Fig. 5. Like AT89C51, it also belongs to the 8031/8051 family. Microcontroller AT89C52 has a 256 × 8-bit internal random-access memory (RAM), eight interrupt sources and 8 kB of flash memory compared to 128x8-bit

internal RAM, six interrupt sources and 4 kB of flash memory in AT89C51. By combining a versatile 8-bit CPU with flash memory on a monolithic chip, Atmel AT89C52 is a powerful, highly flexible and cost-effective solution to many embedded control applications.

Ports 0 and 2 are 8-bit bidirectional input/output (I/O) ports. These ports haven't been used in this temperature indicator.

Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Ports 1.0 through 1.7 are connected to pins 7 through 14 of the LCD. Port-1 output buffers can sink/source four TTL inputs.

Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Ports 3.0 and 3.1 of IC2 are connected to serial clock line (SCL) and serial data line (SDA) of IC3, respectively. Ports 3.2 through 3.4 are connected to pins 4 through 6 of the LCD, respectively. Port-3 output buffers can sink/source four TTL inputs.

A 12MHz crystal oscillator is connected to X_{TAL1} and X_{TAL2} pins for operation of the microcontroller. A high pulse on RST pin (pin 9) while the oscillator is running resets the microcontroller. In this circuit, this pin is connected to +Vcc through capacitor C5 (10 μF, 16V). The external-access enable pin (EA) is connected to +Vcc for internal program executions. This pin also receives the 12V programming-enable voltage (V_{pp}) during flash programming when

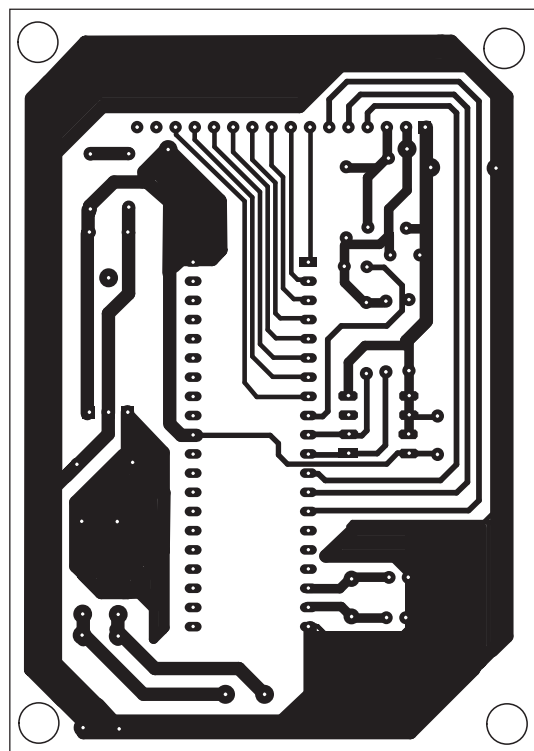


Fig. 6: Solder-side PCB layout for temperature indicator using AT8952

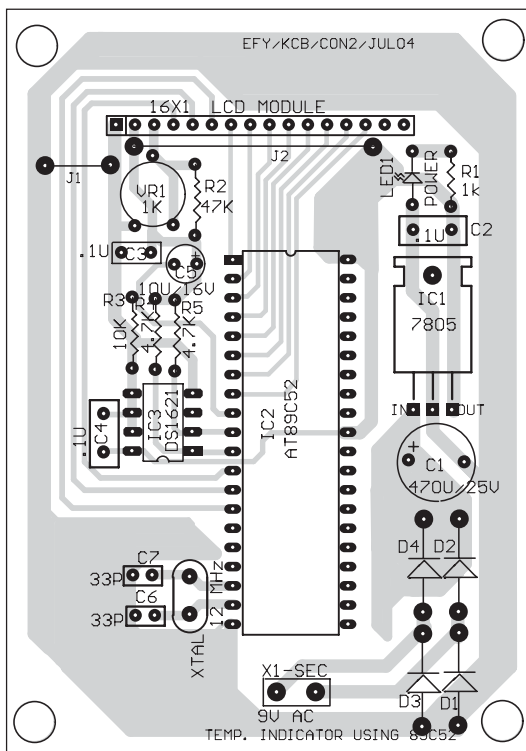


Fig. 7: Component layout for the PCB

TABLE I
DS1621 Command Set

| Instruction | Description | Protocol |
|----------------------|---|----------|
| Read Temperature | Reads last converted temperature value from temperature register. | Aah |
| Read Counter | Reads value of count remaining from counter. | A8h |
| Read Slope | Reads value of the slope accumulator. | A9h |
| Start Convert T | Initiates temperature conversion. | EEh |
| Stop Convert T | Halts temperature conversion. | 22h |
| Access TH | Reads or writes high temperature limit value into TH register. | A1h |
| Access TL | Reads or writes low temperature limit value into TL register. | A2h |
| Access Configuration | Reads or writes configuration data to configuration register. | ACh |

12V programming is selected.

The program

The C-language program for microcontroller AT89C52 is compiled using cross-compiler C51 Version 7.10 from Keil Software. The demo version of this compiler is available for free on the Website 'www.keil.com.' It can compile programs up to 2 kB only, which is sufficient for writing most programs.

For testing the display, the program Hello.c is given here. This program, when loaded to AT89C52, displays "Hello! How R U?" on the LCD. The Hello.c program has nothing to do with temperature. It just guarantees a perfect communication between the LCD and the microcontroller. For temperature indication, the program Temp52.c is used. The programs Hello.c and Temp52.c, along with the hex files, are given at the end of this article.

The communication interface between the temperature sensor and the microcontroller chip follows the I²C (Inter Integrated Circuit) standard, which is implemented in 'C' here. I²C is a simple master/slave type interface. Simplicity of the I²C system is primarily due to the bidirectional 2-wire (SDA and SCL) design and the protocol format. Bidirectional communication is through 2-wire lines (which are either active-low or passive-high). In the program, the `i2c_stop`, `i2c_start`, `i2c_write` and `i2c_read` functions are used for communicating Clock and Data from DS1621 to P3.0 and P3.1 of AT89C52, respectively. Such functions as `command`, `ready` and `display` in the program are used for driving the LCD.

Program compilation for 8051 family controller. Keil C51 can compile C programs for most of the Atmel family microcontrollers. It also supports other devices. Unlike other cross-compilers (Hi-Tech, IAR, SDCC, etc), Keil C51 offers such features as fast code generation, strong multitasking environment, real-time operating system and inbuilt code optimisation. To enjoy these features, you'll need full version of the compiler.

Keil C51 has options to generate Assembly code and all the code listing supported by 8051 family, but Assembly language generated cannot be recompiled on any other assembler. As far as code generation is concerned, it uses minimum RAM and on-chip flash, allowing faster and optimised program in Intel-Hex format, which can be loaded to the microcontroller

TABLE II
Pin Connections of the LCD

| Pin No. | Functions |
|---------|------------------------------------|
| Pin 1 | Ground (Gnd) |
| Pin 2 | + Vcc |
| Pin 3 | V0 (display intensity control) |
| Pin 4 | RS (connected to P3.2 of AT89C52) |
| Pin 5 | R/W (connected to P3.3 of AT89C52) |
| Pin 6 | EN (connected to P3.4 of AT89C52) |
| Pin 7 | D0 (connected to P1.0 of AT89C52) |
| Pin 8 | D1 (connected to P1.1 of AT89C52) |
| Pin 9 | D2 (connected to P1.2 of AT89C52) |
| Pin 10 | D3 (connected to P1.3 of AT89C52) |
| Pin 11 | D4 (connected to P1.4 of AT89C52) |
| Pin 12 | D5 (connected to P1.5 of AT89C52) |
| Pin 13 | D6 (connected to P1.6 of AT89C52) |
| Pin 14 | D7 (connected to P1.7 of AT89C52) |
| Pin 15 | Backlight +Vcc (not used) |
| Pin 16 | Backlight Gnd (not used) |

using any programmer. Conversion of C program into Intel-Hex format takes only a few seconds. In fact, you don't require all that long Assembly program in order to generate the output hex file.

LCD

For display, a Lampex make 16x1 LCD (model GDM1601A) was used. Pin connections of this LCD are given in Table II. Pins 15 and 16 haven't been used. Pin 3 is connected to the circuit ground through a 1-kilo-ohm preset that is used to control the light intensity of the LCD. Note that the Hitachi make 16 × 1 LCD (HD44780A00) will not work in this project.

Construction

The circuit of this temperature indicator using microcontroller AT89C52 can be assembled on any general-purpose, single-side PCB. The microcontroller chip and the temperature sensor chip are mounted on the respective IC bases. Ensure a proper contact between pins of the IC bases and the solder points on the PCB. Capacitors C3 and C4 must be connected near IC2 and IC3, respectively. The actual-size, single-side PCB layout for the circuit and its component layout are shown in Figs 6 and 7, respectively.

Program compilation

After you've installed Keil C51 in your PC, you can compile C program and generate hex file in either DOS or Windows mode. Here, program compilation for the

program Hello.c has been explained. The same procedure is to be followed for the temperature indication program Temp52.c. For more example programs, refer to the directory in your hard drive where Keil is installed in the example folder.

DOS mode. 1. Installation of Keil C51 automatically generates 'Keil' folder in your computer's C drive.

2. Go to 'C:\Keil\C51\Bin' folder inside 'Keil' folder.

3. Copy 'Hello.c' into 'Bin' folder.

4. Copy 'Regx52.h' from 'C:\Keil\C51\Inc\Atmel' folder into 'C:\Keil\C51\Bin' folder.

5. Type 'c51 Hello.c' against the prompt and press Enter key.

6. Type 'bl51 Hello.obj.' This command is used for linking the Hello.obj file created by Keil C51.

7. Type 'oh51 Hello.' This command is used for creating the hex file.

Windows mode. 1. Installation of Keil C51 software automatically creates the icon 'Keil uVision2' on the desktop.

2. Double-click 'Keil uVision2.'

3. Suppose you have kept 'Hello.c' under 'C:\Windows\Desktop\Hello' folder. Open 'Hello.c' from the 'File' menu.

4. From the menu bar, select 'Project/New Project.' Name the new project and save it with extension '.uv2.'

5. Select CPU as Atmel/AT89C52.

6. Choose 'Yes' in the option "Copy standard 8051 code to current project folder."

7. Choose 'View/Project Window.' A 'Project Workspace' window appears.

8. Double-click 'Target 1.'

9. Right-click 'Source Group1' and select "Add files to Group 'Source Group1.'" A window appears.

10. Add 'Hello.c' and close this window.

11. Double-click 'Source Group1' on the 'Project Workspace' window. Now the file name 'Hello.c' appears.

12. From 'Project' menu, select 'Options for File 'Hello.c.' In 'Properties,' choose file type as 'C source file.'

13. Again from 'Project' menu, select 'Options for Target 'Target1.'" A screen appears.

14. Choose 'Output' and tick on 'Hex File' for generating the hex file. Again choose 'Listing' option and tick on 'Conditional and Assembly Code'.

15. Open the Project menu and select 'Build Target' or press F7. The compiler shows ""Hello" 0 Error(s), 0 Warning(s)" in the output window just below the project window.

16. Close the screen and go to the 'Hello' folder to see the generated hex file and listing file.

Load the hex file into the microcontroller chip using a programmer. (Here we've used Atmel Flash Programmer from Frontline Electronics.) Now integrate the microcontroller chip into the populated PCB comprising the temperature sensor and the LCD module.

Troubleshooting

1. Check the COM port on your PC before programming.

2. In case there is no message even if all the connections are correct, adjust the intensity control potentiometer (VR1) for display.

3. Check whether your hex file matches with the hex file given below in

the article.

4. If the LCD shows wrong characters, replace it with another make LCD.

5. If DS1621 is not connected properly to AT89C52, the display will be completely blank.

EFY Lab note. All the source codes and relevant files of this article have been included in this month's EFY-CD.

TEMP52.C

```

/* Written By: Aditya Rane
T.E Computer Engg, Lokmanya Tilak College of Engineering,
New Bombay, Vashi
E-mail: aditya@orionengg.com
Program for temperature indicator compiled under keil
C */

#include <stdio.h>
#include <string.h>
#include <Regx52.h>

//-----
//Global Variable
//-----
int temperature;
#define HIGH 0x01 // Active High Signal
#define LOW 0x00 // Active Low Signal
#define TRUE 0x01 // Active High State
#define FALSE 0x00 // Active Low State

//-----
// Functions Prototyping
//-----
void ready (void);
void command (int);
void display (char *);
void i2c_stop (void);
void i2c_start (void);
void i2c_write (unsigned char);
unsigned char i2c_read (void);
void convert (unsigned char);

//-----
// Port Defination
//-----
#define DATA P3_1 // Serial data
#define CLOCK P3_0 // Serial clock

//Beginning of Main Program
void main (void)
{
int tmp;
char str[16];
bit flag = FALSE;
unsigned char ch;
void command (int);
void display (char *);
command(0x3c);
command(0x0c);
command(0x06);
while(1)
{
i2c_start();
i2c_write(0x90);
i2c_write(0xEE);
i2c_stop();

i2c_start();
i2c_write(0x90);
i2c_write(0xAA);
i2c_start();
i2c_write(0x91);
ch = i2c_read();
i2c_stop();
temperature = 0;
convert(ch);
if(flag == FALSE)
{
flag = TRUE;

tmp = temperature;
}
else
{
if(tmp != temperature)
{

```

```

tmp = temperature;
sprintf(str, "%d%s", temperature, " Centigrade");
command(0x01);
command(0x80);
display(str);
}
}

//Delay Servive Routine
void delay_time (void)
{
unsigned int i;
for(i=0;i<100;i++);
}

//I2C Start Function
void i2c_start (void)
{
DATA = HIGH;
delay_time();
CLOCK = HIGH;
delay_time();
DATA = LOW;
CLOCK = LOW;
}

//I2C Stop Function
void i2c_stop (void)
{
unsigned char i;
CLOCK = LOW;
DATA = LOW;
CLOCK = HIGH;
delay_time();
DATA = HIGH;
i = DATA;
}

//I2C Data Write Function
void i2c_write (unsigned char j)
{
unsigned char i;
for(i=0;i<8;i++)
{
DATA = ((j & 0x80) ? 1 : 0);
j <<= 1;
CLOCK = HIGH;
delay_time();
CLOCK = LOW;
}
}

//I2C Data Read Function
unsigned char i2c_read (void)
{
unsigned char i,j;
j = 0;
i = DATA;
for(i=0;i<8;i++)
{
j <<= 1;
CLOCK = HIGH;
j |= DATA;
delay_time();
CLOCK = LOW;
}
}

```

```

return j;
}

//Binary to Decimal Conversion Function
void convert (unsigned char ch)
{
char x;
unsigned char arr[8] = {128,64,32,16,8,4,2,1};
if(((ch & 0x80) ? 1 : 0) == 0)
{
for(x=0;x<8;+ + x)
{
if(((ch & 0x80) ? 1 : 0))
temperature = temperature +
arr[x] * ((ch & 0x80) ? 1 : 0);
ch <<= 1;
}
}
else
{
ch = ~ch;
ch = ch + 1;
for(x=0;x<8;+ + x)
{
if(((ch & 0x80) ? 1 : 0))
temperature = temperature +
arr[x] * ((ch & 0x80) ? 1 : 0);
ch <<= 1;
}
temperature = -temperature;
}
}

//Display Ready Check Function
void ready (void)
{
P3_4 = 0x00;
P1 = 0xff;
P3_2 = 0x00;
P3_3 = 0x01;
P3_4 = 0x01;
while(P1_7)
{
P3_4 = 0x00;
P3_4 = 0x01;
}
P3_4 = 0x00;
}

//Display Command Function void command (int a)
{
ready();
P1 = a;
P3_2 = 0x00;
P3_3 = 0x00;
P3_4 = 0x01;
P3_4 = 0x00;
}

//Display Write Function
void display (char *str)
{
unsigned int i;
for(i=0;i<=strlen(str)-1;+ + i)
{
if(i == 8)
command(0xc0);
if(i == 16)
command(0x80);
ready();
P1 = str[i];
P3_2 = 0x01;
P3_3 = 0x00;
P3_4 = 0x01;
P3_4 = 0x00;
}
}
}

```

TEMP52.HEX

```
:100F270025642573002043656E7469677261646583
:090F3700008040201008040201B2
:100DA800C2007F3C7E00120F9A7F0C7E00120F9AC1
:100DB8007F067E00120F9A120F8B7F90120EB87F5B
:100DC800EE120EB8120F6A120F8B7F90120EB87FB8
:100DD800AA120EB8120F8B7F91120EB8120F098F3C
:100DE80034120F6AE4F508F509AF34120CF220004A
:100DF8000AD20085082285092380BC5E236509708D
:100E08004E522650860B08508228509237538FF46
:100E180075390F753A2785083B85093C753DFF757F
:100E28003E0F753F2C7B007A00792412085C7F0105
:100E38007E00120F9A7F80E00120F9A7B007A0044
:080E48007924120E50020DBFC7
:0E0F7C00E4FFFE0FB00010EEF64644E70F53F
:010F8A002244
:0F0F8B00D2B1120F7CD2B0120F7CC2B1C2B02211
:100F6A00C2B0C2B1D2B0120F7CD2B1A2B1E433FC6A
:010F7A003541
:010F7B002253
:020E8B00AD0784
:100EBA00E4FCED30E703D38001C392B1ED25E0FDF8
:100ECA00D2B0120F7CC2B00C80E7A2B1E433FC6A
:070EDA00D2B0120F7CC2B080
:010EE10022EE
:100F9000E4FDA2B1E4FCED25E0FDD2B0A2B1E433E9
:0D0F19004205120F7CC2B00C80E7A2B1E433E9
:010F260022A8
:020CF2008F353C
:100CF40078377C007D007BFF7A0F79387E007F088F
:100D0400120C2C2E53530E7047F0180027F00EF7080
:100D140041F536E53530E7047F0180027F00EF605E
:100D24002374372536F8E6FD7C00E5357E0030E790
:100D3400047F0180027F00120C7FEF2509F509EE84
:100D44003508F508E53525E0F5350536E536B4080A
:100D5400C2226335FF0535E4F536E53530E7047F17
:100D64000180027F00EF602374372536F8E6FD7CAE
:100D74000E5357E0030E7047F0180027F00120C1D
:100D84007FEF2509F509E53508F508E53525E0F589
:100D9400350536E536B4080C2C3E49509F509E4958A
:030DA40008F50847
:010DA7002229
:100F4000C2B47590FFC2B2D2B3309706C2B4D2B465
:050F500080F7C2B4228D
:0E0F9A00120F408F90C2B2C2B3D2B4C2B422C2
:060E50008B358A3689375C
:100E5600E4F538F539AB35AA36A937120F55EF2424
:100E6600FFFE34FFED3E5399FE5389E5042E59D
:100E7600396408453870067FC0FE120F9AE539645A
```

```
:100E860010453870067F80FE120F9A120F40AB3560
:100E9600AA36A937853982853883120C52F590D245
:100EA600B2C2B3D2B4C2B40539E53970A80553880E8
:010EB600A497
:010EB7002218
:03000000020FA844
:0C0FA800787FE4F6D8FD758148020DA8A2
:100B5C00E709F608DFFA8046E709F208DFFA803E7B
:100B6C0088828C83E709F0A3DFFA8032E309F60868
:100B7C00DFFA8078E309F208DFFA807088828C83D0
:100B8C00E309F0A3DFFA806489828A83E0A3F60884
:100B9C00DFFA805889828A83E0A3F208DFFA804C5E
:100BAC0080D280FA80C680D4806980F28033801035
:100BBC0080A680E809A80A880DA80E280CA80339E
:100BC0089828A83ECFAE493A3C8C582C8CC58316
:100BD200CFC0A3C8C582C8CC583CDDFE9DEE780E6
:100BEC00D89828A83E493A3F680DFF9ECFA9F065
:100BFC00EDFB2289828A83ECFAE0A3C8C582C8CCBB
:100CC00E583CCF0A3C8C582C8CC583CCDFEADED3
:100C1C00E880DB89828A83E493A3F208DFF980CC35
:100C2C0080F0E6010E4E60C388F0ED2402B4042E
:100C3C000050B9F582EB2402B4040050AF232345D5
:060C4C008223900BAC7343
:100C5200BB0110CE58229F582E5833AF583E0225057
:100C620006E92582F8E622BBFE06E92582E22A1
:0D0C7200E58229F582E5833AF583E49322BB
:100C7F00E8FD0A4A8F0CF8CF0A428CE8DF0A42E89
:020C8F00FE2243
:10080000E544238F8E60544227835300802783883
:10081000E475F00120C2C020C912001EB7F2ED28A
:10082000018018EF540F2490D43440D4FF30050BCE
:10083000EF24BFB41A0050032461FFE545600215A0
:10084000450548E5487002054730080D7835E475E0
:10085000F001120C2C0E202CA0A020EE27403D208E3
:100860008003E4C208F5448B358A368937E4F545C0
:10087000F547F548E54560077F2012083B80F57590
:1008800046FFC202C201C203C204C206C207C209B5
:10089000120809FF700D3008057F0012084CAF48A0
:1008A000AE4722B4255FC2D5C205120809FF24D085
:1008B000B40A00501A75F00A784530D505086FF1D
:1008C000106C6A426F62D5047002D20480D924DD
:1008D000CFB41A00EF5004C2E5D205020A4CD2028E
:1008E00080C6D20180C0D20380BCD2D580BAD206E5
:1008F00080B47F2012083B2003077401B5450040F7
:10090000F1120800FF12083B020874D209D20780D6
:1009100095120800FB120800FA120800F94A4B7001
:1009200006791D7A0B7BFF20032E545602A7E00A9
:100930008E82758300120C5260060EE654670FD2
```

```
:10094000C2D5EBC0E0EAC0E0E9C0E0EE120A93D005
:10095000E0F9D0E0FAD0E0FB120C91FF60AAEBC006
:10096000E0EAC0E0E9C0E012083BD0E02401F9D0A1
:1009700003400FAD0E0FB5E460460DCD546D980DF
:10098000877BFF7A0A798FD203809C791080027965
:1009900008C207C2098008D2D5790A8004790AC240
:1009A000D5E546047002F546E4AFDFEFF120800A4
:1009B000FC7B08200213120800FD7B1030010A1294
:1009C0000800FE120800FF7B20CE3382592D55040
:1009D00013C3E43001069FFFE49FEFE42002039D0
:1009E000FDE49CFCE4CBFC8202EC700CCFCCECC85
:1009F000E824F8F87F38017C3EF33FFEE33FEED11
:100A000033FDEC33FCB33FB994002FB0FD8E9EBF1
:100A1000300205F8D0E0C448B202C0E00AEC4D4E06
:100A20004F78207B007C2EAB546004BCC0E0129F
:100A30000A95D0F0D0E0200204C4C0E0C4B202C0E5
:100A4000F0120824D0F0D5F0E0B020874120CC0997
:100A5000115309B5808E24C08DE42098F4099761
:0F0A60004099743098F743099D550981460981C3
:100A6F00450981470B3D5008E62D08EA2E090D2B4D
:100A7F0008E23090D0E0200262A08A6400000905B
:100A8F003F3F3F00790A2D5200414300609B91060
:100A9F00020404B9080104A2D52007025001042062
:100AAF0003689203B545005034C0E07F203004192D
:100ABF007F30A20372077206500F120AEC203C2F4
:100ACF0007C206C2097F30800F300603E9C0E0126B
:100ADF00083300603D0E0F9D0E0B545CC3006171F
:100AEF007F30B9100C12083B7F583005077F788094
:100AFF0003B9080312083B3003057F2D02083B7F23
:100BF00202009F87F2B2007F322920380CF286E35
:100B1F00756C62900D2021208003002F8C20278B
:100B2F00453D050108F60208A62D50434958120842
:100B3F00002403B405004001E4900B389312082CF5
:0D0B4F00743A12082CD20475450402098B7B
:100F5500E4FFFE120C91600C0FEF7001E0E9E970B1
:050F6500F20A80EF22FA
:100C9100BB010689828A83E022500E722BBFE0261
:090CA100E32289828A83E4932294
:100CAA00BB010689828A83F0225002F722BBFE0129
:020CBA00F32E23
:100CBC00F0A8080E80E9F25F0F618E6CA3AF62239
:100CC00D083D082F8E4937012740193700D0A3A3B7
:100CDC0093F8740193F5828883E4737402936860CB
:060CEC00EFA3A380DF26
:100EE200E4B40A07740D120EED740A309811A89926
:100EF200B8130CC2983098FD8A99C298B811F63070
:070F020099FDC299F5992247
:00000001FF
```

HELLO.C

```
#include <stdio.h>
#include <string.h>
#include <Regx52.h>
```

```
void ready(void);
void command(int);
void display(char *);
void main (void)
{
    command(0x3c);
    command(0x0c);
    command(0x06);
    command(0x01);
    command(0x80);
    display("Hello! How R U?");
    while(1);
}
```

```
void command(int a)
{
```

```
void ready(void);
ready();
P1 = a;
P3_2 = 0x00;
P3_3 = 0x00;
P3_4 = 0x01;
P3_4 = 0x00;
}
```

```
void display(char *str)
{
    unsigned int i;
    for(i = 0; i <= strlen(str)-1; ++ i)
    {
        if(i == 8)
            command(0xc0);
        if(i == 16)
            command(0x80);
        ready();
        P1 = str[i];
    }
}
```

```
P3_2 = 0x01;
P3_3 = 0x00;
P3_4 = 0x01;
P3_4 = 0x00;
```

```
}
}
}
void ready(void)
{
    P3_4 = 0x00;
    P1 = 0xff;
    P3_2 = 0x00;
    P3_3 = 0x01;
    while(P1_7)
    {
        P3_4 = 0x00;
        P3_4 = 0x01;
    }
    P3_4 = 0x00;
}
```

HELLO.HEX

```
:0300000002092AC8
:0C092A00787FE4F6D8FD75810E0208AE5F
:1009190048656C6C6F212048677270522055203F25
:0109290000CD
:1008AE007F3C7E001209067F0C7E001209067F0631
:1008BE007E001209067F017E001209067F807E00EF
:0E08CE001209067BFF7A09791912080080FED4
:100906008E0D8F0E1208DC850E90C2B2C2B3D2B421
:03091600C2B42246
:060800008B088A09890A39
```

```
:10080600E4F50BF50CAB08A0A9A90A1208F1EF24C6
:10081600FFFFFEE34FFED3E50C9F50B9E5042E54D
:100826000C6408450B70067C0FE12090650C64D1
:1008360010450B70067F80FE1209061208DCAB0815
:10084600A0A9A90A850C82850B83120868F590D23D
:10085600B2C2B3D2B4C2B4050CE50C70A8050B80C5
:01086600A4ED
:01086700226E
:1008DC00C2B47590FFC2B2D2B3309706C2B4D2B4D0
:0508EC0080F7C2B422F8
```

```
:10086800BB010CE58229F582E5833AF583E0225045
:1008780006E92582F8E622BBFE06E92582F8E2228F
:0D088800E58229F582E5833AF583E49322A9
:1008F100E4FFFE120895600C0FEF7001E0E9E9701C
:0509010020A80EF226C
:10089500BB010689828A83E022500E722BBFE0261
:0908A500E32289828A83E4932294
:00000001FF
```