PRL-TN-2011-100

Technical Note

# Microcontroller based 1-Wire temperature sensor network

by
**T.A. Rajesh**

भौतिक अनुसंधान प्रयोगशाला, अहमदाबाद
Physical Research Laboratory, Ahmedabad

PRL-TN-2011-100

Technical Note

# Microcontroller based 1-Wire temperature sensor network

by

**T.A. Rajesh**

भौतिक अनुसंधान प्रयोगशाला, अहमदाबाद

Physical Research Laboratory, Ahmedabad

*Published by*
*The Dean's office, PRL.*

# Contents

# Microcontroller based 1-Wire temperature sensor network

**T.A. Rajesh**
Physical Research Laboratory, Ahmedabad
*(rajeshta@prl.res.in)*

***Abstract:*** *The 1-Wire DS18B20 digital temperature sensor network is built using 8-bit microcontroller ATmega32 from AVR. The microcontroller application program is complied in Bascom basic cross compiler, PonyProg2000 is used to program and configure the microcontroller and Visual basic (VB6.0) is used for the development of the graphical user interface (GUI). The microcontroller is programmed to detect and read maximum 20 DS18B20sensors. As an application case study the system is used in the Aerosol monitoring laboratory to read and log temperature at six points of the aerosol sampling line.*

## 1. Introduction

Temperature is one of the important parameters in any process control. Temperature sensors come in a wide variety and they all measure temperature by sensing some change in a physical characteristic. There are many types of temperature sensors that will use various technologies and have different shapes. These sensors are used in many fields in the industry and in household equipment. When selecting which type of sensor to uses there are various considerations that must be made depending specifically on the application. When selecting a sensor it is important to consider the temperature range, the required accuracy and response time as these will vary with different measuring methods.

The various types of temperature sensors are thermocouples, resistive temperature devices (RTDs), thermistors, Infra-red and thermal radiation sensors, bimetallic devices, Integrated circuit temperature sensor (LM35), solid state temperature sensor (AD590), digital temperature sensors (DS1820, LM92, LM32) etc [1]. A conventional temperature sensor gives a change in temperature as a change in resistance, current or voltage. A digital temperature sensor gives the measured temperature in a serial data stream. It comes with the following interfaces; Inter-Integrated Circuit ($I^2C$) interface, Serial Peripheral Interface (SPI), 1-Wire interface and Pulse Width Modulation (PWM) [2].

This note discusses, in details, about the work carried out by the author on a 1-Wire digital temperature sensor DS18B20 and its networking with 8-bit microcontroller ATmega32. The DS18B20 is a low cost, low power and accurate sensor, ideal for portable and distributed measurement applications. The microcontroller interface allows to integrate a large number of sensors – theoretically the number is unlimited, but practically depends on the microcontroller and its memory resources. The range is 750 meters without the use of any repeaters; topology is distributed i.e. the sensors are attached in parallel with each other.

## 2. 1-Wire Bus system

The 1-Wire bus system consists of a master controller which is connected to one or more slave devices [3]. The master communicates with slave devices using the 1-Wire protocol developed by Dallas Semiconductor, receiving and sending signals over a single data line with respect to ground. It has by definition only a single data line for communication. It uses conventional CMOS/TTL logic levels (maximum 0.8V for logic "zero" and minimum 2.2V for logic "one"). Both master and slaves are configured as transceivers allowing bit sequential data to flow in either direction in half duplex mode. It synchronizes the slave devices to the master. The master initiates and controls all communications on the 1-Wire bus.

Each 1-Wire slave has a unique 64-Bit serial code stored in an on-board ROM that acts as its node address. The 1-Wire device interfaces to the data line via an open drain or 3-state port. This allows each device to release the data line when it is not transmitting data so the bus is available for use by another device. It requires an external pull-up resistor of approximately 5kΩ; thus, the idle state for the 1-Wire bus is high. All communications on the 1-Wire bus begin with an initialization sequence.

## 3. 1-wire temperature sensor: DS18B20

The DS18B20 from Dallas Semiconductors is an integrated solid state temperature sensor, an "analog-to-digital" convertor with 1-Wire network interface [4]. It communicates over a 1-Wire bus with a microprocessor or microcontroller. It measures temperature from -55°C to +125°C. The resolution of the sensor is user configurable to 9, 10, 11 or 12 bits corresponding to temperature increments of 0.5°C, 0.25°C, 0.125°C and 0.0625°C, respectively. The default resolution at the power on state is 12 bit and the output temperature data is calibrated in degrees centigrade.

Figure 1 shows a block diagram of the DS18B20 temperature sensor. It measures temperature through the use of an on board proprietary temperature measurement technique. It counts the number of clock cycles that an oscillator with a low temperature coefficient goes through during a gate period determined by a high temperature coefficient oscillator as illustrated in Figure 2. The slope accumulator is used to compensate the non-linear behavior of the oscillators over temperature.

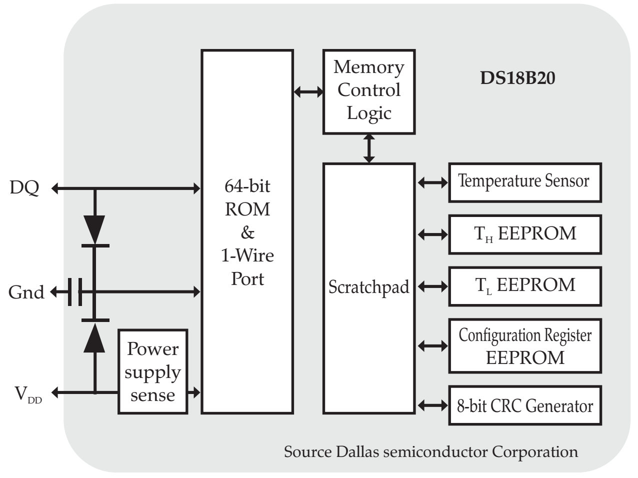
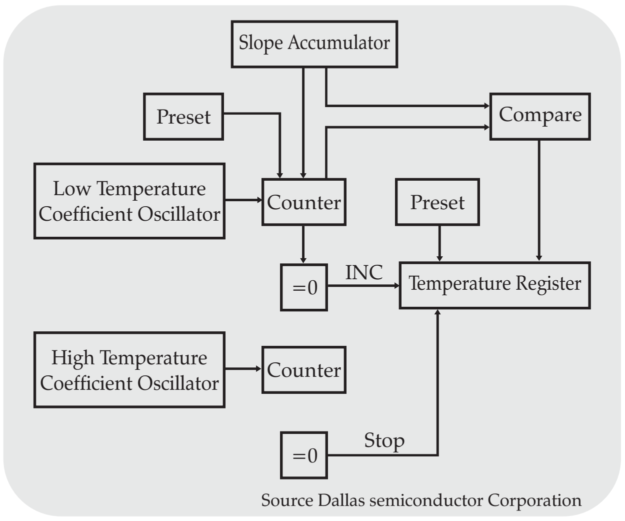


*Fig.1: DS18B20 Block Diagram*



Slope Accumulator
Preset
Compare
Low Temperature Coefficient Oscillator
Counter
Preset
=0
INC
Temperature Register
High Temperature Coefficient Oscillator
Counter
=0
Stop
Source Dallas semiconductor Corporation

*Fig.2: DS18B20 temperature measuring circuitry*

| 0 | R1 | R0 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |

*Fig.3: Configuration register*

| R1 | R0 | Resolution bits | Decimal steps | Conversion time (ms) |
|---|---|---|---|---|
| 0 | 0 | 9 | 0.5 | 93.75 |
| 0 | 1 | 10 | 0.25 | 187.5 |
| 1 | 0 | 11 | 0.125 | 375 |
| 1 | 1 | 12 | 0.0625 | 750 |

*Table 2: DS18B20 resolution configuration*

Each DS18B20 has a unique 64 bit serial code, which allows multiple DS18B20s to communicate on the same 1-Wire bus. The 64 bit ROM stores the serial code. Table 1 shows the scratchpad memory map. The memory consists of a static random access memory (SRAM) scratchpad and a nonvolatile electrically erasable programmable read only memory (EEPROM) storage for upper and lower alarm trigger registers ($T_H$ and $T_L$) and the 1 byte configuration register. Byte 0 and byte 1 contains the least significant byte (LSB) and most significant byte (MSB) of the temperature register, respectively which stores the temperature sensor's digital output. The byte 2 and byte 3 provides access to the $T_H$ and $T_L$ registers. Byte 4 contains the 1 byte configuration register data; it allows setting the resolution of the temperature to digital conversion to 9, 10, 11 or 12 bits. Bytes 5, 6 and 7 are reserved for internal use by the device. The byte 8 contains the cyclic redundancy check (CRC) code for the bytes 0 through 7 of the scratchpad memory and is used for the data validation when data is read from the DS18B20. Data is written to bytes 2, 3 and 4 of the scratchpad using the ROM command. Bit 7 and bits 0 to 4 in the configuration register are reserved for internal use by the device as illustrated in Figure 3. Bit 6 and bit 5 (R0 and R1) are user programmable in order to set the conversion resolution of the DS18B20 as shown in Table 2.

| DS18B20 Memory map Scratchpad | | |
|---|---|---|
| **Byte** | **Content** | **Startup value** |
| 0 | Temperature LSB | 0x5005 (+85$^0$C) |
| 1 | Temperature MSB | |
| 2 | TH Register / User Byte 1 | Content stored in EEPROM |
| 3 | TL Register / User Byte 2 | |
| 4 | Configuration register | |
| 5 | Reserved (FFh) | – |
| 6 | Reserved (0Ch) | – |
| 7 | Reserved (10h) | – |
| 8 | CRC | – |

*Table 1: DS18B20 memory map*

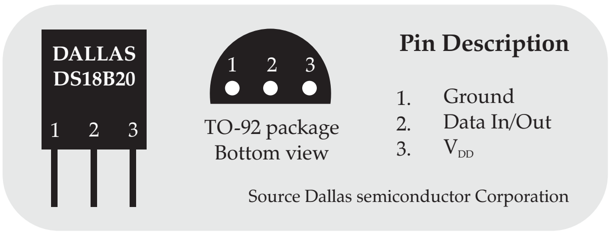


*Fig.4: DS18B20 Pin description*

Figure 4 shows the pin description of the DS18B20, TO-92 package temperature sensor. The DS18B20 can operate in two mode; 1) Parasite power mode and 2) external power supply mode. The parasite power mode allows the DS18B20 to operate without any local external power supply. This mode takes the power from the 1-Wire bus via the Data In/out pin when the bus is high. When the DS18B20 is performing temperature conversions or copying data from the scratchpad memory to EEPROM, the operating current goes up to 1.5mA. The significant voltage drop across the 4.7k pull-up resistor occurs due to this high current. Hence a strong pull-up on the 1-Wire bus is accomplished by using a MOSFET as shown in Figure 5. The advantages of parasite power modes are; 1) no local power source is required for remote sensing of temperature and 2) the ROM can be read in absence of normal power. The DS18B20 can also be powered by an external power supply as illustrated in Figure 6. The transaction sequence for accessing the DS18B20 is initialization, issue of ROM commands and issue of DS18B20 function commands. Initially a reset pulse is transmitted by the master followed by presence pulse(s) transmitted by the slave(s). The presence pulse lets the master know that slave devices are on the bus and are ready to operate. The master then issues a ROM command, as shown in Table 3. There are five ROM commands and each is 8-bit long. The SEARCH ROM command allows determining the number and types of devices on the bus. The master should issue the appropriate ROM command before issuing a DS18B20 function command. There are six Ds18B20 function commands and these commands allow the master to read from and write to the DS18B20 scratchpad memory as shown in Table 4.

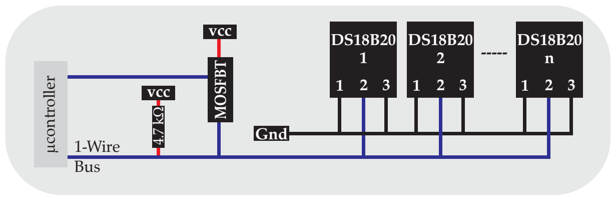


*Fig.5: Parasite powered DS18B20*

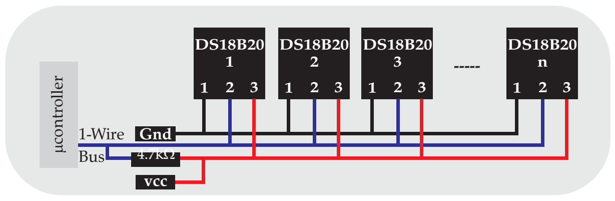


*Fig.6: External supply powered DS18B20*

| DS18B20 ROM Commands | | | |
|---|---|---|---|
| **No.** | **Command** | **Code** | **Description** |
| 1 | SEARCH ROM | F0h | This command is used to identify the ROM codes of the available slaves in the bus and also the total number of slaves devices. |
| 2 | READ ROM | 33h | This command is used when there is one slave on the bus. It has the same effect as SEARCH ROM. |
| 3 | MATCH ROM | 55h | This command is followed by a 64-bit ROM code sequence allows the bus master to address a specific slave on a multidrop or single-drop bus. |
| 4 | SKIP ROM | CCh | This command is used to address all devices on the bus simultaneously without sending out any ROM code. |
| 5 | ALARM SEARCH | ECh | This command allows to determine if any DS18B20s experienced an alarm condition during the most recent temperature conversion. |

*Table 3: DS18B20 ROM Commands*

| DS18B20 FUNCTION Commands | | | |
|---|---|---|---|
| **No.** | **Command** | **Code** | **Description** |
| 1 | CONVERT T | 44h | This command is used to start a temperature conversion, that is stored in the first two bytes of the scratchpad. |
| 2 | WRITE SCRATCHPAD | 4Eh | This command writes data into scratchpad bytes $T_H$, $T_L$ and configuration registers. |
| 3 | READ SCRATCHPAD | BEh | This command reads the entire scratchpad including the CRC byte. |
| 4 | COPY SCRATCHPAD | 48h | This command copies he content of $T_H$, $T_L$ and configuration registers from the scratchpad to EEPROM. |
| 5 | RECALL $E^2$ | B8h | This command recalls the alarm trigger values ($T_H$ and $T_L$) and configuration registers from EEPROM to scratchpad. |
| 6 | READ POWER SUPPLY | B4h | This command is used to determine if a slave is externally powered or uses parasite power. |

*Table 4: DS18B20 Function Commands*

## 4. Microcontroller – ATmega32

The ATmega32 is 8-bit low power CMOS microcontroller based on RISC architecture from ATMEL [5]. It uses Harvard architecture with separate memories and buses for data and program, in order to maximize the performance and parallelism. Figure 7 shows the block diagram of ATmega32 [6]. It has the following specifications

1. Program flash memory: 32K bytes
2. EEPROM: 1K bytes
3. SRAM: 2K bytes
4. 32 8-bit general purpose registers
5. Up to 16MIPS throughput at 16MHz
6. 32 General purpose I/O lines
7. 8 channels 10-bit Internal ADC
8. 2 Internal 8-bit Timer/Counters
9. 1 Internal 16-bit Timer/Counter
10. 1 Programmable serial USART
11. 4 PWM channels
12. 1 Internal Analog Comparator
13. Master/slave SPI serial interface
14. Serial interface: 1-Wire and $I^2C$

ATmega32 microcontroller board from Sunrom Technologies (http://www.sunrom.com) has been used in the present work, as shown in Figure 8. The technical features of the controller board are Crystal: 16.000MHz, On-board RS232 port for PC interface, RS232 port for In-circuit serial programmer, On-board ADC adjustable reference voltage, On-board reset switch and operates over a wide range of supply voltage: 7 to 15V. The microcontroller can be programmed and configured for the necessary read, write, lock, and fuse functions using the freeware serial device programmer PonyProg2000 software version 2.07a (http://ponyprog.sourceforge.net) and an ezAVR in-circuit serial programmer from SUNROM Technologies as shown in figure 8.

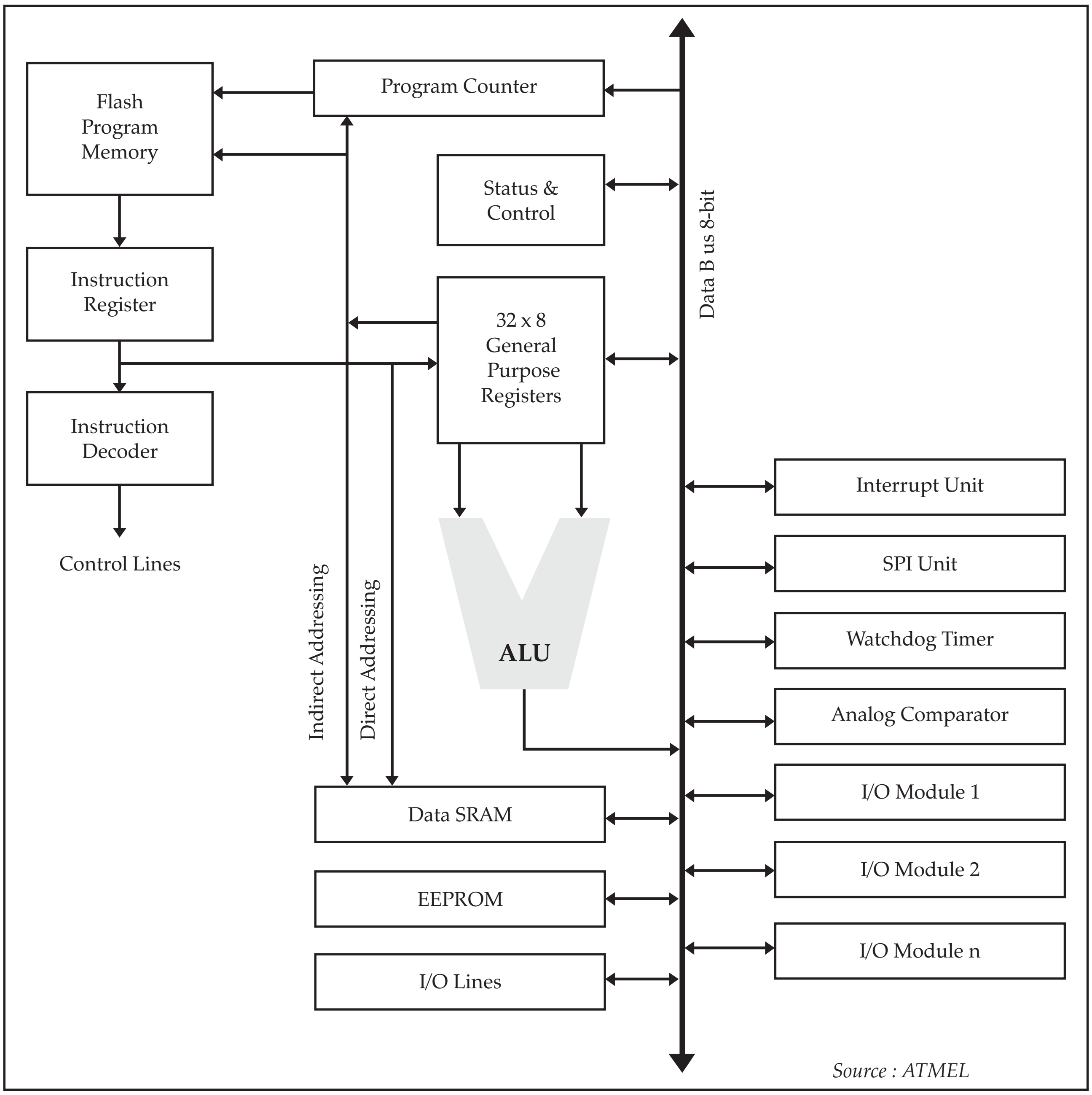


Fig.7: Block diagram of ATmega32 microcontroller

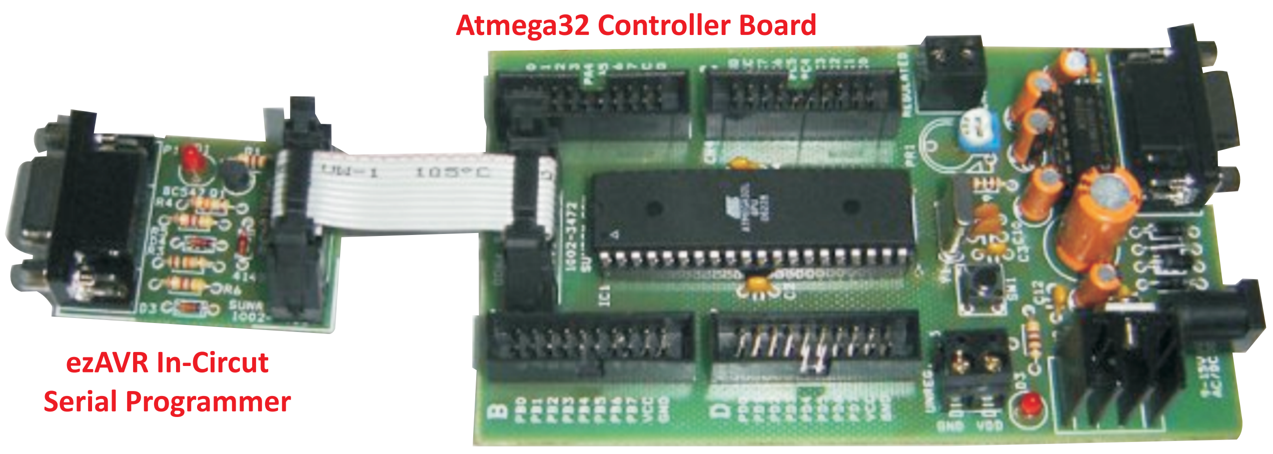


Fig.8: ATmega32 microcontroller board

## 5. System Hardware

The overall schematic design is shown in figure 9. The System design is based around ATmega32 microcontroller connected through Port B data bus to the 16 characters × 2 lines alphanumeric LCD display, PortC.0 data bit to the DS18B20 1-wire temperature sensor and RS232 serial port to the PC.

The system uses the smart LCD display to output visual information. Hitachi's LCD HD44780U module is inexpensive and easy to use. The LCD is using 7 IO lines for communication, the four data lines and three control ones. It uses 4-bit data bus for LCD display and the data lines are in tri-state when it is not enabled. The LCD is interfaced with microcontroller as shown in figure 10.

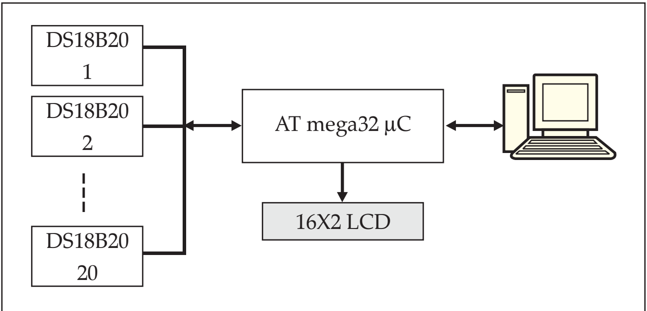


*Fig.9: Schematic diagram*

The DS18B20 sensors are configured in the externally powered mode with a pull up resistor of 4.7kΩ to +5VDC. It is connected to the microcontroller board PortC.0 bit, which is configured as 1-Wire interface. The present hardware uses six DS18B20 sensors, but the number can be as per user requirement. The data acquisition program running on a PC is connected to the microcontroller board via RS232 port.

## 6. System software - functions and processes

The BASCOM BASIC cross compiler is used to write the application program for the microcontroller ATmega32. The BASCOM-AVR (1.11.9.2) from MCS Electronics is the Windows BASIC Compiler for the AVR microcontroller family [7]. It compiles BASIC programs into a binary and/or hexadecimal file. These files can be programmed into the AVR microcontrollers that have unique Flash ROM. It supports special commands for LCD displays, I$^2$C interface, 1Wire interface, SPI, UART, matrix keyboard and PC keyboard. On the successful compilation of the BASCOM code it generates the respective hexadecimal file of the same.

The hexadecimal file of the program code is written into the ATmega32 microcontroller using PonyProg2000 software and ezAVR serial programmer. It is serial device programmer software with user friendly GUI framework. It supports I$^2$C Bus, Microwire, SPI eeprom, the Atmel AVR and Microchip PIC micro. It is designed to support the following AVR microcontrollers; ATmega103/161/ 163/ 323/ 128/8/ 16/ 64/ 32/ 162/ 169/ 8515/ 8535/ 44/ 88/ 168/ 164/ 324/ 644/ 640/ 1280/ 1281/ 2560, AT90can32/64, ATtiny12/ 15/ 26/ 2313/ 13/ 25/ 45/ 85/ 261/ 461/ 86 and AT89S8252/53 micro. It writes lock bits to protect the microcontroller from reading.

The front end microcontroller data acquisition program "1-Wire Temperature Sensor DAS" is written in Visual Basic 6.0 (VB). It is an event-driven programming language and integrated development environment (IDE) from Microsoft for its Component object programming model. It enables the rapid application development (RAD) of graphical user interface (GUI) applications [8 & 9].

The flowcharts of the microcontroller and GUI program are illustrated in appendix A. When the program is loaded and executed for the first time, it is mandatory to configure the application through 'Setup' menu. Under the 'Setup' menu the

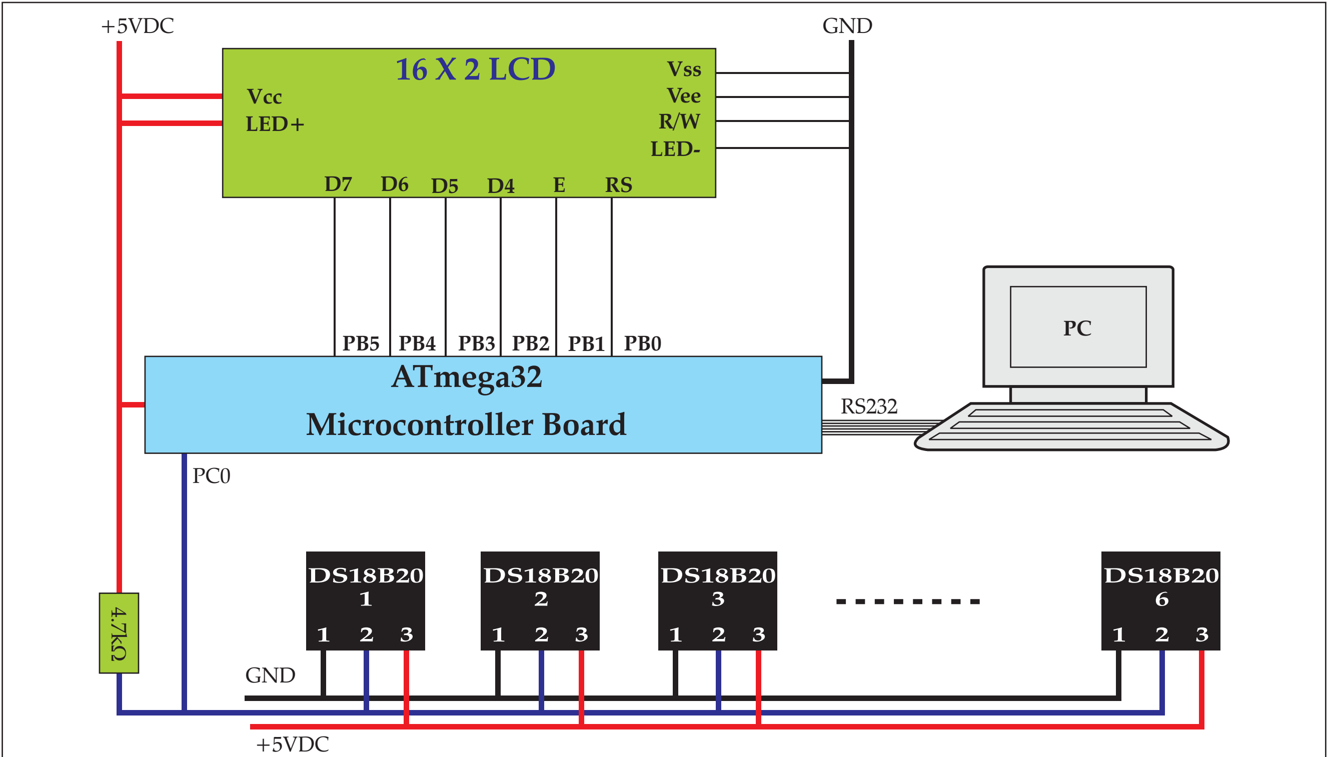


*Fig.10: Hardware circuit diagram*

user is supposed to enter the following field; Lab, Station, Sensor, Instantaneous Log, Directory, COM port and the COM Port setting. When 'Fetch Sensor Data' command button is executed the program will request to reset the microcontroller board. The program will scan for the number of DS18B20 sensors interfaced with the microcontroller board and its ROM ID and the result is displayed on the 'No. of Sensors' field and the ROM-ID field respectively as illustrated in appendix B. The ROM ID of each sensor is noted individually by running the program with the individual sensor. In the Label field the respective location of the sensors are labeled for the user reference. The setup data is saved on an ascii file setup.txt through the 'Save' command button. Quit the Setup frame using 'Back' command button. Start the data acquisition process through 'Start' menu. The screen displays the instantaneous temperature data from all the sensors and the data are logged in an ascii file with date and time stamped sensor temperature data. The data acquisition can be stopped through 'Stop' menu. The microcontroller and the GUI software are configured and programmed to handle maximum 20 DS18B20 sensors (as per our requirement). The GUI display screen adjusts its size as per the number of sensors used in the network.

## 7. Application – Temperature monitoring of the Aerosol sample line

In order to study the effect of aerosol optical (scattering) properties on the metrological parameters (temperature and humidity), it is required to measure the temperature and humidity along the sampling line and at the ambient.

The '1-Wire Temperature Sensor DAS' application software with the hardware is used in the Aerosol Monitoring Laboratory (670-b) for monitoring the temperature of the Aerosol sample line at multiple points. As illustrated in figure 11, we have used 6 DS18B20 sensors for monitoring the ambient temperature (S1), Stainless steel manifold outlet temperature (S2), mid of conductive sample line temperature (S3), room/lab temperature (S4), Nephelometer inlet temperature (S5) and Nephelometer sample temperature (S6). The six sensors are compared with the Hewlett Packard 5890 Gas chromatography temperature sensor and the correlation coefficient ($r^2$) is better than 0.9 for all the six sensors. The five minutes averaged multi points temperature diurnal plots for 14 July 2010 are shown in figure 12.

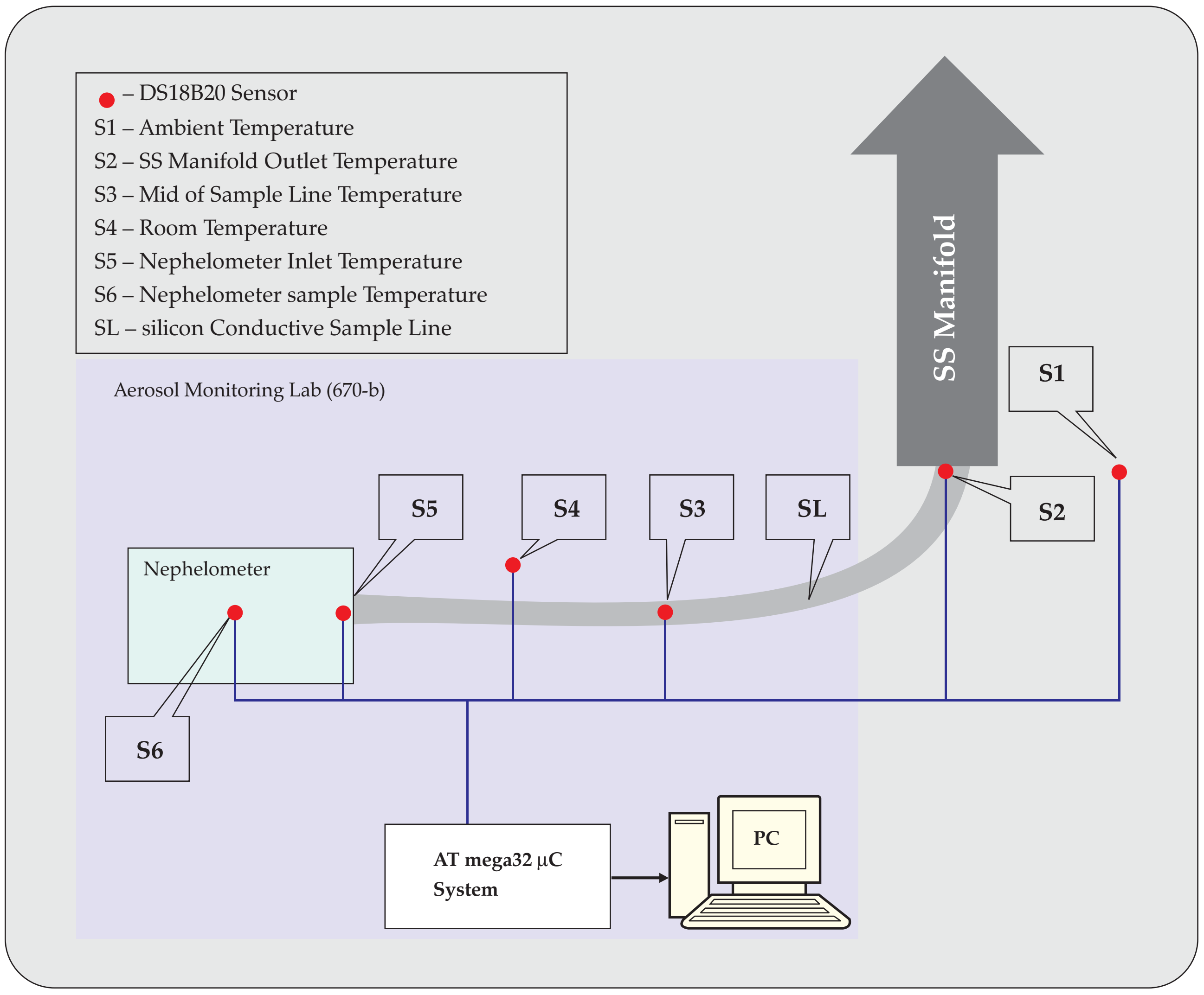

● – DS18B20 Sensor
S1 – Ambient Temperature
S2 – SS Manifold Outlet Temperature
S3 – Mid of Sample Line Temperature
S4 – Room Temperature
S5 – Nephelometer Inlet Temperature
S6 – Nephelometer sample Temperature
SL – silicon Conductive Sample Line
Aerosol Monitoring Lab (670-b)
SS Manifold
S1
S5
S4
S3
SL
S2
Nephelometer
S6
AT mega32 µC System
PC

*Fig.11: DS18B20 arrangement in aerosol sample*

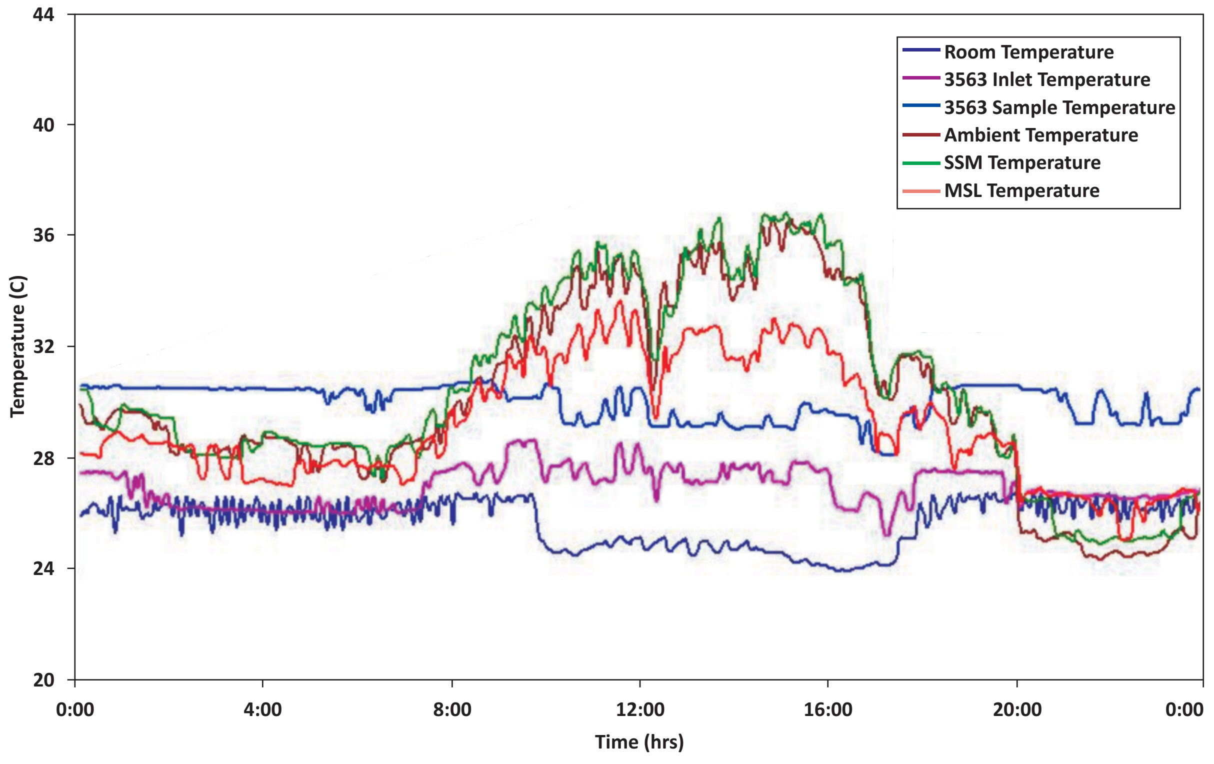


*Fig.12: Diurnal aerosol sample line temperature variation*

## 8. Summary

The present note discusses about the ATmega32 8-bit microcontroller based 1-Wire temperature sensor DS18B20 network. All the sensors, used for the multipoint temperature, are connected with microcontroller through 1-Wire interface. The system conceptually developed here, allows investigating some problems related to the distributed temperature measurements applications and the possible solutions. The Microcontroller and front end GUI software are programmed to work with maximum 20 DS18B20 sensors. The GUI is a user friendly windows application package to be used with the microcontroller unit described in this note. The program in the distribution form is available with the author for any interested user.

## 9. Acknowledgements

I acknowledge Mr. Pranav Adyarau of Planetry GeoSciences Division (PGSD) PRL for providing BASCOM-AVR Compiler. I am thankful to the members of the Space and Atmospheric Sciences Division of PRL who helped in this work.

## 10. Bibliography

[1] Dogan Ibrahim, 2002, Microcontroller based temperature monitoring and control, Elsevier science and technology Books, USA.

[2] Goes F, 1996, Low-cost smart sensor interfacing, Delft University Press, Delft.

[3] www.1wire.org

[4] www.datasheetcatalog.org/datasheet/maxim/DS1820-DS1820S.pdf

[5] Dhananjay V Garade, 2001, Programming and Customizing the AVR Microcontroller, McGraw-Hill Publishers, USA.

[6] www.atmel.com/dyn/resources/prod_documents/doc2503.pdf

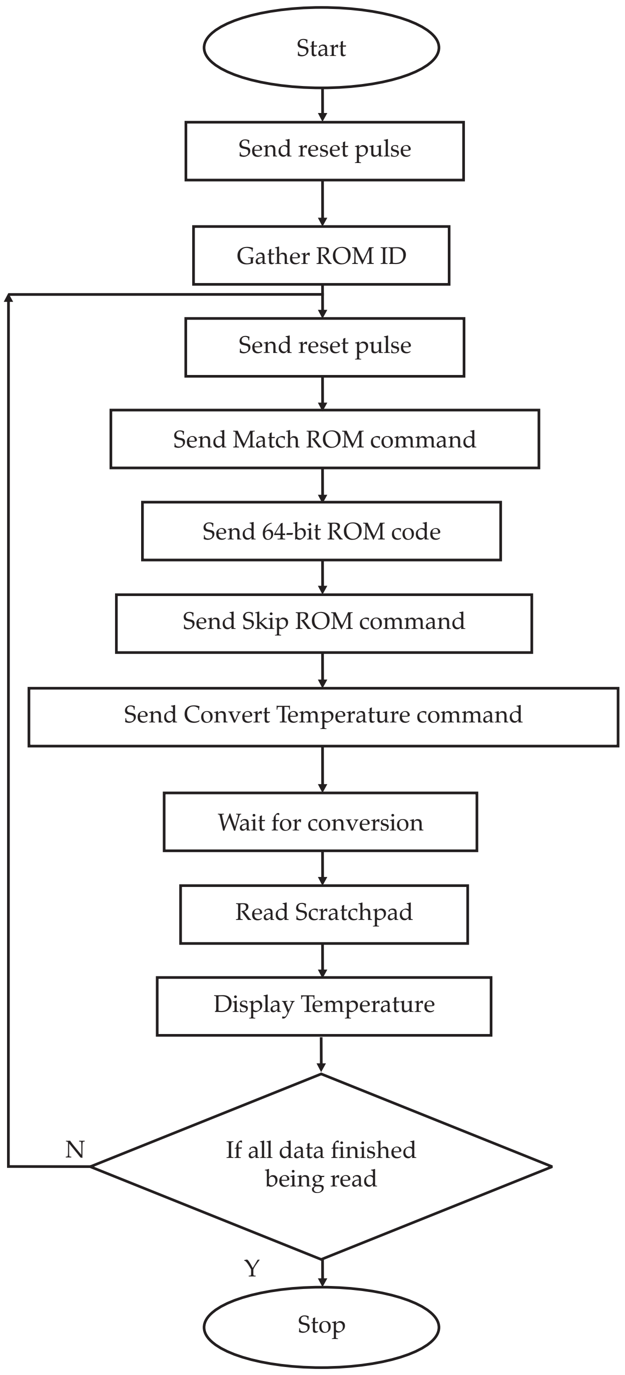
[7] Claus Kuhnel, 2001, BASCOM Programming of Microcontrollers with Ease, Universal Publishers, USA.

[8] Curland Mathew, 2000, Advanced Visual Basic 6, Addison Welsey Publications, UK.

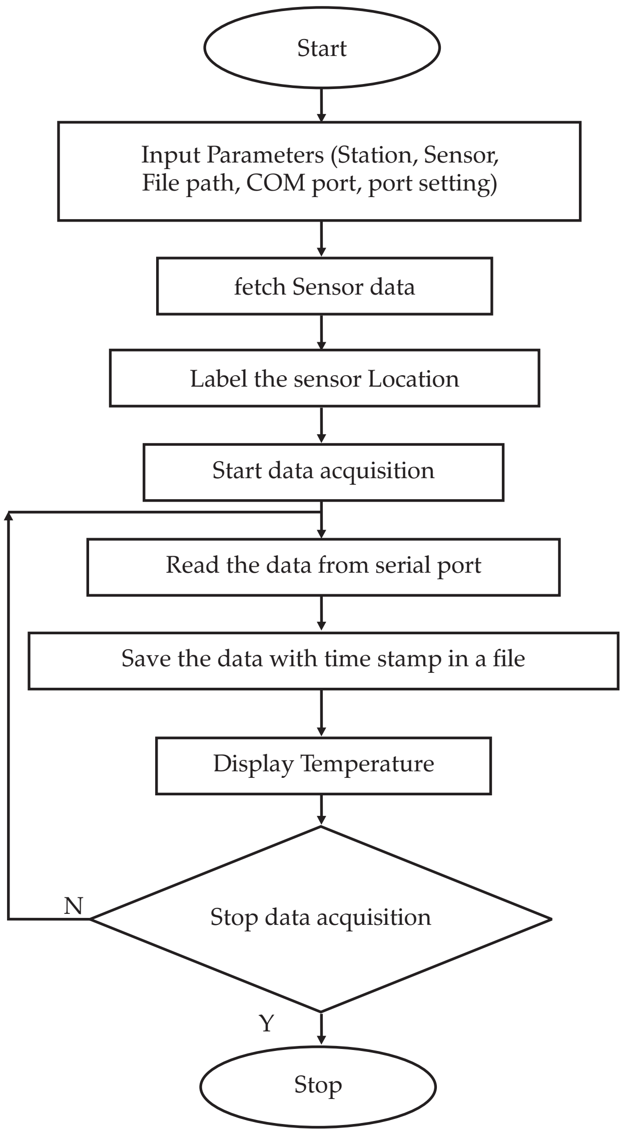
[9] Holzner Steven, 2005, Visual Basic Programming Black Book, Coriolis Publications, Chennai.

## Appendix: A – Flowchart

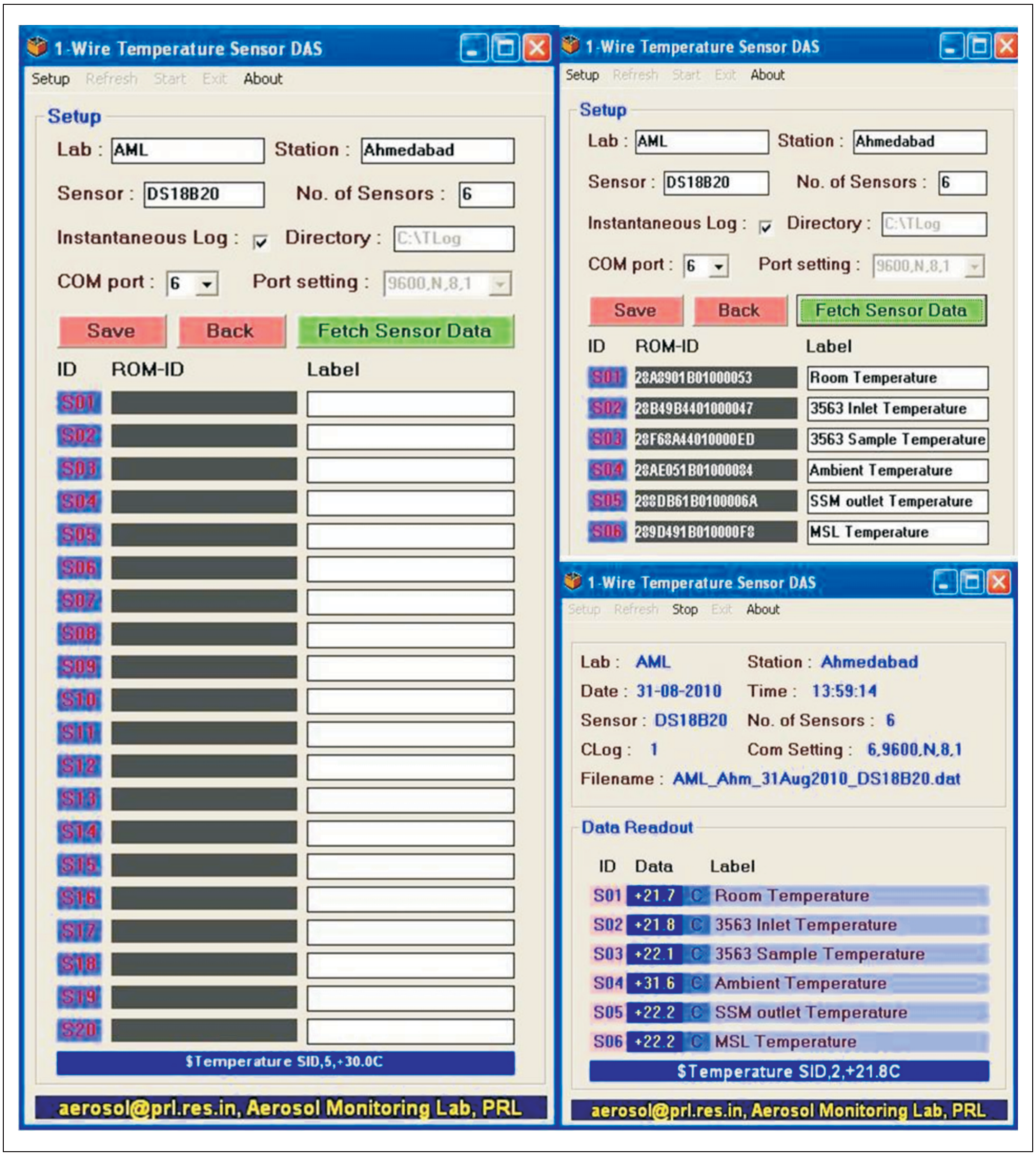
**Flowchart for Microcontroller – DS18B20 data acquisition system**



**Flowchart for GUI – Microcontroller data acquisition system**


Start
Input Parameters (Station, Sensor, File path, COM port, port setting)
fetch Sensor data
Label the sensor Location
Start data acquisition
Read the data from serial port
Save the data with time stamp in a file
Display Temperature
Stop data acquisition
N
Y
Stop

## Appendix: B – GUI Screen shots


1-Wire Temperature Sensor DAS
Setup Refresh Start Exit About
Save
Back
Fetch Sensor Data

Setup

Lab : AML Station : Ahmedabad

Sensor : DS18B20 No. of Sensors : 6

Instantaneous Log : ☑ Directory : C:\TLog

COM port : 6 Port setting : 9600,N,8,1

| ID | ROM-ID | Label |
|---|---|---|
| S01 | | |
| S02 | | |
| S03 | | |
| S04 | | |
| S05 | | |
| S06 | | |
| S07 | | |
| S08 | | |
| S09 | | |
| S10 | | |
| S11 | | |
| S12 | | |
| S13 | | |
| S14 | | |
| S15 | | |
| S16 | | |
| S17 | | |
| S18 | | |
| S19 | | |
| S20 | | |

$Temperature SID,5,+30.0C

aerosol@prl.res.in, Aerosol Monitoring Lab, PRL

1-Wire Temperature Sensor DAS
Setup Refresh Start Exit About
Save
Back
Fetch Sensor Data

Setup

Lab : AML Station : Ahmedabad

Sensor : DS18B20 No. of Sensors : 6

Instantaneous Log : ☑ Directory : C:\TLog

COM port : 6 Port setting : 9600,N,8,1

| ID | ROM-ID | Label |
|---|---|---|
| S01 | 28A8901B01000053 | Room Temperature |
| S02 | 28B49B4401000047 | 3563 Inlet Temperature |
| S03 | 28F68A44010000ED | 3563 Sample Temperature |
| S04 | 28AE051B01000084 | Ambient Temperature |
| S05 | 288DB61B0100006A | SSM outlet Temperature |
| S06 | 289D491B010000F8 | MSL Temperature |

Lab : AML Station : Ahmedabad

Date : 31-08-2010 Time : 13:59:14

Sensor : DS18B20 No. of Sensors : 6

CLog : 1 Com Setting : 6,9600,N,8,1

Filename : AML_Ahm_31Aug2010_DS18B20.dat

Data Readout

| ID | Data | | Label |
|---|---|---|---|
| S01 | +21.7 | C | Room Temperature |
| S02 | +21.8 | C | 3563 Inlet Temperature |
| S03 | +22.1 | C | 3563 Sample Temperature |
| S04 | +31.6 | C | Ambient Temperature |
| S05 | +22.2 | C | SSM outlet Temperature |
| S06 | +22.2 | C | MSL Temperature |

$Temperature SID,2,+21.8C

aerosol@prl.res.in, Aerosol Monitoring Lab, PRL

## *Appendix C: Bascom Code ' Program initialization*

```
$regfile = "m32def.dat"
$crystal = 16000000   ' crystal frequency
$baud = 9600

' Variable declaration
Declare Sub Convallt
Declare Sub Meas_to_cel(offset As Byte)
Declare Sub To_decicel
Declare Sub Monitor
Declare Sub Disp_temp(cnt As Byte , Offset As Byte)
' Up to 20 devices - each having an ' 8 byte ROMID
Const Max1wire = 20
Dim Dsid(160) As Byte

' Temperature measurement
Dim Cel As Integer
Dim Cel_frac_bit As Byte
Dim Subzero As Bit
Dim Decicel As Integer
Dim Sc(9) As Byte
Dim Cnt1wire As Byte

'Temp variables
Dim B As Byte
Dim C As Byte
Dim B1 As Byte
Dim B2 As Byte
Dim I As Byte
Dim W As Word

Const 750ms = 42098
Config Timer1 = Timer , Prescale = 256
On Timer1 Timer1_int

Dim Time1_count As Byte
Dim Time1_ok As Bit

Const Ds18b20_conf_reg = 4

' constant to convert the fraction
Const Ds18x20_fracconv = 625

' DS18x20 ROM ID
Const Ds18s20_id = &H10
Const Ds18b20_id = &H28

' DS18B20 ROM COMMANDS
Const Ds18x20_convert_t = &H44
Const Ds18x20_read = &HBE
Const Ds18x20_write = &H4E
Const Ds18x20_ee_write = &H48
Const Ds18x20_ee_recall = &HB8
Const Ds18x20_read_power_supply = &HB4

'LCD config
Config Lcdpin = Pin , Db4 = Portb.2 , Db5 = Portb.3 , Db6 =
Portb.4 , Db7 = Portb.5 , E = Portb.1 , Rs = Portb.0
Config Lcd = 16 * 2

' DS1820 on Port C.0
Config 1wire = Portc.0

Cursor Off
Cls
Lcd "Initialising..."
Print "<Initialising...>"

For I = 1 To 10
Lcd "<" ; I
Print "<" ; I
Wait 1
Next I
Cls

' Gather ROM ID for all 1-wire
' devices
Cnt1wire = 1wirecount()
Lcd Cnt1wire
Print "#No. of DS18B20," ; Cnt1wire
If Cnt1wire > Max1wire Then
 Cnt1wire = Max1wire
End If

B = 1
Dsid(b) = 1wsearchfirst()
For I = 1 To Cnt1wire
B = B + 8
Dsid(b) = 1wsearchnext()
Next

' Show the result on the bus
B1 = 1
B2 = 8
For I = 1 To Cnt1wire
Cls
If Dsid(b2) = Crc8(dsid(b1) , 7) Then
Lcd "CRC OK sensor " ; I
Print "#CRC OK sensor," ; "SID" ; "," ; I ; ",";
Waitms 500
Cls
Lcd "ROM ID "
Print "ROM ID,";

For B = B1 To B2
Lcd Hex(dsid(b))
Print Hex(dsid(b));
Next
Print
Else
Lcd "CRC BAD sensor " ; I
Print ">CRC BAD sensor " ; I
End If
Wait 1
B1 = B1 + 8
B2 = B2 + 8
Next

' Monitor temperature sensors
Time1_count = 0 : Timer1 = 750ms : Time1_ok = 0
Enable Timer1
Enable Interrupts
Start Timer1
Do
If Time1_ok = 1 Then
Stop Timer1
If Time1_count = 0 Then
Convallt
Elseif Time1_count = 1 Then
Monitor
End If
```

```
Reset Time1_ok
Start Timer1
End If
Loop

' Timer interrupt
Timer1_int:
Timer1 = 750ms
Set Time1_ok
Incr Time1_count
If Time1_count > 1 Then Time1_count = 0
Return
End

Sub Monitor
B = 1
For I = 1 To Cnt1wire
If Dsid(b) = Ds18s20_id Or Dsid(b) = Ds18b20_id Then
' Only process TEMP sensors
1wverify Dsid(b)
If Err = 1 Then
Lcd "18B20 not on bus"
Print ">DS18B20 not on bus"
Elseif Err = 0 Then
1wwrite Ds18x20_read
Sc(1) = 1wread(9)
If Sc(9) = Crc8(sc(1) , 8) Then
Call Disp_temp(i , B)
If I < Cnt1wire Then
Wait 1
End If
End If
End If
End If
B = B + 8
Next
End Sub

Sub Convallt
1wreset
' reset the bus
1wwrite &HCC
1wwrite Ds18x20_convert_t
End Sub

Sub Meas_to_cel(offset As Byte)
Dim Meas As Word
Meas = 0
Meas = Makeint(sc(1) , Sc(2))

' 18S20 is only 9bit upscale to 12bit
If Dsid(offset) = Ds18s20_id Then
Meas = Meas And &HFFFE
Shift Meas , Left , 3
B1 = 16 - Sc(6)
B1 = B1 - 4
Meas = Meas + B1
End If

W = Meas And &H8000
If W = &H8000 Then
Set Subzero
'positive
Meas = Meas Xor &HFFFF
Incr Meas
Else
Reset Subzero
End If

If Dsid(offset) = Ds18b20_id Then
B1 = Sc(ds18b20_conf_reg)

If B1.5 = 1 And B1.6 = 1 Then          ' 12
Elseif B1.6 = 1 Then                  ' 11
Meas = Meas And &HFFFE
Elseif B1.5 = 1 Then                  '10
Meas = Meas And &HFFFC
Else                               ' 9
Meas = Meas And &HFFF8
End If
End If

Cel = Meas
Shift Cel , Right , 4
Cel_frac_bit = Meas And &HF
End Sub

Sub To_decicel
Decicel = Cel_frac_bit * Ds18x20_fracconv
Decicel = Decicel / 1000
Cel = Cel * 10
Decicel = Decicel + Cel
If Subzero = 1 Then
Restore Rounding
For B1 = 1 To 8
Read B2
If Cel_frac_bit = B2 Then
Incr Decicel
Exit For
End If
Next
End If
End Sub

' Display the temperature
Sub Disp_temp(cnt As Byte , Offset As Byte)
Call Meas_to_cel(offset)
Call To_decicel
Cls
Lcd "Temp: " ; Cnt ; " "
Print "$Temperature SID," ; Cnt ; ",";
If Subzero = 1 Then
Lcd "-"
Print "-";
Else
Lcd "+"
Print "+";
End If
W = Decicel / 10
B1 = Decicel Mod 10
Lcd W ; "." ; B1
Print W ; "." ; B1;
Lcd " C"
Print "C"
End Sub
```

## Appendix D: VB 6.0 Source Code

```
' variable declaration
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim l As Integer
Dim sdata As String
Dim start_time As Long
Dim words() As String
Dim words1() As String
Dim hfile As Long
Dim hfile1 As Long
Dim fdata As String
Dim filename As String
Dim filename1 As String
Dim tmp(1 To 20) As Integer
Dim n(1 To 20) As Integer
Dim avg(1 To 20) As Double

Private Sub Combo1_Click()
For j = 1 To 20
Text1(j).Enabled = False
Next j
For j = 1 To Val(Trim(Combo1.Text)) - 1
Text1(j).Enabled = True
Next j
End Sub

Private Sub a1_Click() 'About
msg = "1-Wire Temperature Sensor Data Acquisition System" +
Chr$(13)
msg = msg + "Ver : 11.01A" + Chr$(13) + Chr$(13)
msg = msg + "Developed @ AML, PRL Ahmedabad"
MsgBox msg, vbInformation, "1-WTSDAS"
End Sub

Private Sub Command10_Click() ' back
Frame3.Visible = False
Frame1.Visible = True
Frame2.Visible = True
s2.Enabled = True
e1.Enabled = True
r1.Enabled = True
Call read_file
If l = 1 Then
MSComm1.PortOpen = False
End If
End Sub

' Save the configuration
Private Sub Command4_Click() 'save
hfile = FreeFile
Open "c:\TLog\setup.txt" For Output As hfile
Print #hfile, "0"; ","; Text1(26).Text; ","; Text1(21).Text; ",";
Text1(22).Text; ","; Text1(23).Text; ","; Text1(25).Text; ",";
Check1.Value; ","; Combo1.Text
For i = 1 To Val(Trim(Text1(23).Text))
Print #hfile, Trim(Str(i)); ","; Label3(21 + i).Caption; ",";
Label14(i - 1).Caption; ","; Text1(i - 1).Text
Next i
Close hfile
End Sub

' DS18B20 ROM ID fetch function
Private Sub Command9_Click() 'fetch
msg = "1-Wire Temperature Sensor Data Acquisition System" +
Chr$(13)
msg = msg + "Ver : 10.01A" + Chr$(13) + Chr$(13)
msg = msg + "Restart the Microcontroller unit"
MsgBox msg, vbInformation, "1-WTSDAS"
MSComm1.PortOpen = True
l = 1
End Sub

Private Sub e1_Click()
End
End Sub

' Load form
Private Sub Form_Load()
Call load_setup
End Sub

Private Sub load_setup()
For i = 1 To 20
tmp(i) = 0
Label14(i - 1).Caption = vbNullString
Text1(i - 1).Text = vbNullString
Label4(i - 1).Caption = vbNullString
Label6(i - 1).Caption = vbNullString
Next i
Text1(26).Text = vbNullString
Text1(21).Text = vbNullString
Text1(23).Text = vbNullString
Text1(22).Text = vbNullString
Combo1.Text = vbNullString
Label12(0).Caption = vbNullString
Label12(1).Caption = vbNullString
Label12(4).Caption = vbNullString
Label12(5).Caption = vbNullString
Label12(6).Caption = vbNullString
Label12(8).Caption = vbNullString
Label12(7).Caption = vbNullString
Frame1.Height = 8415 '1215 + Val(Trim(words1(4))) * 360
Label13.Top = 8040 ' + Val(Trim(words1(4))) * 360
Label7.Top = 10920 '3720 + Val(Trim(words1(4))) * 360
Form1.Height = 12045 '4845 + Val(Trim(words1(4))) * 360
k = 0
Frame3.Visible = False
i = 0
l = 0

s3.Visible = False
Combo2.Enabled = False
Text1(25).Enabled = False
Combo1.AddItem "1"
Combo1.AddItem "2"
Combo1.AddItem "3"
Combo1.AddItem "4"
Combo1.AddItem "5"
Combo1.AddItem "6"
Combo1.AddItem "7"
Combo1.AddItem "8"
Combo1.AddItem "9"
Combo1.AddItem "10"
Combo1.AddItem "11"
Combo1.AddItem "12"
```

```
Combo2.AddItem "1200,N,8,1"
Combo2.AddItem "2400,N,8,1"
Combo2.AddItem "4800,N,8,1"
Combo2.AddItem "9600,N,8,1"
Combo2.AddItem "19200,N,8,1"
Call read_file
Call config_mscomm
End Sub

' Read file subroutine
Private Sub read_file()
hfile = FreeFile
Open "c:\TLog\setup.txt" For Input As hfile
Do While Not EOF(hfile) 'Check for end of file.
Line Input #hfile, fdata 'Read line of data.
words1() = Split(fdata, ",")
Label12(0).Caption = words1(1)
Text1(26).Text = words1(1)
Label12(1).Caption = words1(2)
Text1(21).Text = words1(2)
Label12(4).Caption = words1(3)
Text1(22).Text = words1(3)
Label12(5).Caption = words1(4)
Text1(23).Text = words1(4)
Label12(8).Caption = words1(6)
Check1.Value = words1(6)
Label12(6).Caption = words1(7)
Combo1.Text = words1(7)
Label12(2).Caption = Format(Date, "DD-MM-YYYY")
For i = Val(Trim(words1(4))) + 1 To 20
Label3(i - 1).Visible = False
Label4(i - 1).Visible = False
Label5(i - 1).Visible = False
Label6(i - 1).Visible = False
Next i
Frame1.Height = 1215 + Val(Trim(words1(4))) * 360
Label13.Top = 840 + Val(Trim(words1(4))) * 360
Label7.Top = 3720 + Val(Trim(words1(4))) * 360
Form1.Height = 4845 + Val(Trim(words1(4))) * 360

Close hfile
End Sub

' interrupt serial comminucation
Private Sub MSComm1_OnComm()
filename = Trim(Label12(0).Caption) + "_" +
Mid(Trim(Label12(1).Caption), 1, 3) + "_" + Format(Date, "DD")
+ Format(Date, "MMM") + Format(Date, "YYYY") + "_" +
Trim(Label12(4).Caption) + "." + "dat"
sdata = vbNullString
start_time = Timer
Label12(3).Caption = Format(Time, "HH:MM:SS")
Label12(7).Caption = filename
Do
sdata = sdata & MSComm1.Input
On Error Resume Next
Loop Until InStr(sdata, vbCr) 'Or Timer - start_time > 0.5
If Len(Trim(sdata)) <> 0 Then
Call extract(Trim(sdata))
End If
Label15.Caption = Trim(sdata)
If Trim(Label12(8).Caption) = "1" Then
hfile = FreeFile
Open "C:\TLog\Data\" + filename For Append As hfile
Print #hfile, Format(Date, "DD-MMM-YYYY"); ","; Format(Time,
"HH:MM:SS"); ","; Mid(Trim(sdata), 1, 24); ","; "C"; ",";
Trim(Label6(Val(Trim(words(1))) - 1).Caption)
Close hfile
'Call print_clog
End If
End Sub

, Check time subroutine
Private Sub check_time()
hh = Val(Format(Time, "HH"))
mm = Val(Format(Time, "MM"))
ss = Val(Format(Time, "SS"))
tt = hh * 3600 + mm * 60 + ss
If tt Mod 10 = 0 Then
'Print Time
'Print Val(Trim(Label12(5).Caption))
   filename1 = Trim(Label12(0).Caption) + "_" +
Mid(Trim(Label12(1).Caption), 1, 3) + "_" + Format(Date, "DD")
+ Format(Date, "MMM") + Format(Date, "YYYY") + "_" +
Trim(Label12(4).Caption) + "." + "avg"

hfile1 = FreeFile
Open "C:\TLog\Data\" + filename1 For Append As hfile1
For i = 1 To Val(Trim(Label12(5).Caption))
avg(i) = tmp(i) / n(i)
Next i
For i = 1 To Val(Trim(Label12(5).Caption))
Print #hfile1, Format(Date, "DD-MMM-YYYY"); ",";
Format(Time, "HH:MM:SS"); ","; i; Format(avg(i), "##.0"), n(i)
Next i
For i = 1 To 20
avg(i) = 0
tmp(i) = 0
n(i) = 0
Next i
Close hfile1
End If
End Sub

' Print the data in file
Private Sub print_clog()
hfile = FreeFile
Open "C:\TLog\Data\" + filename For Append As hfile
Print #hfile, Format(Date, "DD-MMM-YYYY"); ",";
Format(Time, "HH:MM:SS"); ","; Mid(Trim(sdata), 1, 24); ","; "C";
","; Trim(Label6(Val(Trim(words(1))) - 1).Caption)
Close hfile
End Sub

' Extract the info from the string
Private Sub extract(a As String)
'Print Len(a)
words() = Split(a, ",")
Label13.Caption = a
Select Case Len(Trim(a))

Case 19
Text1(22).Text = Mid(Trim(words(0)), 9, 7)
Text1(23).Text = Mid(Trim(words(1)), 1, Len(Trim(words(1))) - 2)
For i = Val(Trim(Text1(23).Text)) + 1 To 20
Label3(21 + i).Visible = False
Label14(i - 1).Visible = False
Text1(i - 1).Visible = False
```

```
Label3(i - 1).Visible = False
Label4(i - 1).Visible = False
Label5(i - 1).Visible = False
Label6(i - 1).Visible = False
Next i

Case 27
Label4(Val(Trim(words(1))) - 1).Caption = Mid(Trim(words(2)), 1,
5)
tmp(Val(Trim(words(1)))) = tmp(Val(Trim(words(1)))) +
Val(Mid(Trim(words(2)), 1, 5))
n(Val(Trim(words(1)))) = n(Val(Trim(words(1)))) + 1

Case 28
Label4(Val(Trim(words(1))) - 1).Caption = Mid(Trim(words(2)), 1,
5)
tmp(Val(Trim(words(1)))) = tmp(Val(Trim(words(1)))) +
Val(Mid(Trim(words(2)), 1, 5))
n(Val(Trim(words(1)))) = n(Val(Trim(words(1)))) + 1

Case 45
Label14(Val(Trim(words(2))) - 1).Caption = Mid(Trim(words(4)),
1, 16)

Case 46
Label14(Val(Trim(words(2))) - 1).Caption = Mid(Trim(words(4)),
1, 16)

Case 47
Label14(Val(Trim(words(2))) - 1).Caption = Mid(Trim(words(4)),
1, 16)
End Select
End Sub

Private Sub config_mscomm()
With MSComm1
.CommPort = Val(Trim(Label12(6).Caption))
.Settings = "9600,N,8,1" 'Trim(setup_data3)
.InputLen = 0
.RThreshold = 1
.DTREnable = False
End With
End Sub

' refresh menu
Private Sub r1_Click() 'refresh
Call load_setup
End Sub

Private Sub s1_Click()
Frame3.Visible = True
Frame1.Visible = False
Frame2.Visible = False
r1.Enabled = False
s2.Enabled = False
e1.Enabled = False
Call read_file
Frame1.Height = 8415 '1215 + Val(Trim(words1(4))) * 360
Label13.Top = 8040 ' + Val(Trim(words1(4))) * 360
Label7.Top = 10920 '3720 + Val(Trim(words1(4))) * 360
Form1.Height = 12045 '4845 + Val(Trim(words1(4))) * 360
End Sub

Private Sub s2_Click()
r1.Enabled = False
s3.Visible = True
s2.Visible = False
s1.Enabled = False
e1.Enabled = False
MSComm1.PortOpen = True
End Sub

Private Sub s3_Click()
r1.Enabled = True
s2.Visible = True
s3.Visible = False
s1.Enabled = True
e1.Enabled = True
MSComm1.PortOpen = False
End Sub
```

PRL research
encompasses
the earth
the sun
immersed in the fields
and radiations
reaching from and to
infinity,
all that man's curiosity
and intellect can reveal

पीआरएल के
अनुसंधान क्षेत्र में
समविष्ट हैं
पृथ्वी एवं
सूर्य
जो निमीलित हैं
चुंबकीय क्षेत्र एवं विकिरण में
अनंत से अनंत तक
जिन्हे प्रकट कर सकती है
मानव की जिज्ञासा एवं विचारशक्ति