Design of a multichannel temperature data logger with SD card storage

TEMPERATURE is basically the measurement of how hot or cold something is, and it is one of the fundamental entities in any kind of process control. Temperature is also a measure of the average kinetic energy of particles in a system. Adding heat to a system increases the kinetic energy of its particles and hence an increase in the temperature of the system.

Temperature is easily measured using a thermometer. Nowadays, thermometers come in all shapes and sizes. In computer-based automated process control applications, where it may be required to control the temperature of a plant, it is necessary to interface the thermometer to the computer. This is usually done either by using analogue temperature sensor devices (such as thermistors, thermocouples, RTD devices etc) and then converting the outputs into digital format using A/D converters, or digital sensors (such as semiconductor sensor chips) are used with direct interface to the computer's.

A Selection of Temperature Data Loggers

There are many applications where one needs to store the variation of temperature with time continuously over a long period of time. In such applications it is common to use temperature data logging devices. There are many types of temperature data logging devices in the market with prices ranging from tens of dollars to over thousands of dollars. Examples include Squirrel OQ610 (**www.tempcon.co.uk**), which is a 6-channel temperature data logger using thermocouple probes. The device is battery-operated and up to 260,000 temperature readings can be stored in its non-volatile memory.

The TH-03 temperature data logger (**www.picotech.com**) is a 3-channel resistive logger with PC interface and ±0.3˚C accuracy. The device is connected to the serial port of a PC and sends the temperature readings to the PC where the data can be analysed using the supplied software.

The USB-5201 (**www.microdaq.com**) is an 8-channel temperature data logger with Compact Flash card for data storage. Temperature is measured using thermocouple probes. The device is supplied with a number of software packages for data analyses and it can also provide high/low alarms. In addition, 8-bits of digital I/O pins are provided. Although the device is very powerful its cost is over $600, typically only afforded by professional organisations.

OM-EL-USB (**www.omega.co.uk**) is a small, low-cost, USB type temperature data logger which plugs into the USB port of a PC to collect temperature data. The device has built-in memory that can store 16,000 temperature readings over a single channel. The logging interval can be changes from 12 seconds to 12 hours. Although the device is low-cost, its memory is limited.

Interested readers should note that other devices in the same family have higher memory capacities. For example OM-EL-USB4 (**www.winling.com.tw/OM-EL-USB.pdf**) is supplied with a memory that can store up to 32,000 temperature readings.

TL-500 (**www.audon.co.uk**) is a wireless temperature logging device with a USB based receiver and up to 50 remotely located transmitters with a range of 20-40 metres. The receiver has built-in memory and can operate standalone. The measurement interval is 45 seconds and data can be collected for over 110 days using one channel only. Wireless sensors are useful in remote applications which may be difficult to reach easily.

MCU-Based Design

This article describes the design of a microcontroller-based multichannel temperature data logger device with real-time clock chip and an SD card for the storage of very large amount of data. One of the nice features of the device is that it is based on the 1-Wire bus where a large number of temperature sensors can all be connected on the same bus.

A battery-operated RTC chip keeps the real-time clock information and the temperature is written on the SD card with real-time clock stamp. Each sensor on the bus has a unique 64-bit ID and a sensor responds only when its own ID is on the bus, thus enabling the data to be read selectively.

The RTC data, data logging interval and the ID of each sensor can be set by connecting the device to the serial port of a PC and running terminal emulation software on it. Once the device is configured it can be operated standalone to collect and save the temperature from a large number of sensors. The collected data can then be imported offline into a spreadsheet package such as Excel, and statistical analysis can be carried

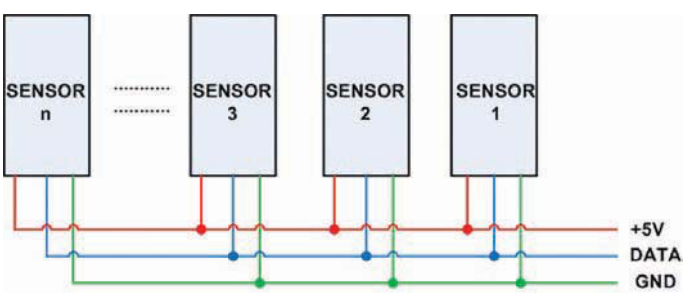


Figure 1: *1-Wire bus structure*

Professor Dr Dogan Ibrahim from the Department of Computer Engineering at the Near East University in Cyprus describes the design of a microcontroller-based multichannel temperature data logger device with SD card storage and real time clock interface

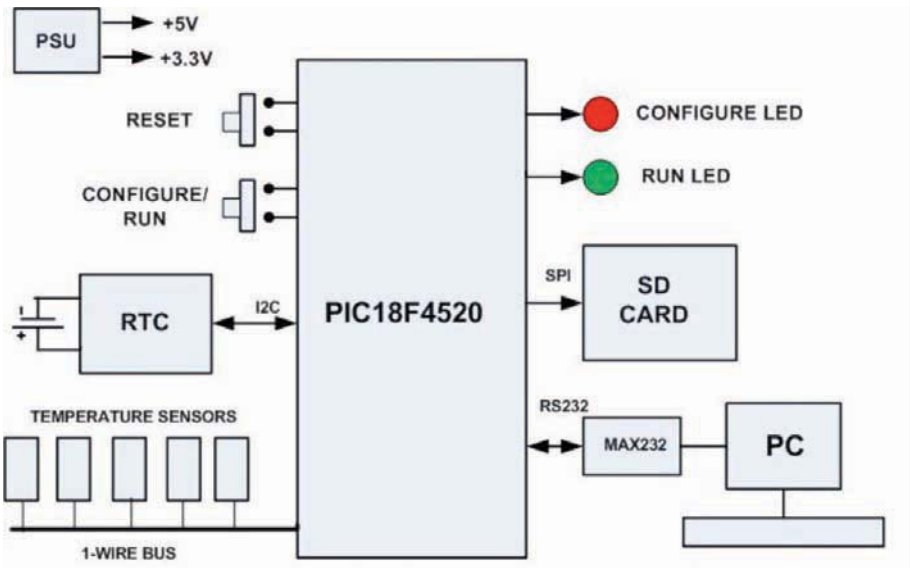


Figure 2: *Block diagram of the temperature data logger*

out, or graphs of the collected data can easily be drawn.

The device also sends out from its serial port the same data as written to the SD card. Thus, it is also possible to capture this data on a PC and perform online analysis of the data in real-time.

1-Wire Bus

1-Wire bus is a serial protocol using a single data line (plus ground reference) for communication. 1-Wire bus provides low-speed data rate and is similar in concept to I2C bus, but with lower data rates and longer range. It was developed by Dallas Semiconductors (**www.maxim-ic.com**) as a means of connecting a large number of sensors over a long distance to a single bus complete with integrated power source.

There are many 1-Wire supported devices, such as temperature sensors, humidity sensors, pressure sensors, switches, counters, real-time clock chips, EEPROM memories, A/D converters and so on.

Depending on the 1-Wire device used, there are two ways to power a device on the bus: parasitic mode and external power mode. In parasitic mode, a capacitor is charged in each device during the logic high state of the data line. The capacitor then provides power to the device during its normal operation and is re-charged whenever the data line is at logic high. In this configuration, the 1-Wire bus consists of only two wires, data and ground. If there are many devices on the bus then a current source may be required to feed the bus to charge the capacitor of each device. Parasitic mode is normally used in small applications with only a few devices on the bus.

In external power mode, a third wire is used to supply the 5V directly to each device and this mode is recommended in large 1-Wire networks.

Dallas recommends that unshielded CAT5 cables should be used as the 1-Wire bus. Shielded cables should not be used as the network may be upset as a result of the increased capacitance. For small network applications and short runs, standard telephone cables can be used for the bus. The design and layout of the cabling is also important to avoid any timing, loading, or reflection problems. In small networks, star or radial connections can be used where each device is connected to a central point with its own cable. In large networks, the recommended method is to use daisy chaining where a single continuous cable is used to loop from sensor to sensor, as shown in **Figure 1**.

In a typical application, a 1-Wire master initiates communication on the bus and controls the bus with one or more slave devices present on the bus. Each device on the bus has a unique, factory programmed and unalterable 64-bit identification number (ID). When there is communication on the bus, only the device with the referenced ID responds to the request. When there is only one slave device on the bus the ID can be "skipped" and the device will respond to all requests on the bus.

The 64-bit ID consists of 1-byte family code, 6 bytes serial number and 1 byte CRC code.

Each device type has a unique family code. For example, the family code of DS1820 temperature sensor devices is 0x10. The serial number uniquely identifies the device. The CRC byte is used to detect any errors in sending the ID.

The ID of a device is not supplied by the manufacturer, but it can be found easily when there is only one device on the bus, for example by writing a program to read and display the ID on an LCD. When there is more than one device on the bus it is much more difficult to find the ID of each device and special algorithms are needed to determine the ID of each device. Alternatively, a 1-Wire Evaluation Kit, such as the DS9090K (**www.maximic.com/quick_view2_cfm/qv_pk/4135**) can be used to find the ID of all devices on the bus. The evaluation kit is plugged into the USB port of a PC and is supplied with a Java program named OneWireViewer.

DS1820 1-Wire Temperature Sensor

Although there are several types of 1-Wire temperature sensors, the DS1820 chip is used in this design. DS1820 is a 3-pin sensor with the following specifications:

- 3-pin chip
- Direct 1-Wire interface with no external components
- Measures temperatures from -55˚C to +125˚C in 0.5˚C increments
- User definable, non-volatile temperature alarm settings
- 9-bit digital output
- 200ms conversion time.

In addition to the normal temperature measuring functions, the DS1820 (**www.systronix.com/Resource/ds1820.pdf**) can be used in thermostatic control applications in industrial systems, for example to set temperature limits with alarm outputs.

The output of DS1820 is in 16-bits sign extended two's complement format, but for practical applications only 8-bits define the temperature and the ninth bit is used to indicate the sign, as shown in **Table 1**. It is possible to obtain much more accurate temperature readings from the device by using the on-chip counter values, but this application is beyond the scope of this paper.

The conversion of the read digital value to actual temperature in degrees Celsius is straight forward; for positive readings, simply divide the value read by 2. For example, if the 9-bit reading is "0 00110010" (i.e. decimal 50), dividing by 2 gives the temperature as +25˚C.

For negative readings, take the complement of the reading, add 1 and then divide by 2. For example, if the 9-bit reading is "1 11001110", taking the complement and adding 1 gives "0 00110010", dividing by 2 we get the temperature as -25˚C.

TEMPERATURE	DIGITAL OUTPUT
	0 11111010
+25˚C	0 00110010
+0˚C	+125˚C
-25˚C	1 11001110
-55˚C	1 10010010

Table 1: *DS1820 output data format*

DS1820 contains an 8-byte scratchpad register where the converted temperature is stored in the first two bytes. A typical temperature conversion process consists of the following steps.

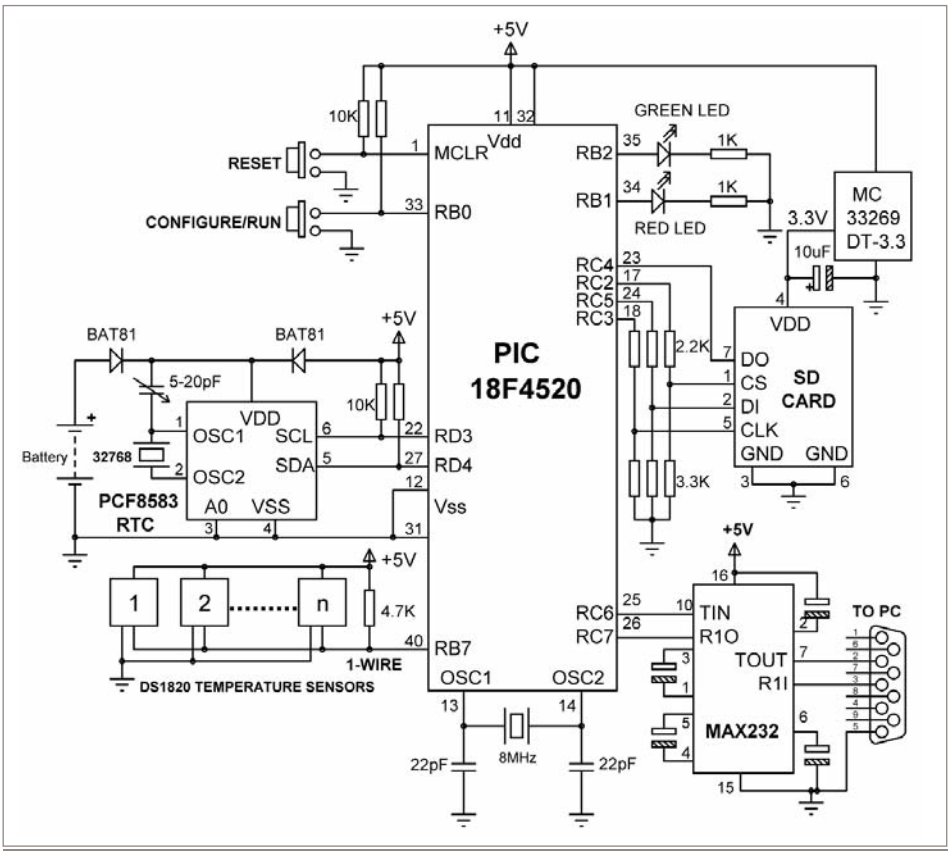


Figure 3: *Circuit diagram of the temperature data logger device*

For a single DS1820 on the bus:

- Send "SKIP ROM" command (0xCC).
- Send convert temperature command (0x44)
- Wait for the conversion (it can take up to 400ms)
- Send the read temperature command. First of all, send the "SKIP ROM" command (0x55), followed by the READ SCRATCHPAD command (0xBE).
- Read the low-byte and high-byte of the temperature from the scratchpad register.

For more than one DS1820 on the bus:

- Send the "MATCH ROM" command (0x55), followed by the 64-bit device ID to the required device.
- Send convert temperature command (0x44)
- Wait for the conversion (it can take up to 400ms)
- Send the read temperature command. First of all, send the "MATCH ROM" command (0x55), followed by the 64-bit device ID, and then the READ SCRATCHPAD command (0xBE).
- Read the low-byte and high-byte of the temperature from the scratchpad register.

Design of the Temperature Logger

The data logger is designed around a PIC18F4520 (**www.microchip.com**) type microcontroller. This is a highly powerful PIC18 series microcontroller having the following basic features:

- 40-pin package
- 32 K-byte flash program memory
- 1536 byte RAM memory
- 256 byte EEPROM memory
- 36 I/O pins
- 13 channel, 10-bit A/D converter
- Analogue comparators
- 4 Timer/counters
- Prioritized interrupts
- SPI, I2C, USART support
- Operation up to 40MHz
- 8 x 8 hardware multiplier
- C compiler optimized
- Sleep/Idle mode.

The block diagram of the temperature data logger device is shown in **Figure 2**. The user controls the device with two push-button switches: Reset and Configure/Run. Reset simply resets the device to a known state.

When in Configuration mode the date and time of the RTC and various logger parameters can be set through the RS232 port using a terminal emulation program on a PC.

In Run mode, the device reads the temperature from all the sensors attached to the 1-Wire bus and stores the date/time and the temperature in degrees Celsius in a file on

the SD card. The data is stored in a format compatible with the Excel spreadsheet.

The Hardware

The circuit diagram of the temperature data logger device is shown in **Figure 3**. At the centre of the circuit is a PIC18F4520 microcontroller, operated with an 8MHz crystal. A PCF8583 (**www.nxp.com**) type real-time clock (RTC) chip is connected to port pins RD3 and RD4 using pull-up resistors. The RTC chip is operated in I2C bus mode and a small battery is used to power the chip to retain the date and time information when external power is not available.

The SD card is connected to port pins RC2 through RC5 and is operated in SPI mode. A card holder is used to physically make connections to card pins. The voltage at output pins of the microcontroller is too high and can damage the input circuitry of the SD card. A pair of potential divider resistors (using 2.2K and 3.3K resistors) is used to lower the microcontroller output voltages to a level acceptable by the SD card inputs. The SD card is powered using a 3.3V regulated supply, obtained using a MC33269DT-3.3 (**www.volkin.com/MC33269DT-3.3.html**) type regulator.

A MAX232 (**www.maxim-ic.com/quick_view2_cfm/qv_pk/1798**) level converted chip is used to obtain RS232 compatible signals from the device. This serial port is used to set the RTC date and time and also configure various parameters used in the design. The date/time and the temperature data are sent to the serial port at the requested intervals, as well as being stored on the SD card.

The Operation

The device is operated by the Reset and Configure/Run buttons. There are two operating modes: Configuration mode and Run mode.

Configuration mode: This mode is entered by pressing and releasing the Reset button while holding down the Configure/Run button. In this mode, the RS232 port should be connected to a PC and a terminal emulation program (e.g. HyperTerm) should be used on the PC to establish communication with the logger device. The default communication parameters are: 2400 Baud, 8 bits and No parity bit.

In the Configuration mode the user is displayed a MENU with the options to either set the configuration, or to display the current configuration. **Figure 4** shows a typical display in Configuration mode. The following parameters can be set or displayed in this mode:

- RTC date and time
- Data logging interval (in minutes)
- Number of sensors used
- 64-bit ID of each sensor (must be entered as 16 hexadecimal numbers).

The red LED is turned on when the device is in the Configuration mode.

Run mode: The Run mode is entered after either pressing the Reset button, or applying power to the device. In this mode the device checks the presence of the SD card and, if the card is missing, both red and green LEDs are turned on to indicate an error condition and the logger is halted.

If, on the other hand, a card is detected in the card holder, the green LED is turned on to indicate that data logging has started. The device reads the temperature from all sensors on the 1-Wire bus at the requested intervals. The temperature, together with the current RTC date and time are stored in a file on the SD card, as well as sent to the RS232 port. Data logging stops when the Configure/Run button is kept pressed for about five seconds. The SD card should only be removed after the data logging is stopped.

A new file is created on the SD card each time a new logging session starts. The filename is unique and is made up of the current day, month, hours and minutes and is given the extension ".TXT". For example, if the logging session is started on the 12th of March, at 5 minutes past 10 o'clock, the filename will be "12031005.TXT".

Data is stored on the SD card (and also sent to the RS232 port) in the following format:
dd/mm/yy hh:mm:ss pp.p qq.q rr.r ss.s .....
where dd/mm/yy/ hh:mm:ss is the current date and time, and pp.p qq.q rr.r ss.s are the temperature read from the sensors.

Figure 5 shows the data stored on the SD card in a typical session. In this example there were only two sensors on the 1-Wire bus and the sensors were initially read on the 29/11/2008 at 14:29:01. The temperatures of both sensors were initially 22.5°C.

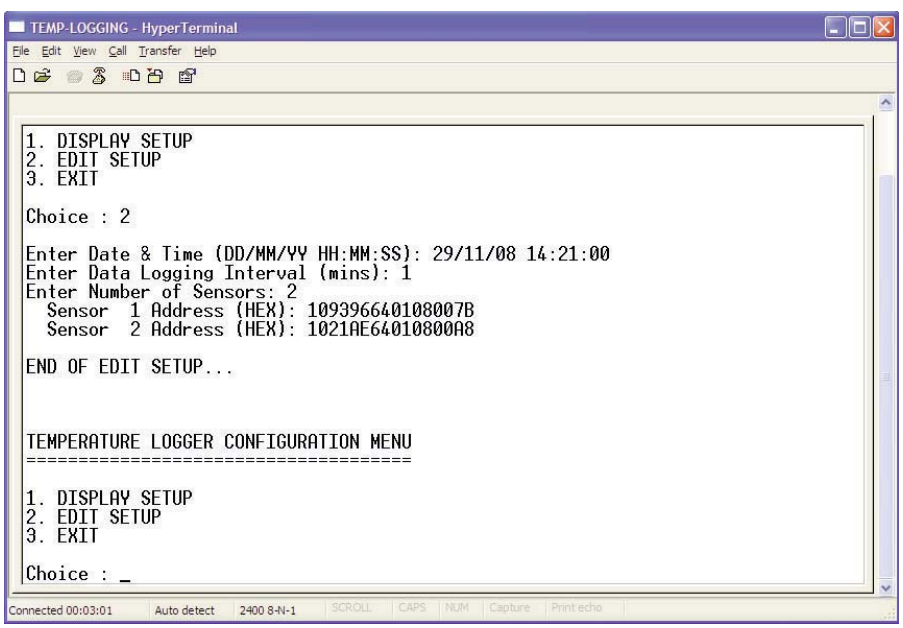


Figure 4: *The Configuration mode is entered by pressing and releasing the Reset button while holding down the Configure/Run button*

```
29/11/08 14:29:01,22.5,22.5
29/11/08 14:30:01,31.5,23.5
29/11/08 14:31:01,31.0,24.0
29/11/08 14:32:01,25.5,23.5
29/11/08 14:33:01,26.0,23.5
29/11/08 14:34:01,28.0,23.5
29/11/08 14:35:01,28.5,28.0
29/11/08 14:36:01,24.5,24.0
29/11/08 14:37:01,23.5,28.0
29/11/08 14:38:01,28.5,24.0
29/11/08 14:39:01,27.0,27.5
29/11/08 14:40:01,24.0,29.0
29/11/08 14:41:01,23.5,24.0
29/11/08 14:42:01,23.5,23.5
```

Figure 5: *Typical data stored on the SD card*

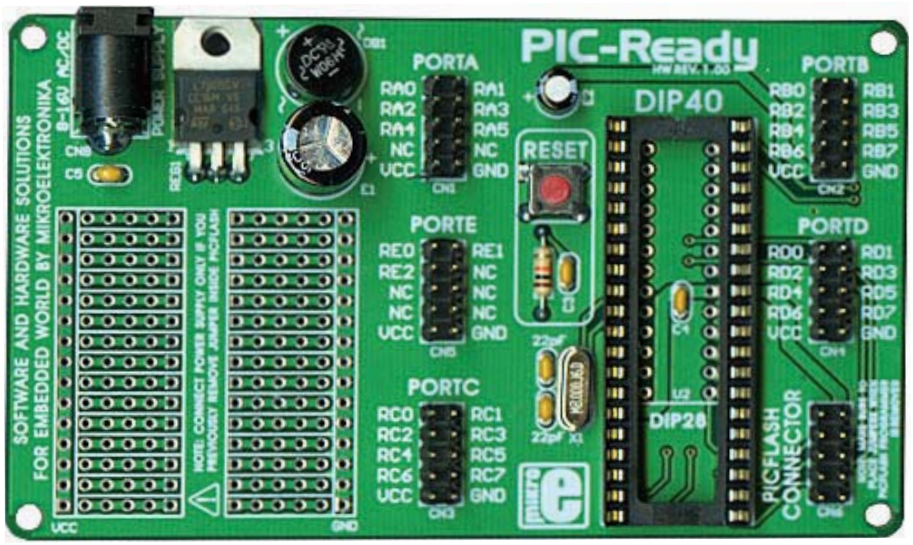


Figure 6: PIC-Ready development board

Construction

The hardware was constructed on a PIC-Ready (www.mikroe.com) development board (see **Figure 6**), manufactured by mikroElektronika. This is a low-cost ($24.00) powerful development board with the following features:

- Socket for 40-pin PIC microcontrollers
- 8MHz crystal
- +5V regulator (an external 9-12V power supply is required)
- In-circuit debugger (ICD) and PIC programmer interface
- Reset button
- Easy access to port pins via 10-way IDC connectors
- Plug-in compatible with most mikroElektronika development modules
- Small development solder area.

One of the nice things about PIC-Ready development board is the built-in ICD and the programmer. This requires the use of a PICFlash2 ICD device, manufactured by mikroElektronika. During program development one can easily insert breakpoints or single step a program with the help of the ICD. The ICD can also program most of the PIC chips on-board, without having to remove the chip from its socket. This feature is extremely useful during program development.

The RTC module is available from mikroElektronika as a plug-in module and this was connected to the PIC-Ready board.

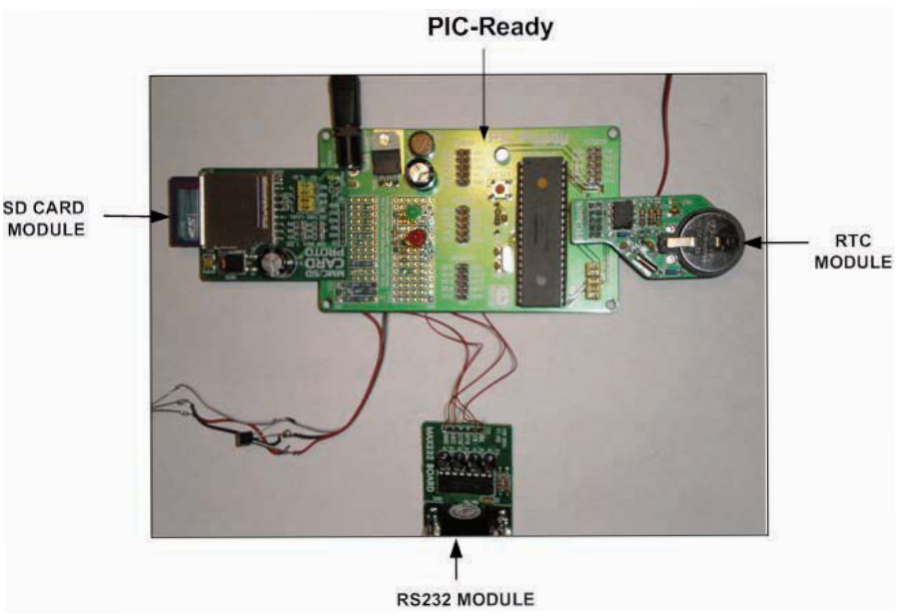


Figure 7: Data logger development system

Similarly, the SD card module and RS232 modules were connected to PIC-Ready by making the necessary wirings. Two external LEDs and a push-button switch were connected to the board. **Figure 7** shows the complete data logger development system, built around the PIC-Ready development board.

The Software

The software was developed using the mikroC compiler (www.mikroe.com) developed by mikroElektronika. This is a very powerful C language compiler developed for PIC microcontrollers and supports both PIC16 and PIC18 series. The compiler provides a very rich library of routines for developing applications for SD cards, Compact Flash cards, RS232/RS485 devices, USB, CAN bus, I2C, 1-Wire bus and much more.

Figure 8 shows operation of the software as a Program Description Language (PDL). The program is modular and consists of a number of functions and procedures for easy modification, update, or maintenance of the code. The following functions and procedures are used:

Initialize_RTC:	Initialises the I2C bus
Read_RTC:	Reads the RTC date and time and stores in array RTCData
Write_RTC:	Updates the RTC date and time from array USART
Send_Text:	Sends a text message to USART
Read_Usart:	Reads text from USART until carriage-return is detected
Send_Newline:	Sends a carriage-return and new-line to USART
Display_Setup:	Display configuration parameters on serial port
Edit_Setup:	Modifies the configuration parameters from serial port
MENU:	Displays MENU on serial port
Initialize_SD:	Initialises the SPI bus
Hex_Byte:	Converts a single character into decimal
Conv_Hex:	Converts hex bytes into decimal
Read_Temps:	Reads temperature data from sensors
Trim:	Removes spaces from a text
Format_Temp:	Formats the temperature data, writes to the SD card, and also sends to the RS232 port.

At the beginning of the program port directions are configured, USART is initialised and so is the I2C bus. The I2C bus is controlled using the Software I2C Library routines of mikroC compiler.

The program then checks to find out whether or not the Configuration/Run button is down, and if so, enters the configuration mode. In this mode, a MENU is displayed (see Figure 4) and the user has the option of either displaying the current configuration parameters or to modify these parameters.

The configuration parameters are stored in the EEPROM memory of the microcontroller so that they are not lost after the removal of the power. The layout of the configuration parameters in the EEPROM is shown in **Table 2**. Location 0 stores the year field of the RTC. Location 1 stores the data logging interval in minutes. Location 2 stores the number of DS1820 type sensors connected to the 1-Wire bus. The remaining locations of the EEPROM are used to store the 64-bit ID of each sensor.

Each sensor ID consists of 8 bytes (16 hexadecimal letters), occupying 16 locations of the EEPROM memory. Thus, the first sensor ID starts at location 3 and terminates at location 18. Second sensor ID starts at location 19 and terminates at location 34, and so on.

When the device is in Run mode, a new file is opened on the SD card, RTC date and time are read, temperature data are read from sensors and then all of this data are written to the opened file. The same data are also sent to the serial port of the microcontroller.

The data logging activity is terminated when the Configure/Run button is kept pressed down for about five seconds. At this point the Run LED is turned off and the SD card can be removed from its adapter safely.

EEPROM ADDRESS	EEPROM CONTENTS
0	RTC year
1	Logging interval (mins)
2	Number of sensors used
3	Sensor 1 ID
.....	.....
.....	.....
18	Sensor 1 ID
19	Sensor 2 ID
.....	.....
.....	.....
34	Sensor 2 ID
.....	.....

Table 2: *Microcontroller EEPROM layout*

Main Program:
BEGIN
Initialise port directions
Initialise USART
Initialise I2C bus
Declare and initialise program variables

IF Configuration/Run pressed
Call Configuration Mode
ELSE
Call Run Mode
END IF
END

Configuration Mode:
BEGIN
Turn on Configuration LED (red)
DO FOREVER

Display MENU
1. Display Configuration
2. Edit Configuration
3. Edit
Choice:
IF Choice = "1"
Display Configuration
ELSE IF Choice = "2"
Edit Configuration
ELSE IF Choice = "3"
Wait to enter Run mode
END IF
END

Display Configuration:
BEGIN
Display date and time
Display logging interval
Display number of sensors
Display ID of each sensor
END

Edit Configuration:
BEGIN
Read Date and Time
Update RTC chip
Read logging interval
Read number of sensors
Read ID of each sensor
Store configuration in EEPROM
END

Run Mode:
BEGIN
IF SD card missing
Turn off both LEDs
Error - Halt
END IF
Turn on Run LED (green)
Initialise SPI bus
Create a new file on SD card

DO
Read temperature from sensors
Read date and time from RTC chip
Save RTC data and temperatures on SD card
Send RTC data and temperatures to serial port
UNTIL Configuration/Run button is kept pressed for 5 seconds
END

Figure 8: *Operation of the software*

Importing to Excel

The data stored on the SD card is compatible with the Excel spreadsheet and can easily be imported into Excel for either statistical calculations or for drawing graphs of the change of temperature with time. The analysis can either be carried out Offline, where the data stored on the SD card is used, or Online, where the data is analysed in real-time as it is captured from the data logger. Both methods are described here.

'Offline' Analysis

The steps for drawing a graph of the change of temperature for sensor 1 are given below:

- Start the Excel package
- Insert the SD card into an adaptor so that it can be read on a PC.
- Open the required temperature ".TXT" file on the SD card. Select File -> Open
- Select Delimited and click Next
- Select Delimeters as Space and Comma, and click Next
- Click Finish. You should see the collected data in an Excel sheet
- Highlight the time column (column B) and sensor 1 output column (column C)
- Click Chart Wizard icon in top menu
- Click Finish to select the default graph type (Bar chart)
- Click on the graph. Select Chart → Chart Options to enter the axes titles
- The graph of the change of temperature will be drawn as in **Figure 9** (this data was collected on the 29th of November at 14:28.

The graphs of other sensors, or multiple

sensors on the same graph, can easily be drawn with the Excel package. In addition, other types of graphs can be drawn, or the data can be analysed statistically, for example to find the maximum and minimum temperature in a given date and time interval, or the average, or the standard deviation of the temperature in a given interval, and so on.

'Online' Analysis

The data logger also sends data to its RS232 serial port at the requested intervals (see **Figure 10**) and it is possible to develop a program on the PC (e.g. using Visual Basic) to capture this data from the serial port dynamically and then draw the graph of the change of temperature with time as it happens. Such an application would be very useful in real-time temperature monitoring and control applications where it may be required for example to know the temperature of an oven at any instant of time, or to know the trend of change in the temperature.

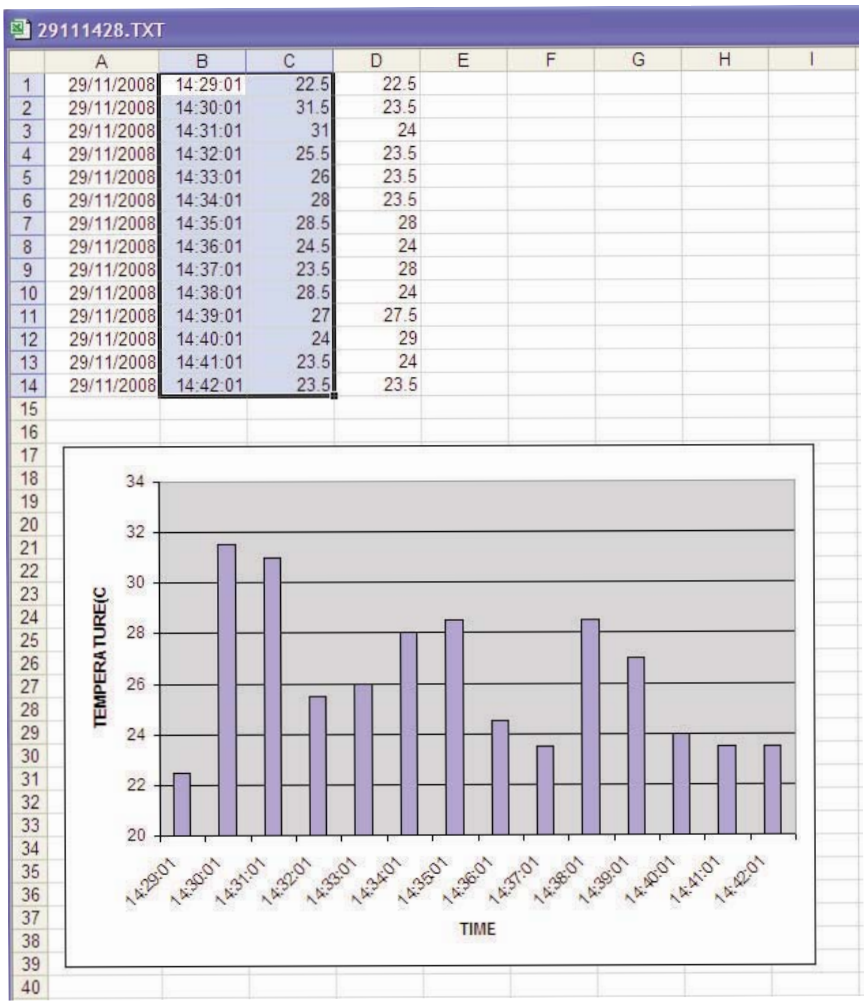
1-Wire Bus Device

The design of a microcontroller-based multichannel temperature data logger device, based on the 1-Wire bus has been described. The program code of the device is general and allows any number of temperature sensors to be connected to the 1-Wire bus, with a maximum bus length of several hundred meters.

Using one channel only, and logging every minute, the device writes about 36K of data to the SD card every day. Thus, with a 1GB SD card and providing the device is operated from a continuous power source, the data collection period is many years.

One of the problems with portable SD card based projects is that writing to the card draws high current from the battery and thus high capacity batteries (or a suitable mains adaptor if possible) should be used in applications where data is to be collected for long periods of time.

There are many types of 1-Wire devices in the market and the device described here can be expanded by the addition of other 1-Wire devices such as, humidity and pressure sensors, switches, counters, A/D converters EEPROM memories etc. Such an expansion will require modifications to the existing software code to read, format, and then store the relevant data on the SD card. ■



	A	B	C	D
1	29/11/2008	14:29:01	22.5	22.5
2	29/11/2008	14:30:01	31.5	23.5
3	29/11/2008	14:31:01	31	24
4	29/11/2008	14:32:01	25.5	23.5
5	29/11/2008	14:33:01	26	23.5
6	29/11/2008	14:34:01	28	23.5
7	29/11/2008	14:35:01	28.5	28
8	29/11/2008	14:36:01	24.5	24
9	29/11/2008	14:37:01	23.5	28
10	29/11/2008	14:38:01	28.5	24
11	29/11/2008	14:39:01	27	27.5
12	29/11/2008	14:40:01	24	29
13	29/11/2008	14:41:01	23.5	24
14	29/11/2008	14:42:01	23.5	23.5

Figure 9: *Offline analysis of the collected temperature data*

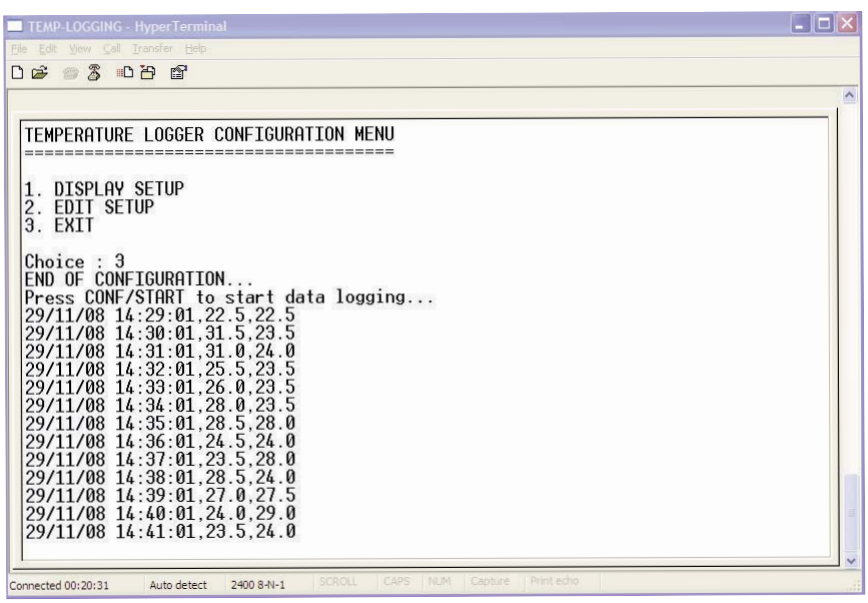


Figure 10: *Data sent to the microcontroller serial port*