MONITORING SYSTEM FOR SOLAR DRYING ROOM IN BASE OF PIC CONTROLLER AND INTELLIGENT THERMO SENSORS

Stanko Vl. Shtrakov, Anton Stoilov, Veselin Krekmanov

South - West University "Neofit Rilski",
66 Ivan Mihailov Str., 2700 - Blagoevgrad, BULGARIA

Abstract*: One of the most important parts in scientific researches is realization of reliable monitoring system. In this article is presented one possibility for using the microcontroller PIC16F84A as a controlling unit, and interface with the Dallas DS1820 1-Wire™ Digital Thermometers. Special software application for program service of this device is developed in visual programming environment DELPHI. The real experimental device, which uses this monitoring system, is a solar drying-room system. The interface for data receiving is COM port of the computer. After receiving, the data are archived in database format. The present monitoring device is applicable for different systems after some small changes..*

Keywords: *Monitoring, microcontrollers, intelligent thermo sensors, solar dryer-room*

1. INTRODUCTION

Automatic registration of experimental data in scientific researches is main purpose of using computers in modern investigation processes. For this goal there are developed effective intelligent devises, which do most of function, needed for converting the analogous signals in digital values of measured parameters.

This paper describes the complete design of a temperature logger.

2. THE BASIC DEVICE

The paper proposes one possibility for uses the CPU and intelligent thermo sensors for monitoring of the physical parameters of environment. The logger probes measure temperatures from –55°C to +125°C (–67°F to +257°F). A monitoring system acts as interface between the Dallas temperature sensors (DS 18S20) and a logging PC. There are 8 terminations or single wire busses. Each of the 8 busses can support up to 8 sensors. As many sensors can be installed as required, limited to 64 maximum. Each individual Dallas temperature sensor unique serial number. Where a particular probe is in relation to its peers, or particular bus pin on the PIC is irrelevant. The Arbiter sends to the PC the unique serial number of the temperature probe and its temperature in a serial frame.The build software program identifies sensors on each bus. Additional sensors can be plugged in at any time. Each device is polled every five seconds: its address and temperature data are sent via a RS232 serial link to the PC. The DS18S20 Digital Thermometer provides 9-bit centigrade temperature measurements.

The DS18S20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. It has an operating temperature range of –55°C to +125°C and is accurate to _0.5_C over the range of –10°C to +85°C. In addition, the DS18S20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply. Each DS18S20 has a unique 64-bit serial code, which allows multiple DS18S20s to function on the same 1-Wire bus; thus, it is simple to use one microprocessor to control many DS18S20s distributed over a large area. Applications that can benefit from this feature include HVAC environmental controls, temperature monitoring systems inside buildings, equipment or machinery, and process monitoring and control systems.

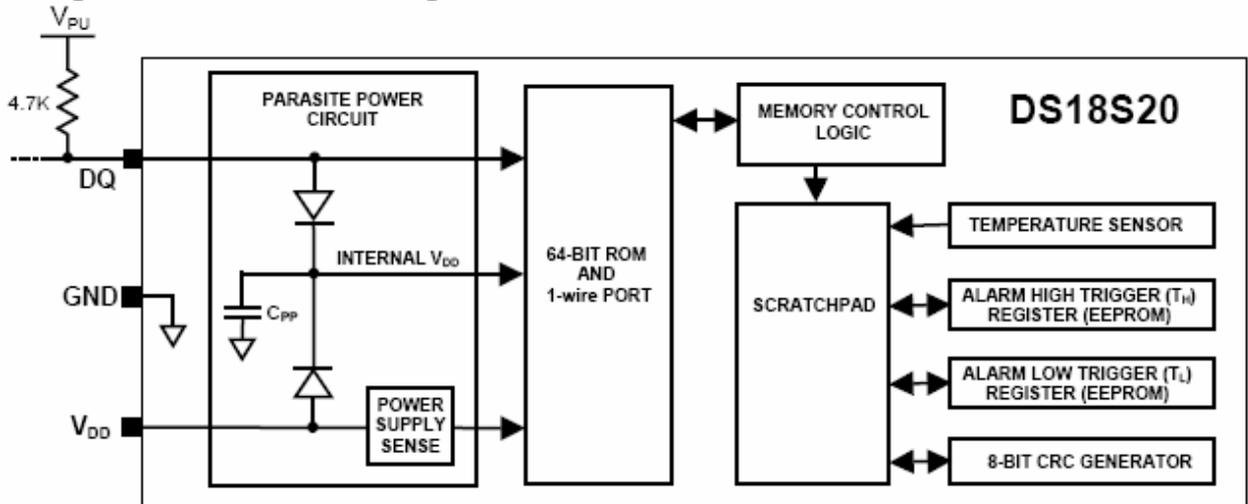Figure 1 shows a block diagram of the DS18S20.

Fig.1. DS18S20 block diagram

The core functionality of the DS18S20 is its direct-to-digital temperature sensor. The temperature sensor output has 9-bit resolution, which corresponds to 0.5^0C steps. The DS18S20 powers-up in a low-power idle state; to initiate a temperature measure-ment and A-to-D conversion, the master must issue a Convert T [44h] command. Fol-lowing the conversion, the resulting thermal data is stored in the 2-byte temperature register in the scratchpad memory and the DS18S20 returns to its idle state. If the DS18S20 is powered by an external supply, the master can issue "read-time slots" after the Convert T command and the DS18S20 will respond by transmitting 0 while the tem-perature conversion is in progress and 1 when the conversion is done. If the DS18S20 is powered with parasite power, this notification technique cannot be used since the bus must be pulled high by a strong pullup during the entire temperature conversion. The DS18S20 output data is calibrated in degrees centigrade. The temperature data is stored as a 16-bit sign-extended two's complement number in the temperature register (Figure 2). The sign bits (S) indicate if the temperature is positive or negative: for posi-tive numbers S = 0 and for negative numbers S = 1. Table 1 gives examples of digital output data and the corresponding temperature reading.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
LS Byte	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}
	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
MS Byte	S	S	S	S	S	S	S	S

Fig.2. Temperature register format

Resolutions greater than 9 bits can be calculated using the data from the temperature, COUNT REMAIN and COUNT PER °C registers in the scratchpad. Note that the COUNT PER °C register is hard-wired to 16 (10h). After reading the scratchpad, the TEMP_READ value is obtained by truncating the 0.5^0C bit (bit 0) from the temperature data (Figure 2). The extended resolution temperature can then be calculated using the following equation:

$$TEMPERATURE = TEMP\_READ - 0.25 + \frac{COUNT\_PER\_C - COUNT\_REMAIN}{COUNT\_PER\_C}$$

TEMPERATURE	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+85.0°C*	0000 0000 1010 1010	00AAh
+25.0°C	0000 0000 0011 0010	0032h
+0.5°C	0000 0000 0000 0001	0001h
0°C	0000 0000 0000 0000	0000h
-0.5°C	1111 1111 1111 1111	FFFFh
-25.0°C	1111 1111 1100 1110	FFCEh
-55.0°C	1111 1111 1001 0010	FF92h

Table 1. Temperature/data relationship

Each DS18S20 contains a unique 64-bit code (see Figure 6) stored in ROM. The least significant 8 bits of the ROM code contain the DS18S20's 1-Wire family code: 10h. The next 48 bits contain a unique serial number. The most significant 8 bits contain a cyclic redundancy check (CRC) byte that is calculated from the first 56 bits of the ROM code.

8-BIT CRC	48-BIT SERIAL NUMBER	8-BIT FAMILY CODE (10h)
MSB LSB	MSB LSB	MSB LSB

Fig. 3. 64-bit lasered ROM code

The DS18S20's memory is organized as shown in Figure 4. The memory consists of an SRAM scratchpad with nonvolatile EEPROM storage for the high and low alarm trigger registers (TH and TL). Byte 0 and byte 1 of the scratchpad contain the LSB and the MSB of the temperature register, respectively. These bytes are read-only. Bytes 2 and 3 provide access to TH and TL registers. Bytes 4 and 5 are reserved for internal use by the device and cannot be overwritten; these bytes will return all 1s when read. Bytes 6 and 7 contain the COUNT REMAIN and COUNT PER °C registers, which can be used to calculate extended resolution. Byte 8 of the scratchpad is read-only and contains the cyclic redundancy check (CRC) code for bytes 0 through 7 of the scratchpad. Data is written to bytes 2 and 3 of the scratchpad using the Write Scratchpad [4Eh] command; the data must be transmitted to the DS18S20 starting with the least significant bit of byte 2. To verify data integrity, the scratchpad can be read (using the Read Scratchpad [BEh] command) after the data is written. When reading the scratchpad, data is transferred over the 1-Wire bus starting with the least significant bit of byte 0. To transfer the TH and TL data from the scratchpad to EEPROM, the master must issue

the Copy Scratchpad [48h] command. Data in the EEPROM registers is retained when the device is powered down; at power-up the EEPROM data is reloaded into the corresponding scratchpad locations. Data can also be reloaded from EEPROM to the scratchpad at any time using the Recall E2 [B8h] command.
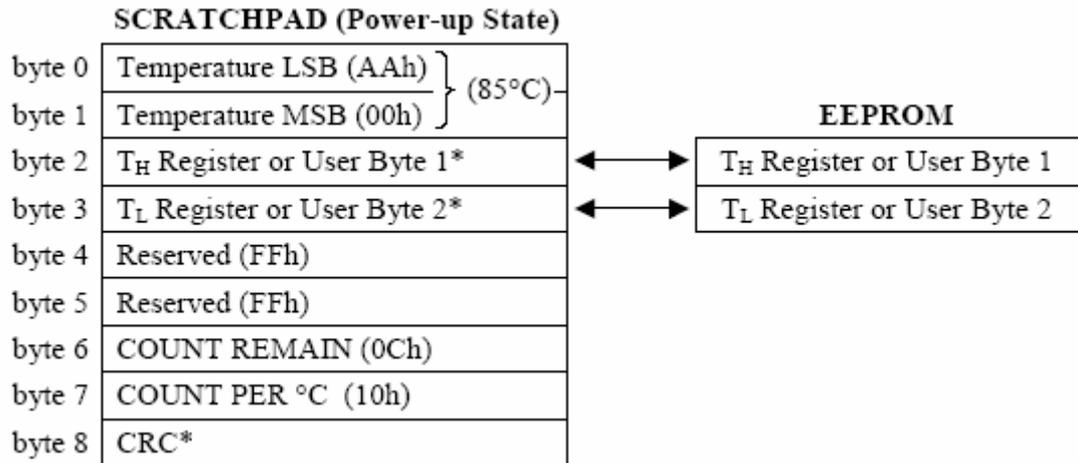

Fig.4. DS18S20 Memory map

The reading from the DS18S20 was realized with collaboration with a microcontroller PIC016F84A. The PIC 16F84A is a High Performance RISC CPU. The PIC16F84A belongs to the mid-range family of the PICmicro™ microcontroller devices. A block diagram of the device is shown in Figure 5.
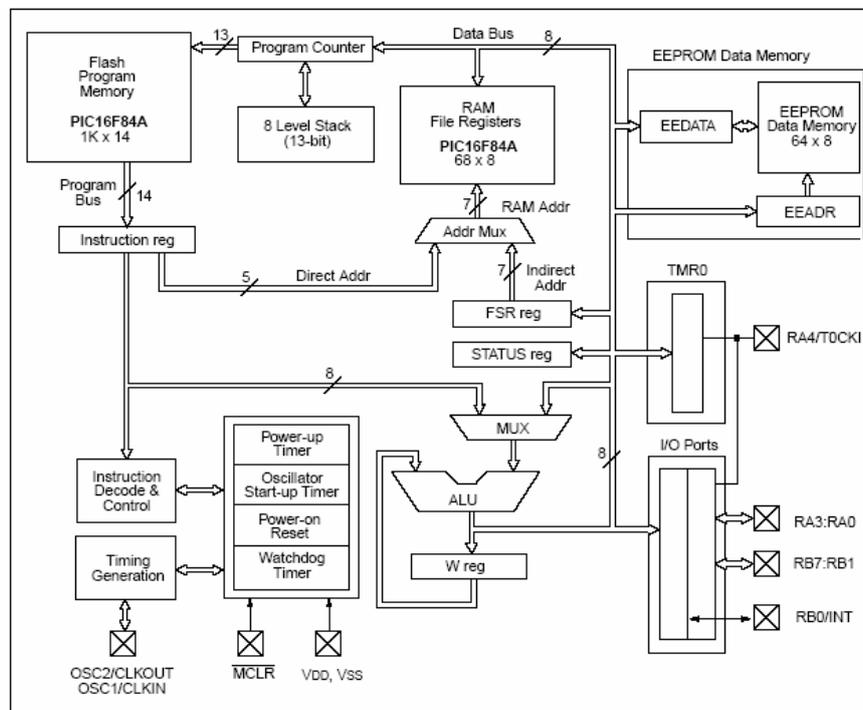

Fig.1. PIC 16F84A Block diagram

The program memory contains 1K words, which translates to 1024 instructions, since each 14-bit program memory word is the same width as each device instruction.The data memory (RAM) contains 68 bytes. Data EEPROM is 64 bytes. There are also 13 I/O pins that are user-configured on a pin-to-pin basis.

There are two memory blocks in the PIC16F84A. These are the program memory and the data memory. Each block has its own bus, so that access to each block can occur during the same oscillator cycle. The data memory can further be broken down into the general purpose RAM and the Special Function Registers (SFRs). The operation of the SFRs that control the "core" are described here. The SFRs used to control the peripheral modules are described in the section discussing each individual peripheral module. The data memory area also contains the data EEPROM memory. This memory is not directly mapped into the data memory, but is indirectly mapped. That is, an indirect address pointer specifies the address of the data EEPROM memory to read/write. The 64 bytes of data EEPROM memory have the address range 0h-3Fh.

2.1.1. 3. The Program service

The temperature values are registered and sent to the RS-232 port once per second. Special terminal capture program is used for data transferring. The output signal from basic plate is a digital signal proportional to the temperature and humidity. The protocol signal for transmitting from microcontroller to the serial port is RTS=0. RS232 send the number of channel (from 0 to 14) to the microprocessor AT90S8535. After that, AT90S8535 send to RS232 current value of this channel, corresponding with one of the sensors. When the data are received, RS232 sends the different value from 0 (such as 255). Protocol signals RTS=1 and RTS=0 are sent consequently for each of the sensors. The source code serves the communication protocol in application for PC is:

```
for i:=Index to 15 do
begin
ComPort1.SetRTS; //RTS-0
ComPort1.WriteByte(Channel);
a:=ComPort1.ReadWord;
a:=((4*a/1024)*100)-273.15;
ComPort1.WriteByte(255);
ComPort1.ClearRTS; //RTS=1
ComPort1.SetRTS; //RTS=0
end;
```

For realization of the transfer protocol it is used a DELPHI component ComPort. It is external component for visual programming environment DELPHI that means it was installed in addition. This component supports all needed properties for communication with AT90S8535. In source code the name of this component is ComPort1.

The digital signal received on RS232 is directly formed from ADC of microcontroller. After receiving this signal, it must be decoded for receiving voltage value in sensors, which is proportional to the real physical parameter (temperature or humidity). For this purpose there is software processing. First, receiving value is multiplied with value of V_{cc} (basic voltage in ADC; it is 4V) and after that it must be divided by 2^{10} or 1024 corresponding with the 10 bit ADC:

$$RD = \frac{4 * ADC}{1024}$$

where RD is a real value of received physical parameter.

The temperature conversion is simple: received voltage is corresponding with the value of thermo sensor in Kelvin, so 2,93 V corresponds to 293K. For humidity-received voltage there is a linear dependence between voltage and humidity. Voltage in channel with humidity sensor is in range of 0 to 3V, so 3V corresponds to 100% humidity.

4. Program structure

When the program is started on the monitor is displayed the screen shown in figure 4. This is the main window of application. The information received on the serial port of computer is shown on the top right part of the screen in table form. This table includes 4 columns (Number of sensor, Value of measurement quantity, Time, Color). The number of rows is corresponding with the number of sensors (15).

The diagram on the screen shows received data in graphic mode as for every one of the sensors was drawn up own line with different color, corresponding with received data. It has possibility to change the color of everyone of the line or hide some of the sensor. In panel "Control" situated in the bottom part of the screen, the user can start reading ("Start"), stop reading ("Stop"), make print preview of diagram ("Print Preview") or close program ("Close"). In the top of the main form of the application is drag-drop menu. It has four segments "Settings", "Diagram", "Help", "Close".

First item "Settings" includes 2 submenus "Time settings" and "Diagram settings".

-"Time settings" – this item activate a window, where a time parameters can be regulated for refresh of data (at what interval read information on RS232), for save data in database (at what interval system makes record in database) and clear diagram (at what interval the canvas of the diagram be cleared).

When button "OK" is pressed, the time settings are saved in file (configuration file .ini), so that when user goes out of the application and then starts again the settings are ready for use. The application makes this file automatically and carries out for is existence. In case of deleting or damaging them, application recovered them with default values.
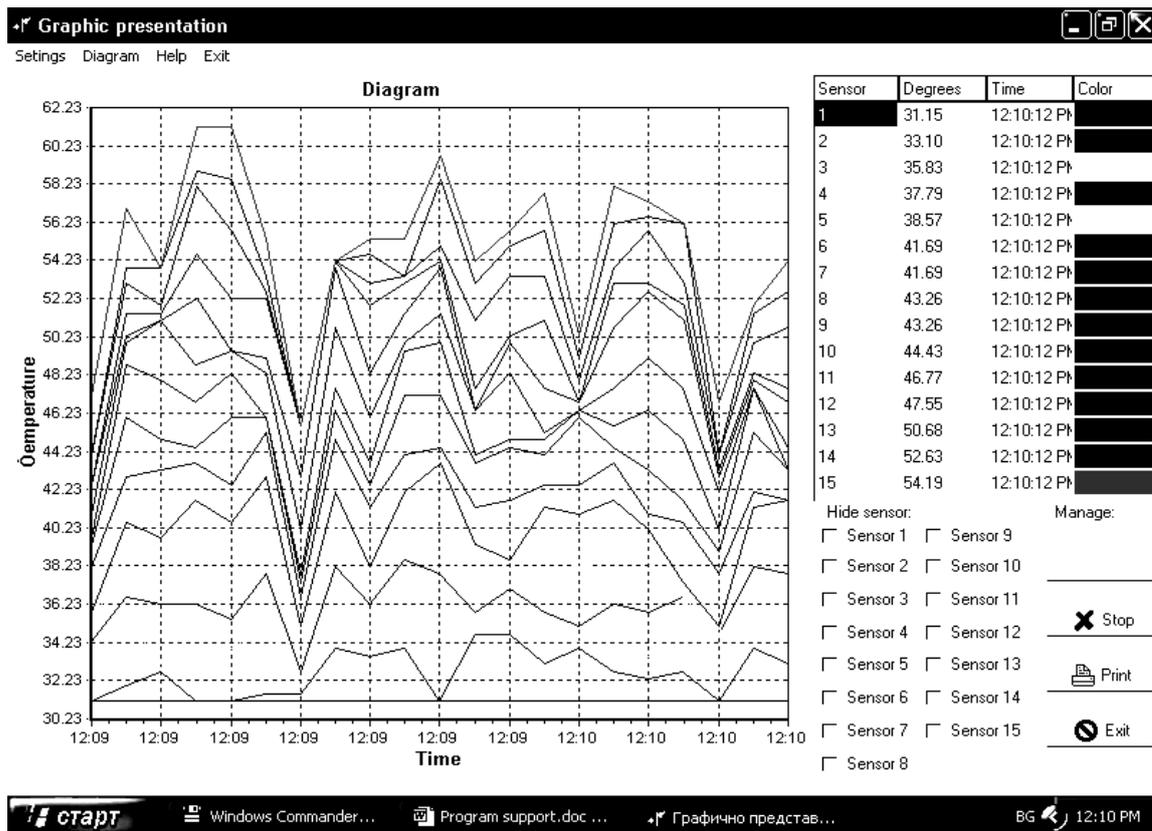
Fig.4. Start screen of application

- "Diagram settings" – this item starts a window with tree pages. There are possibilities for regulating the graphic mode of the diagrams. In first page can be defined colors of lines in diagram for different sensors. Second page regulates scale for data presentation and third page defines the diagram mode – 2D or 3D graphic.

Second item in main menu "Diagram "shows another form. In it is presented current diagram of every one of the sensors (fig.5). Double click at one of the diagram shows print preview of them.

All received experimental data are saved in database form Paradox 7. The package includes another application for reports ensure the possibilities taking from database experimental results, for different periods and for different sensors. The data can be exported in text file or in Excel file.
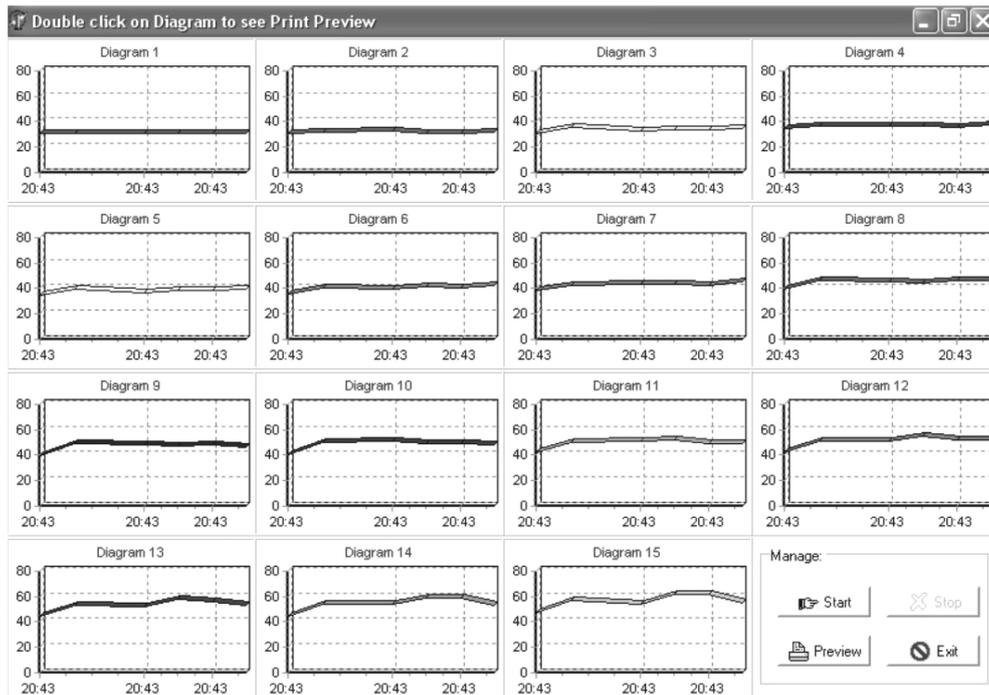
Fig.5. All sensors

4. Conclusions

Monitoring system with AVR microcontroller for registration the physical parameters (temperature and humidity) in special experimental unit was built. The system has been tested and a good reliability and functionality has been registered. This work shows that with on-chip in-system programmable Flash and EEPROM, the AVR is a reasonable choice to optimize for cost and get products to the market quickly.

Software package created with visual programming environment DELPHI allows a PC communicates with a nonstandard device. For realization of the transfer protocol it is used a DELPHI component ComPort. This component supports all needed properties for communication with AT90S8535.

References

[1] 8-bit AVR Microcontroller with 8K Bytes In-System Programmable Flash AT90S8535 AT90LS8535, Rev. 1041H–11/01 (http://www.atmel.com/dyn/resources/prod_documents/DOC1041.PDF)

[2] Gadre Dhananjay, (2001) Programming and Customizing the AVR Microcontroller, The Mchill-Graw co.,

[3] Cantu Marco, (2002) Mastering Delphi 6, Softpress,