INTERFACING SENSOR WITH ADC OF AVR MICROCONTROLLER

■ ARUN DAYAL UDAI

Sensors mimic our biological senses and respond to physical stimuli such as light, heat or movement. These can sometimes even outperform our natural sense organs in certain conditions. One can have additional senses with direction sensors, gyroscopes and infrared sensors. The evolution of microelectromechanical systems (MEMS) sensor technology has surpassed our imagination through its applications in handheld electronic gadgets including digital cameras, cellphones, gaming stations to that in space robots, aircrafts, simulators, etc.

This article explains interfacing a sensor with AVR microcontroller and displaying the sensor's output on a PC monitor.

Sensor signal conversion

Normally a sensor consists of a sensing element and a transducer that converts the energy into a desired form (normally electrical). For example, in a microphone the small displacement of the diaphragm is converted into electrical signals. Most of the sensing elements can be made to generate an analogue direct current (DC) voltage by suitable circuitry. Also, a simple photo-resistive sensor can be used to generate an equivalent voltage signal that can sense light intensity. Similarly, a negative temperature coefficient (NTC) thermistor or a centigrade temperature sensor can be used to convert heat to voltage signals.

Most of the sensors change their resistance or capacitance with change in physical parameters. The change in resistance or capacitance can be converted to voltage signals. Microcontroller simplifies the sensor circuitry by allowing voltage input (0-5V range) through its analogue-to-digital converter (ADC) pins, which can be programmed to trigger a function at a particular value, or can produce variable functions at different values of input voltage.

In context to AVR microcontrollers, any sensed parameter can be fed to one of its single-ended ADC input channel, which accepts a positive DC voltage and converts the signal to equivalent 10-bit digital value in its ADC registers—ADCH and ADCL. The value in ADC registers is given by:

$$ADC = \frac{V_{IN} \times 1024}{V_{REF}}$$

V_{REF} is the selected reference voltage, which ensures the maximum value close to 1024 (3FF hexadecimal, 10-bit maximum value). Default value can be set at 2.56V internal supply.

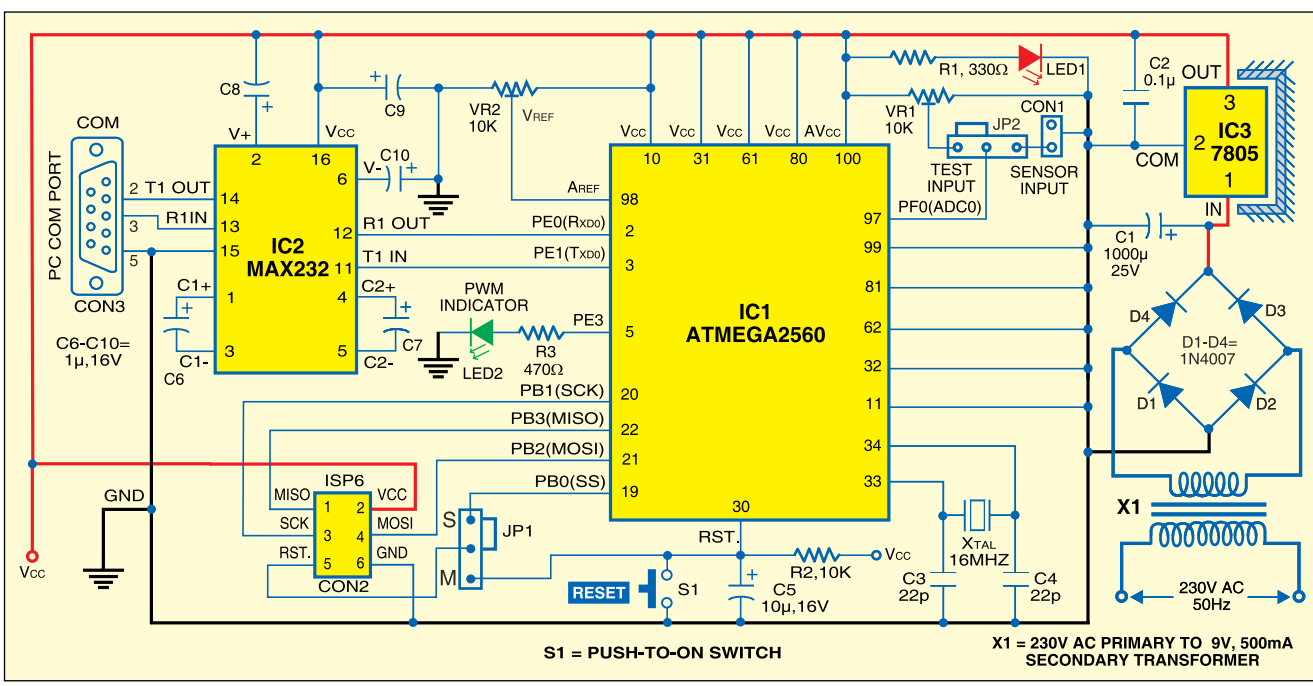

Fig. 1: Interfacing sensor with ADC of AVR microcontroller

Negative voltage can also be measured if differential input mode—which is not very commonly used for sensor input—is used. In such case the value in ADC registers is given by:

$$ADC = \frac{(V_{POS} - V_{NEG}) \times GAIN \times 512}{V_{REF}}$$

Circuit description

Fig. 1 shows the interfacing sensor with ADC of AVR microcontroller. At the heart of the circuit is ATmega2560 (IC1). The ATmega2560 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. It features 128kB Flash memory with read-while-write capabilities, 4kB EEPROM, 8kB SRAM, 86 general-purpose input/output (I/O) lines, 32 general-purpose working registers, six flexible timers/counters with compare modes, real time counter (RTC), 10-bit ADC, internal and external interrupts, a serial programmable USART, a programmable watchdog timer with internal oscillator and an SPI serial port.

System clock plays a significant role in the operation of the microcontroller. A 16MHz quartz crystal connected to pins 33 and 34 provides clock pulse to the microcontroller (IC1). Power-on reset is provided by electrolytic capacitor C5 and resistor R2. Switch S1 is used for manual reset. Port pins PB2 (master out, serial data in or MOSI), PB3 (master in, serial out or MISO) and PB1 (serial clock pin or SCK) are used for in-system programming (ISP). IC MAX232 (IC2) is used for serial communication. T_{XD0} and R_{XD0} pins of the microcontroller too are used for serial communication with the help of COM port. Pins R_{XD0} and T_{XD0} of the microcontroller are used to send converted analogue data (through ADC) to the PC through COM port .

ADC channel PF0(ADC0) of microcontroller is used for sensor input. The ADC converts an analogue input voltage to a 10-bit digital value through successive approximation. The converted data can be read from ADC register. It sends ADC conversion results continuously through serial port to the PC at 115.2kbps baud rate. The value received from the ADC pin is reflected in the form of pulse-width modulated (PWM) output at port pin PE3 of the microcontroller, where LED2 is connected to observe the brightness. The ADC contains a sample-and-hold circuit, which ensures that the input voltage to the ADC is held at a constant level during conversion. The ADC has a separate analogue supply voltage pin (AVcc). The reference voltage (V_{REF}) for the ADC is adjusted through preset VR2, which indicates the conversion range for the ADC.

The 230V, 50Hz AC mains is stepped down by transformer X1 to deliver a secondary output of 9V, 500mA. The transformer output is rectified by a full-wave rectifier comprising diodes D1 through D4, filtered by capacitor C1 and regulated by IC 7805 (IC3). Capacitor C2 bypasses the ripples present in regulated supply. LED1 acts as the power indicator and resistor R1 limits the current through LED1.

An actual-size, single-side PCB for the circuit of interfacing sensor with ADC of AVR microcontroller is shown in Fig. 2 and its component layout in Fig. 3. Assemble the circuit on a PCB to minimise time and assembly errors. Carefully assemble the components and double-check for any overlooked error. Assemble the board and check with multimeter. Burn the hex code to microcontroller using suitable programmer through ISP6 connector. Connect the board to the PC's COM port and run serial port interface application.

Serial port interface application window is shown in Fig. 4 for selection of serial COM port. After selection of COM port, set the voltage reference using preset VR2 and apply the test input through preset VR1. Test-input voltage appears along with the voltage

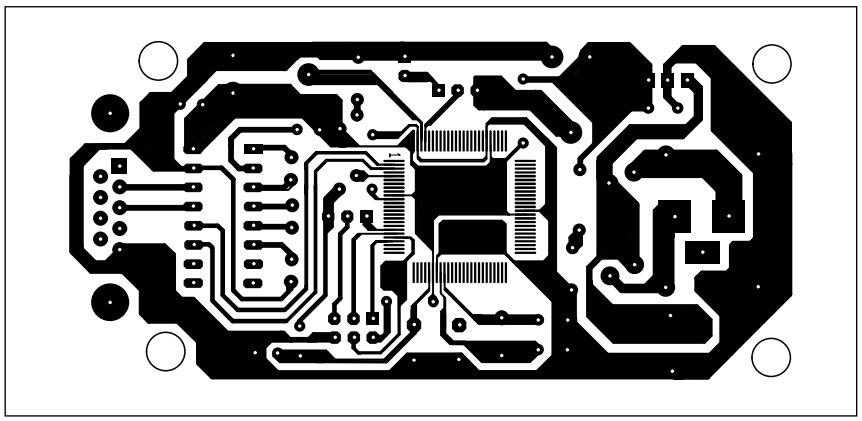

Fig. 2: An actual-size, single-side PCB for the interfacing sensor with ADC of AVR microcontroller

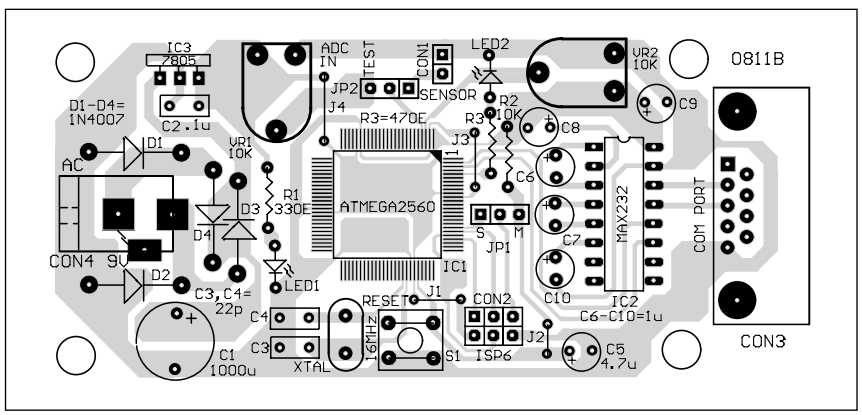

Fig. 3: Component layout for the PCB

level slider control on computer screen. Now it is ready for connecting the sensor such as temperature sensor or LDR to sensor input. The sensor output voltage level will be shown in the PC screen as shown in Fig. 4.

Software

The software for interfacing sensor with ADC of AVR microcontroller are written in 'C' using IAR Embeded Workbench integrated development environment. IAR Embedded Workbench is being developed by IAR Systems and ATMEL developers in parallel and hence it generates the optimised code which uses full 'C' coding capabilities of AVR devices. AVR development tools for embedded systems can be downloaded for free from IAR website www.iar.com. For details of IAR Embedded Workbench, you may refer to 'A Beginners' Guide to ATMEL AVR Development' article published in January issue of EFY.

ADC channel in AVR is initialised by setting ADC enable (ADEN) bit in ADC control and status register A (ADCSRA), which can also be used to set the frequency for ADC conversion. ADC multiplexer selection register (ADMUX) is used to select the channel to be used for ADC input. Setting the ADC start conversion (ADSC) bit starts conversion and sets the ADC interrupt flag (ADIF) in ADCSRA register on completion. The converted data can be read from ADC register.

PARTS LIST

Semiconductors:

IC1	- ATmega2560 microcontroller
IC2	- MAX232 RS-232 driver
IC3	- 7805, 5V regulator
D1-D4	- 1N4007 rectifier diode
LED1, LED2	- 5mm LED

Resistors (all ¼-watt, ±5% carbon):

R1	- 330-ohm
R2	- 10-kilo-ohm
R3	- 470-ohm
VR1, VR2	- 10-kilo-ohm preset

Capacitors:

C1	- 1000µF, 25V electrolytic
C2	- 0.1µF ceramic disk
C3, C4	- 22pF cermaic disk
C5	- 10µF, 16V electrolytic
C6-C10	- 1µF, 16V electrolytic

Miscellaneous:

X1	- 230V AC primary to 9V, 500mA secondary transformer
S1	- Push-to-on tactile switch
CON1	- 2-pin berg strip male connector
CON2	- 6-pin berg strip male connector
CON3	- 9-pin D-type connector
JP1-JP2	- 3-pin berg strip male connector with shorting jumper

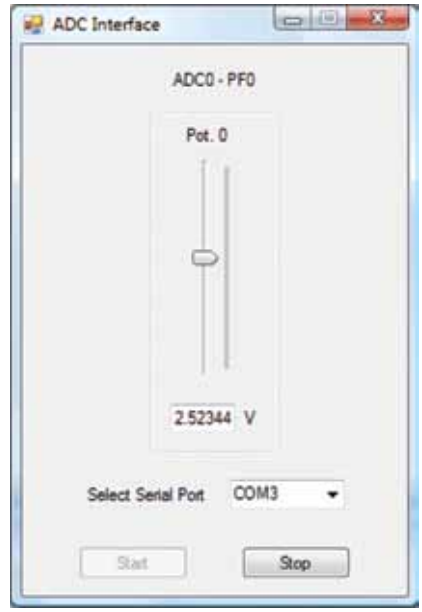

Fig. 4: ADC interface windows

Source code

The source code is demonstrated using one of the ADC inputs of ATmega2560 microcontroller ADC channel. Following the steps demonstrated in the article, you can create your C# application for interfacing the circuit board. You should, however, have a version of Microsoft Visual Studio .NET. You can use C# for effective serial communication in your own development board or the current ADC application for receiving serial data.

To create the application run Microsoft Visual Studio and create a new C# based Windows form application. One is expected to get a blank windows form1. Using the toolbox, drag and drop the controls to create a front-end tool like the one shown in Fig. 4. Set the properties of form1 as MaximizeBox = False, StartPosition as CenterScreen, FormBorderStyle as FixedDialog and name it as 'ADC Interface.' Similarly, for the 'trackBar-Pot1' set the maximum value as 255, orientation as 'Vertical' and 'TickFrequency' as 5. Add texts to the buttons as 'Start' and 'Stop.' This completes the front-end development. Now event-driven codes are to be inserted for the controls. To do this, double click on the controls to enter the coding environment for each event. The code should be as listed below:

```
using System;
using System.Windows.Forms;

namespace Simple_Serial
{
    public partial class Form1 : Form
    {
        int RxData;

        public Form1()
        {
            InitializeComponent();
        }

    }
}
```

RxData is the variable to store the data received through serial port, which is declared in the form as given below:

```
private void Form1_Load(object
sender, EventArgs e)
{
PortCombo.Items.AddRange(System.
IO.Ports.SerialPort.GetPortNames());
       if(PortCombo.Items.Count == 0)
            MessageBox.Show("No
Serial port is available!");
}
```

The above mentioned subroutine executes as soon as the form is loaded. It obtains all the available port names from the system and adds it to the drop-down combo list from where one can select the port where the device is connected. On selecting the port name from the combo box the following code executes. It initialises the 115.2kbps baud rate. The start button is enabled and stop button is disabled.

```
private void PortCombo_
SelectedIndexChanged(object sender,
EventArgs e)
{
serialPort1.PortName = PortCombo.
Text;
serialPort1.BaudRate = 115200
buttonStart.Enabled = true;
buttonStop.Enabled = false;
}
```

On pressing the start button, the serial port is enabled. The start button gets disabled and the stop button is enabled using following code:

```
private void buttonStart_
Click_1(object sender, EventArgs e)
{
if (serialPort1.IsOpen == false)
{
serialPort1.Open();
buttonStart.Enabled = false;
buttonStop.Enabled = true;
}
}
```

On pressing the stop button, the serial port is closed, start button is enabled and stop button is disabled.

```
private void buttonStop_
Click_1(object sender, EventArgs e)
{
if (serialPort1.IsOpen)
{
serialPort1.Close();
buttonStart.Enabled = true;
buttonStop.Enabled = false;
}
}
```

Now, to create an event of the track bar position from the incoming serial port data, select the serial port control in the form1 panel. Click on 'events' in the 'properties' tab, double click on 'serialPort1_DataReceived' event and add the following code:

```
private void serialPort1_
DataReceived(object sender, System.IO.
Ports.SerialDataReceivedEventArgs e)
{
RxData = serialPort1.ReadByte();
this.Invoke(new
EventHandler(ChangeTrackbar));
}
```

As soon as the data is received, the source code above reads the existing byte from the serial port and takes the code to new event handler 'ChangeTrackbar' as defined below:

```
private void ChangeTrackbar(object
sender, EventArgs e)
{
trackBarPot1.Value = RxData;
//Set potentiometer input value
(4.84)
textBoxPot1.Text = Convert.
ToString(RxData*4.84/255);
}
```

The handler sets the 'trackBarPot1' to the position as received from the serial port and stores in the variable RxData. The users may change the value of 4.84 to their own reference voltage. The voltage value is given as follows:

ADC Value × V_{REF}/255, for 8-bit data.

You can change the source code demonstrated in the article to your application, which may be an analogue centigrade sensor (e.g., LM 35) reading the temperature of a bearing or a room.

EFY note. The source codes of this article are available on www.efymag.com. ●

The author is assistant professor in Department of Mechanical Engineering at Birla Institute of Technology, Mesra