Basic Analog to Digital Thermometer using Atmega328p

Douglas Smith, Noah Harvey

November 9, 2014

Abstract

A thermometer is a device used to measure and display the temperature of an external environment. A typical thermometer can be created using a Atmega328p microcontroller, thermistor and seven segment display. The main challenge in this design is measuring and storing the temperature in the microcontroller. A ten-bit analog-to-digital converter (ADC) in the Atmega328p can be used to overcome this challenge. Temperature is converted to an analog voltage signal by the thermistor which is then converted to a binary number by the ADC and is stored in the Atmega328p. Linear interpolation is then performed on this number to obtain a temperature value which is then displayed via a two digit seven segment display. Testing of this method showed that the thermometer worked for temperatures ranging from zero to forty degrees Celsius.

Chapter 1

Program Logic

1.1 Overview

The Atmega328p application program uses two processes to asynchronously control the external display and temperature conversion tasks (as shown in Figure 1.1). Temperature calculations are handled by the ADC interrupt service routine (or ADC ISR). The ADC ISR obtains the result from the ADC and calls setTemp() to handle temperature calculations. An eight-bit timer on the Atmega328p is used to manage the seven segment display. Multiplexing is used to display a value on the and the setNum() function is the application interface to set the number to display.

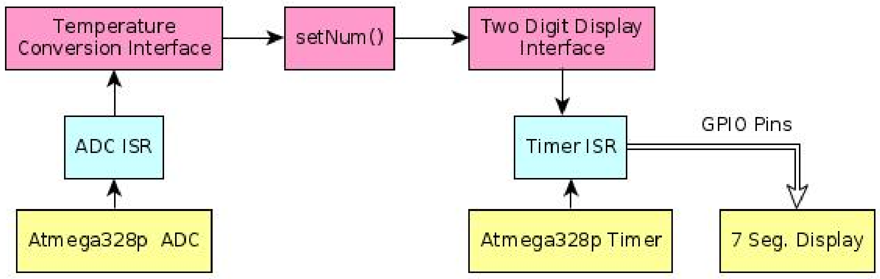

Figure 1.1: Thermometer Program Logic

1.2 Configuration

1.2.1 Timer Configuration

The eight-bit timer is configured to run at approximately fifty-two kilohertz in Clear Timer on Compare mode. CTC mode is an operation mode of the timer in which the internal counter is reset when the counter matches the value stored in the timer's output compare register. Once the timer is reset a timer interrupt vector flag is set and the timer ISR is called to handle the display.

1.2.2 ADC Configuration

The ADC is configured to run in Free running mode. This is an operation mode in which the ADC is constantly performing conversions. Once a conversion is performed an ADC flag is set and an interrupt is called and the ADC immediately begins performing another conversion. The ADC uses the ADC5 port as the voltage signal input.

1.2.3 IO Register Configuration

Table 1.1 shows a table of specific ports and their respective pin configurations.

Port	Pins	Configuration
PORTD	0-7	Output pins to control segment LEDs on seven-segment display.
PORTC	0-1	Output pins to control digit selection on seven segment display.
PORTC	5	Input pin for ADC

Table 1.1: Atmega328p Pin Configurations

1.3 Two Digit Display

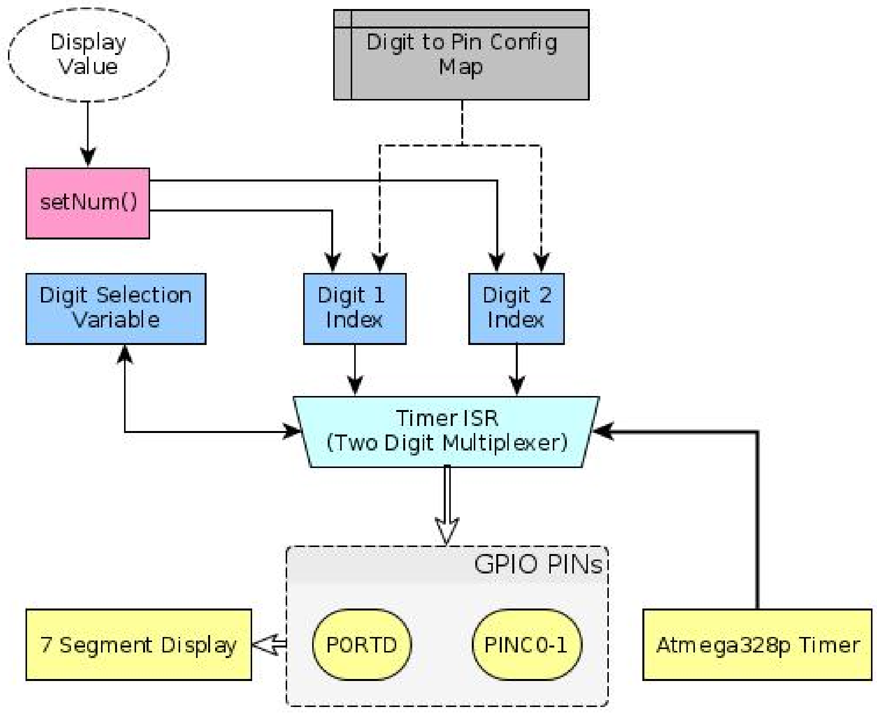

Figure 1.2: Two Digit Display Interface

1.3.1 Single Digit Control

The two digit seven segment display is controlled by the Atmega328p via two I/O ports. PORTD controls the segments on the display and PORTC controls the digit selection. A mapping array is used to map digits 0 through 9 to their respective output register values. For example, the value at index 3 in the mapping array stores the respective value for the PORTD register. This register value is then used to show the number 3 on the seven segment display. Two variables store indexes to this mapping array, each for a single digit on the display. These variables are set by the setNum() function and are used by the timer ISR to set the PORTD register. A variable is used to determine which digit to display when the ISR is called. Figure 1.2 shows an overview of the display system.

1.3.2 Two Digit Multiplexing

To display a two digit number a multiplexing method is used. This is done by alternatively toggling each digit on the display so that only one digit is on for a given time period. When the digits are alternatively toggled at a frequency larger than 10kHz they appear as if they are displaying simultaneously.

The timer ISR handles the multiplexing of the two digits. Below is the process used in the timer ISR to perform two digit multiplexing:

1. Turn currently displayed digit off.
2. Select other digit to display using the digit selection variable.
3. Turn selected digit on using the corresponding index variable.

1.4 ADC to Temperature Conversion

The Atmega328p has a ten bit successive approximation analog to digital converter (noted as ADC). Successive approximation is an analog to digital conversion method in which the input signal is constantly compared to the output of a guessed analog value. (see Figure 1.3 for a simplified flow diagram of a successive approximation ADC.). After all n bits of the converter are set the ADC raises a conversion the ADC interrupt service routine (or ADC ISR) is called to handle temperature conversion (see the ADC to Temperature Conversion section below).

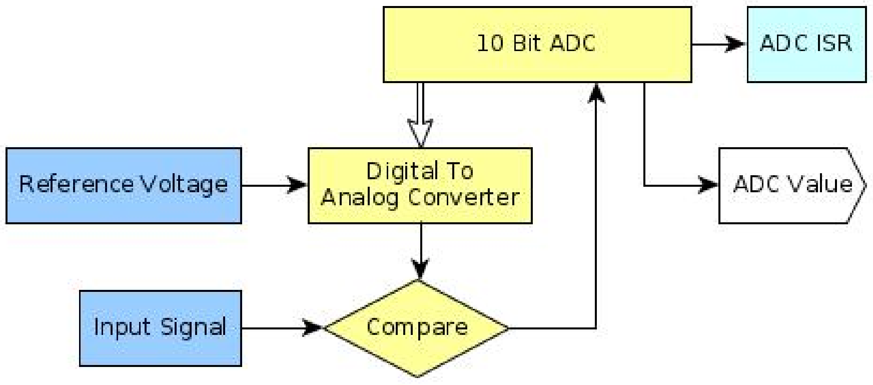

Figure 1.3: ADC Successive Approximation

1.4.1 Obtaining the ADC Conversion Result

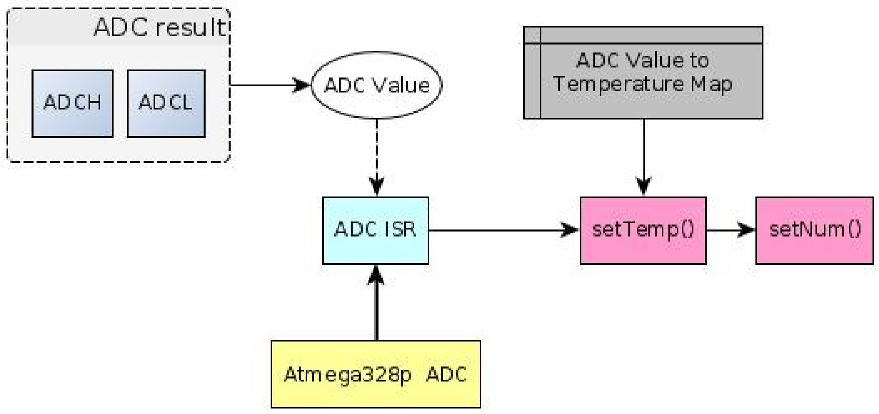

Figure 1.4: Temperature Conversion Interface

The ADC uses two eight bit registers (called ADCL and ADCH) to store the result from an analog to digital conversion. The conversion result is obtained by fetching the ADCL register data first (which is required according to the Atmega328p datasheet) and then the ADCH register data. The final result is stored in a sixteen bit integer (noted as the ADC value) which is then passed to a temperature conversion function setTemp(). Figure 1.4 shows the process in which the temperature conversion interface was designed.

1.4.2 Calculating the Temperature

A function called setTemp() handles the conversion of an ADC value to a temperature. Below is the process for which setTemp() uses to convert and ADC value.

1. Fetch nearest data points from Temperature conversion table.
2. Perform linear interpolation using integer math to compute a temperature.
3. call setNum() to display the result to the seven segment display.

Linear interpolation was the method used to calculate temperatures based on given ADC values. See the below section for details on linear interpolation.

1.4.3 Linear Interpolation

Linear interpolation is a method of estimating function values from a table of known input and function values. The idea is that two points are chosen from the table (one greater than, one less than the desired value) and a linear equation is formed from those two points. Then the desired function value is found by ‘plugging in’ a given input value into this function.

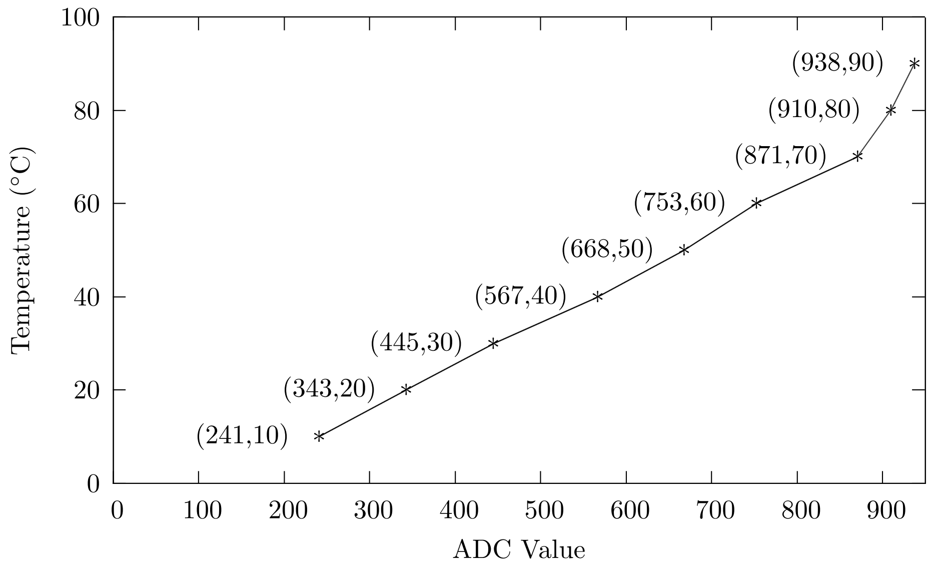

Figure 1.5: Sample ADC value to Temperature Linear Interpolation

Graph 1.5 shows an example of a linear interpolation. Each data point shown is a known ADC value and its corresponding temperature. The lines between each data point represent the functions to be used for linear interpolation.

For example if the ADC value 400 was returned by the ADC and one wished to know the corresponding temperature then one would find the two nearest points ((343, 20) and (445, 30)) and create the line between the two and substitute the ADC value:

$$\begin{aligned} f(x) &= \frac{(30-20)}{(445-343)}(x-343)+20 \\ &\therefore \quad f(400) \approx 25.59°\mathrm{C} \end{aligned}$$

A generic equation would be:

$$f(x) = \frac{(y_2 - y_1)}{(x_2 - x_1)}(x - x_1) + y_1$$

where $(x_{1,2}, y_{1,2})$ are two nearest points in the table surrounding a given ADC value x_n.