Example of a Design Report

Source: Virginia Tech. Writing Guidelines for Engineers and Scientists. Last updated March 2004. Accessed 1 February 2007 { HYPERLINK "http://www.writing.eng.vt.edu/workbooks/designreport.html#introduction" }

Design of a Temperature Measurement and Display System Using the 68HC11 Microcontroller

Executive Summary

This report presents the design of a temperature measurement and display system that uses the Motorolla HC11 microcontroller. This design makes use of the HC11 analog-to-digital converter and the serial subsystems. Temperature measurement and display circuits were built and control software was written to use the added hardware. In this design, the overall objectives were met. By keeping track of the measured temperature, the HC11 is able to control a temperature display that uses light emitting diodes. Also, if the temperature becomes very cold or hot, an alarm message is sent to a host PC terminal. This design has many potential applications, including temperature control and factory automation.

Introduction

This report presents a design of a temperature measurement and display system that incorporated the Motorolla 68HC11 microcontroller, simply referred to here as the HC11. This design made use of the HC11's analog-to-digital (A/D) converter and the serial subsystems.

As shown in Figure 1, the design included a temperature sensor connected to one of the HC11's A/D input pins on Port E, and light emitting diodes (LEDs) connected to Port B. These LEDs acted as temperature indicators. Additionally, the design included a connection between the HC11 and a remote personal computer (PC). This connection served to send messages regarding temperature to the PC. An assembly software program developed for this design performed various functions for using the added hardware.

The design had two main objectives. The first objective was to use the HC11 to measure temperature. Included in this objective was the task of connecting the temperature sensor and the LEDs to the HC11. Also included in this objective was the task of designing software to do the following: initialize the A/D converter and serial subsystems; control the measurement and storage of temperature in a RAM variable called TEMP; and control the display of temperature on the LED outputs. The second objective of the design was to use the HC11 to indicate if the temperature went outside of prescribed limits: below 20 degrees Fahrenheit or above 90 degrees Fahrenheit. Included in this objective was the task of connecting the HC11 to a remote PC terminal through an RS-232 connection. Another task within this objective was developing software to initialize the serial subsystem. The final task of this objective was to create subroutines for the software program of the first objective to have the HC11 send a message to the PC if the measured temperature went outside the stated limits.

In performing the testing and design, my laboratory partner and I divided the work in the following way. My partner assumed the lead role in connecting the hardware, and I assumed the lead role in writing the programs. Although one of us had a lead role in performing either the hardware or the software, we worked collaboratively in checking both the hardware and software and in troubleshooting any problems.

This report first presents the procedures for and assessment of the design to have HC11 measure temperature. Then the report discusses the procedures for and assessment of adding a serial output to the HC11 design to communicate whether the temperature is outside of prescribed limits.

{ INCLUDEPICTURE "http://www.writing.eng.vt.edu/workbooks/pictures/fig1.gif" * MERGEFORMATINET **}**

Figure 1. Temperature measurement and display system developed for the Motorolla 68HC11 microcontroller, which is attached to a universal evaluation board (EVBU).

Connecting a Temperature Measurement Circuit to the HC11

Connecting a temperature measurement circuit to the HC11 microcontroller involved both hardware and software. Hardware was added to control the measurement and display of the temperature. This hardware included a temperature sensor attached to Port E and LEDs attached to Port B. The circuit was designed according to the specifications obtained from the Computer Engineering Laboratories web site for ECPE 4535 [Lineberry, 1998]. For the complete temperature measurement circuit, see Appendix A.

Procedures for Design. Within the circuit was an LM3911 temperature controller integrated circuit (IC), the output of which we connected to a non-inverting op-amp. The output of this op-amp attached to the HC11 A/D input pin E2 through a 1000-ohm resistor. The circuitry was scaled so that 0 volts out corresponded to 0 degrees and 5 volts out corresponded to 110 degrees. To each of the output pins of Port B, we connected LEDs using a 74HC244 buffer IC and 330-ohm current limiting resistors, as shown in Appendix A. The LEDs were located in the breadboard area of the trainer kits.

To control this added hardware, we programmed the HC11 following the pseudo code and program listing given in Appendices B and C, respectively. The program shown in Appendix C consisted of three subroutines that were called from Main. The three subroutines were named Startup, GetTemp, and SetDisp. The Startup subroutine was used to enable the A/D converter subsystem. First the A/D charge pump was powered up by setting bit 7 of the OPTION register, and bit 6 was cleared so that the charge pump used the system E-clock. After a 100 microsecond delay to allow the charge pump to stabilize, the control word $22 was written to the ADCTL register to start continuous, single-scan conversions on Port E pin E2.

The subroutine GetTemp was used to input and scale the analog voltage from the temperature sensor circuit. The register ADR3 held the result of the A/D conversions, which was treated as an 8-bit binary fraction between 0 and 1. This value was loaded into accumulator A and then multiplied by a scale factor of 110 using the MUL instruction. The result of this multiplication is a 16-bit number between 0 and 110, with an 8-bit integer portion stored in accumulator A and an 8-bit fractional portion

stored in accumulator B. The integer portion of the temperature was then stored in the RAM variable TEMP.
The subroutine SetDisp controlled the lighting of the LEDs connected to Port B. The amount of lighting was based on the present value of TEMP. First, TEMP was loaded into accumulator A and compared with the value 20, the designated cut-off for low temperature. Accumulator B was cleared to zero and represented the initial value to be written to Port B. If the value in accumulator A was greater than or equal to 20, then the value in accumulator B was shifted one position left and incremented, and 10 was subtracted from accumulator A. The process then repeated itself as long as the value in accumulator A was greater than or equal to 20. An abbreviated form of this process appears in Figure 2 (the complete process appears in Appendix C). After the number of LEDs to turn on were determined, as shown in Figure 2, the number of bits indicated by the count value in accumulator B were set high on Port B beginning with bit 0.

{ INCLUDEPICTURE "http://www.writing.eng.vt.edu/workbooks/pictures/fig2.gif" * MERGEFORMATINET **}**

Figure 2. Flowchart illustrating the determination of the number of Port B bits to enable for the LED display.

Assessment of Design. To test the operation of the GetTemp and SetDisp subroutines, we measured the actual temperature with a temperature probe and compared that with the measured value represented by the LED display indicators at several different temperature settings. Table 1 shows the results of the measurement comparison, where the actual temperatures measured are shown on the left, and the temperatures represented by the number of LEDs lit are shown on the right. From Table 1, we verified that the developed hardware and software for this part of the lab were functioning properly. Overall, this section of the laboratory went smoothly.

Table 1. Comparison of temperature measurements.

Actual Temperature	**Number of LEDs Lit**
15°F	**0**
28°F	**1**
33°F	**2**
56°F	**4**
110°F	**8**

Adding Serial Output to the HC11

This section presents the addition of four subroutines to the existing software developed in the previous section. The added subroutines, listed in Appendix D, were called InitSCI, SendChar, SendMsg, and CheckLimits. The InitSCI subroutine initialized the serial subsystem of the HC11 so that it could communicate with the host PC at 9600 baud. This initialization was done by writing control words to the

BAUD, SCCR1, and SCCR2 control registers in the HC11 as shown in Appendix C.

Procedures for Design. The subroutine SendChar was added to send a single data byte from the HC11 to the remote PC terminal. The data byte to be sent was contained in accumulator A. After waiting for the TDRE bit in the SCSR register to be set, indicating that the HC11 is ready to transmit another byte, the value in accumulator A was written to the SCDR register to begin the transmission.
The subroutine SendMsg used the SendChar subroutine to write character strings to the remote PC terminal. Before calling SendMsg, the X index register was set to point to the beginning of the character string to be sent. The SendMsg subroutine then sent out the string by calling SendChar for each character until the NULL character was reached, which marked the end of a string.
The third and final subroutine CheckLimits was added to the existing software program to check the temperature range. The subroutine CheckLimits called SendMsg to print the following message if TEMP was less than 20 degrees Fahrenheit: "Temperature is very low." If TEMP was greater that 90 degrees Fahrenheit, CheckLimits called SendMsg to print the following message: "Temperature is very high." If TEMP was between 20 and 90 degrees Farenheit, CheckLimits called SendMsg to print the following message: "Temperature is acceptable." A flag variable called FLG ensured that the messages were not repeatedly sent for each entry into the very hot, very cold, or acceptable temperature regions. FLG was set to zero if TEMP was between 20 and 90 degrees, one if TEMP was less than 20 degrees, and two if TEMP was greater than 90 degrees.

Assessment of Design. While developing the design presented in this section, several mistakes and difficulties were encountered. The initial setup of the serial subsystem of the 68HC11 involved some troubleshooting. We also had problems with sending the alarm messages more than one time because a flag variable was not set. The diagnosis and solutions to these problems are discussed in this section.
Initially, the serial writes from the 68HC11 to the host PC did not work properly because the SendChar routine did not check the TDRE bit before writing to the SCDR register. This caused characters to be dropped when sending a message. We also had a problem sending out messages using SendMsg because we did not terminate the message strings correctly with the NULL zero. By adding the NULL zero to the end of the strings, the sending of messages worked as expected.
A final problem was the output rate of the alarm messages. At first, we did not set a flag to indicate to the program that a message had already been sent to the PC. This failure caused messages to be continually sent to the PC terminal when the temperature was outside of the normal operating region. This problem was fixed by making a variable called FLG that was set as soon as the alarm message was sent and then cleared when the temperature returned to the normal operating region.

Conclusions

This report has discussed the development of a temperature measurement and display system. The objectives of this lab were to develop the necessary hardware and software to have the HC11 measure temperature and indicate whether that temperature fell outside of prescribed limits. Both objectives were met. By keeping track of the measured temperature, the HC11 was able to control an LED temperature

display. Also, if the temperature became very cold or hot, the HC11 sent an alarm message to a host PC terminal.

This lab has introduced us to the important topics of A/D conversion and serial communications. In the lab, an A/D converter allowed us access to analog inputs of temperature from a remote computer. Besides temperature measurement, A/D converters have many applications in automatic control systems and factory automation. For example, in an electric motor drive, the phase currents and flux are continually measured by using scaling circuitry and an A/D converter input to a microprocessor.

>

References

Lineberry, Bob, "Computer Engineering Laboratories Website at Virginia Tech," *http://www.ee.vt.edu/cel* (Blacksburg, VA: ECE Department, 1998), ECpE 4535: Laboratory Assignments, Lab X.

Motorola Corporation, *M68HC11 E Series: Reference Manual,* rev. 3 (Oak Hill, Texas: Motorola Corp. 1991), p. B-5.

Spasov, Peter, *Microcontroller Technology: The 68HC11*, 2nd ed. (Englewood Cliffs, NJ: Prentice Hall, 1996), p. 107.

Appendix A: Hardware Schematic

Figure A-1 presents the hardware schematic for the temperature circuit. The circuit was designed according to the specifications obtained from the Computer Engineering Laboratories web site for ECPE 4535 [Lineberry, 1998].

{ INCLUDEPICTURE "http://www.writing.eng.vt.edu/workbooks/pictures/figa1.gif" * MERGEFORMATINET **}**

Figure A-1. Hardware schematic for the temperature measurement circuit designed for this lab. In an actual report, all the connections, pin numbers, and pin labels should be shown.

Appendix B: Pseudocode for the Software Developed

XXXXXXXXXXXXXXXXXX*
XXXXXXXXXX
XXXXXXXXX
XXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXX
XXXXXXXXXXXX
XXXXXXXXXXXXXXXXX
XXXXXXXXXX
XXXXXXXXX

XXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXX
XXXXXXXXXXX

*In an actual report, the pseudocode would appear here. Also note that some professors allow you to substitute **{** HYPERLINK "http://www.writing.eng.vt.edu/workbooks/appendixb.html" **}**with program flow charts for this appendix.

Appendix C: Program Listing