

**ECE 3534 Microprocessor System Design
Spring 2002
Laboratory Assignment 3 Report Cover Sheet**

Name: _____ **Student ID:** _____

Name: _____ **Student ID:** _____

Instructor: _____ **Class Meeting Time:** _____

Name of Primary Author of this lab report: _____

Pledge (signed by all lab partners): I have neither given nor received unauthorized assistance on this assignment.

Signed: _____

Grading: Both the Communication Skill and the Technical Merit will be graded on a 100 point basis.

Communication Skill Grade _____

Strengths	Errors in Style	Errors in Style	Errors in Form
Depth achieved	Outline not followed	Ambiguity	Format not followed
Paragraphing effective	Depth lacking	Discontinuity	Run-on, Fragment
Key terms defined	Section transition abrupt	Stagnant sentence rhythms	Punctuation error
Transition words used	Imprecision	Illustration not introduced	Incorrect verb tense
Sentences varied	Improper tone	Illustration not captioned	Unclear pronoun ref
References support	Needless complexity	Illustration misplaced	_____

Technical Merit Grade: _____

- Successful validation(s) (partial credit may be given) (30 points).
- Completed comment sheet (5 points)
- General *technical* discussion of the lab requirements and how you did or did not achieve them (15 points).
- Design of lab software (25 points): general discussion of any algorithms used and their operation, inclusion of software development support (pseudo code, flow charts, etc.), inclusion of program listing files (*.LST files), software documentation.
- Design of lab hardware (25 points): general discussion of hardware designed for the lab, inclusion of H/W schematics.

Contents

Introduction	1
Connecting a Temperature Measurement Circuit to the HC11	2
Procedures for Design	2
Assessment of Design	4
Operating in Standalone Mode	4
Procedures for Design	4
Assessment of Testing and Design	5
Conclusions	5
Appendix A: Detailed Hardware Schematic	7
Appendix B: Pseudocode	8
Appendix C: Flow Charts	9
Appendix D: Assembled Code	10
References	11

>
>
>
>
>

Introduction

This report presents a design of a temperature measurement and display system that incorporated the Motorola 68HC11 microcontroller, simply referred to here as the HC11. This design was a valuable experience because similar temperature measurement and display systems often are used in buildings and vehicles [Spasov, 1996]. The design presented in this report 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.

>

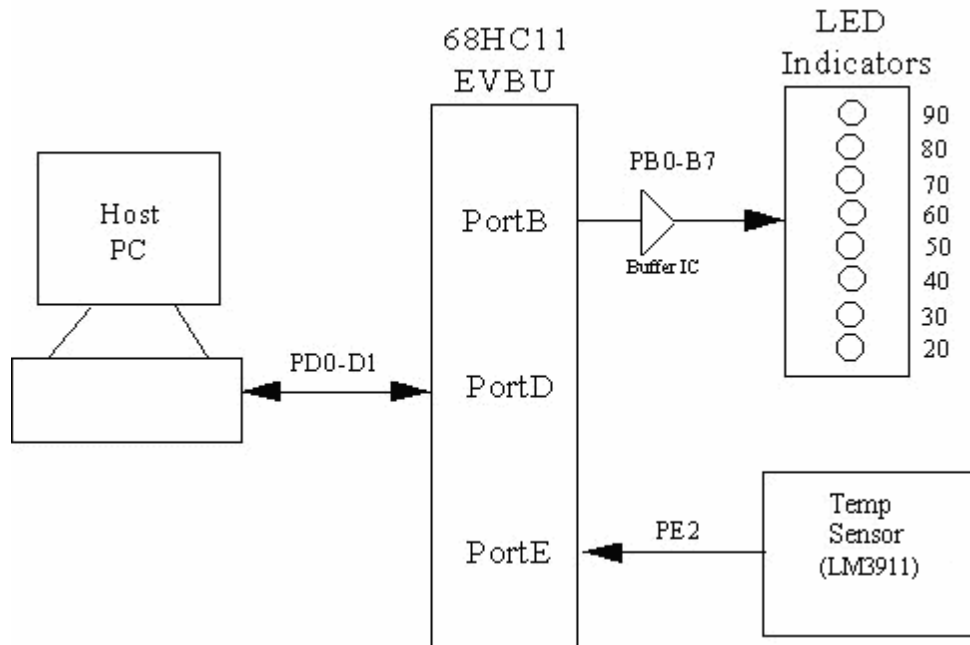


Figure 1. Temperature measurement and display system developed for the Motorola 68HC11 microcontroller, which was attached to a universal evaluation board (EVBU).

>

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.

This report first presents the procedures for and assessment of the design to have the 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.

>
>
>

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. Software served to control this added hardware. In performing the testing and design for this part of the 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.

>
>

Procedures for Design

>

The hardware for the temperature measurement circuit included both a temperature sensor attached to Port E and LEDs attached to Port B. The circuit, which is shown in Figure A-1 of Appendix A, was designed according to the specifications obtained from the Computer Engineering Laboratories web site for ECPE 4535 [Lineberry, 2004].

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, all of which are shown in Figure A-1. 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 the main program (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. Then bit 6 was cleared so that the charge pump used the system E-clock [Spasov, 2002]. 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 pin E2 of Port E.

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. This result was loaded into accumulator A and then multiplied by a scale factor contained in accumulator B by using the MUL instruction. The result contained in accumulator A was then right shifted once giving the temperature in degrees Fahrenheit, which was then stored in the RAM variable TEMP. The equation for finding TEMP is shown in Equation 1:

$$\text{TEMP} = \text{ADR3} (\text{Scale Factor} \gg 1) \quad (1)$$

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 count value for the number of LEDs to turn on. If the value in accumulator A was greater than or equal to 20, then the count in accumulator B was 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.

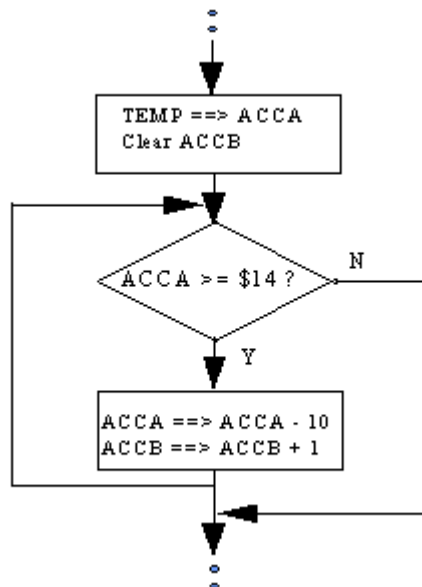


Figure 2. Flowchart illustrating the determination of the number of Port B bits to enable for the LED display.

>
>

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.

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 three subroutines to the existing software developed in the previous section. The added subroutines, listed in Appendix D, were called OUTSCI, MESSOUT, and TEMPCHK. In addition to adding these three subroutines, the subroutine STARTUP, used in the previous section, was modified to initialize the serial subsystem of the HC11 so that it could communicate with the host PC at 9600 baud. This modification was done by writing control words to the BAUD, SCCR1, and SCCR2 control registers in the HC11 as shown in Appendix D.

>

>

Procedures for Design

>

After modifying the STARTUP subroutine, we added the OUTSCI subroutine to write data bytes from the HC11 to the remote PC terminal. The data byte to be sent was contained in accumulator A. The subroutine checked that the TDRE bit was set in the SCSR register, and if it was, the data byte in accumulator A was written to the SCDR register. If the TDRE bit was not set, the program continually read the SCSR register until bit TDRE was set before writing accumulator A to the SCDR register.

The subroutine MESSOUT used the OUTSCI subroutine to write character strings to the remote PC terminal. Before calling MESSOUT, the X index register was set to point to the beginning of the character string to be sent. The MESSOUT subroutine then sent out the string by calling OUTSCI for each character until the NULL character was reached, which marked the end of a string.

The third and final subroutine TEMPCHK was added to the existing software program to check the temperature range. The subroutine TEMPCHK called MESSOUT to print the following message if TEMP was less than 20 degrees Fahrenheit: "Temperature is very low." If TEMP was greater than 90 degrees Fahrenheit, TEMPCHK called MESSOUT to print the following message: "Temperature is very high." A flag variable called FLG ensured that the messages were not repeatedly sent for each entry into the very hot or very cold temperature regions. FLG was set to zero if TEMP was between 20 and 90 degrees and to one otherwise.

>

>

Assessment of Testing and Design

>

While performing the preliminary design presented in this report, 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 SP was not initialized and the data string was not terminated properly. With an incorrect SP, the function calls to the OUTSCI subroutine did not work as expected, and in fact caused the program to crash. By correctly loading the SP, the OUTSCI subroutine worked as expected. We also had a problem sending out messages using MESSOUT 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.

>

>

>

Conclusion

>

>

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.

Appendix A: Hardware Schematic

This appendix presents in Figure A-1 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, 2004].

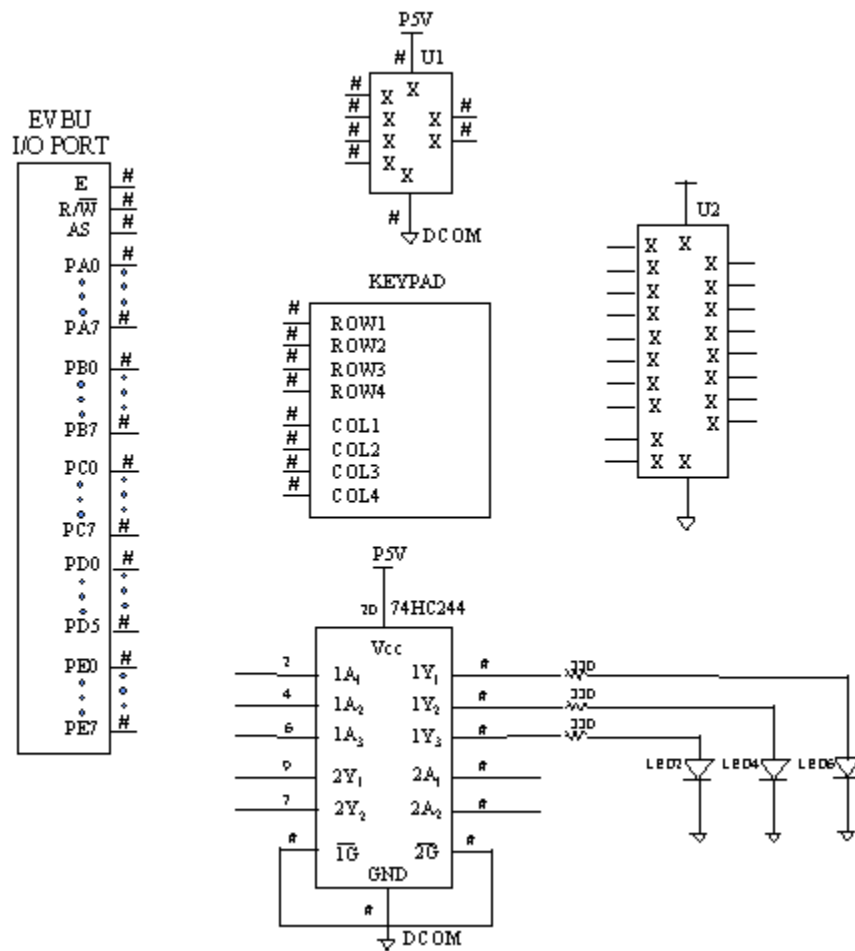


Figure A-1. Hardware schematic for the temperature measurement circuit designed for this lab. In an actual report, all the connections, pin numbers, and pin labels should be shown.

Appendix B: Pseudocode for the Software Developed

Pseudocode for Section 1

```
Call the initialization subroutine
Loop forever the follow subroutine calls
    Call the temperature subroutine
    Call the display subroutine
    Call the range subroutine
End loop
```

Pseudocode for Section 2

```
Temperature Subroutine
    Get the temperature from Port E
    Store the temperature in the variable TEMP
End Subroutine
```

```
Display Subroutine:
    If TEMP < 15* F
        0 LEDs are lit through Port B
    If 15* F < TEMP < 28* F
        1 LED is lit through Port B
    If 28* F < TEMP < 33* F
        2 LEDs are lit through Port B
    If 33* F < TEMP < 56* F
        4 LEDs are lit through Port B
    If 56* F < TEMP
        8 LEDs are lit through Port B
End Subroutine
```

```
Range Subroutine:
    If TEMP < 20* F
        Message 'Temperature is too low'
    If TEMP > 90* F
        Message 'Temperature is too high'
End Subroutine
```

Notes to Students:

1. Pseudocode should be done *before* coding in assembly.
2. Variable should be used instead of accumulators (ACCA) or register names (IX).
3. Port names are acceptable, because that is the only way to represent them.
4. Initialization subroutine should not contain stack pointer calls or special registers (PACTL).
5. It is acceptable to use *if-else*, *while*, and *for* statements as long as you donot use C++ or other software language specific syntax. Thus, your pseudocode should be language independent.

Appendix C: Program Flow Chart

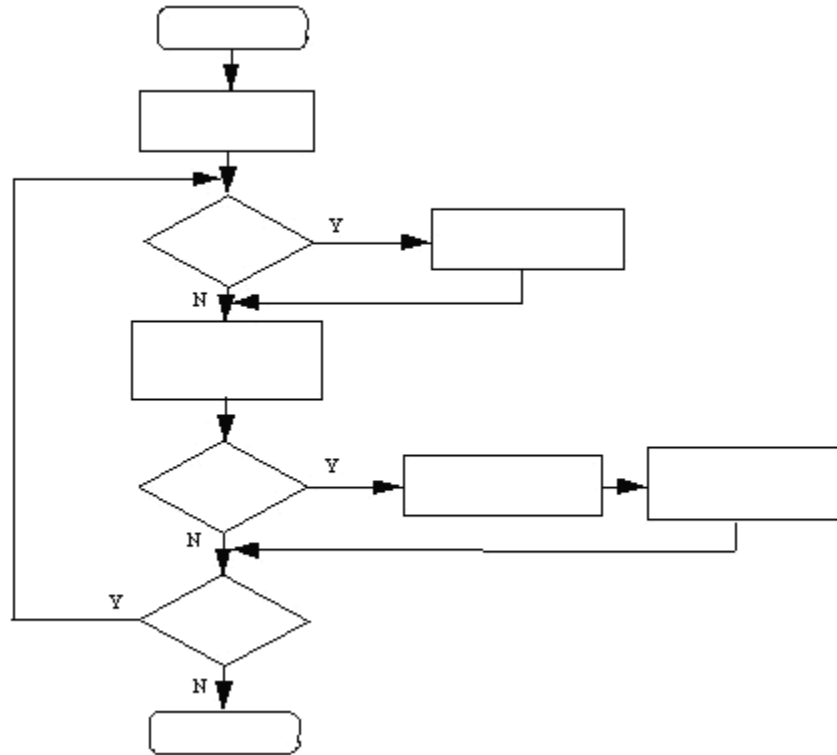


Figure C-1. Flow chart for the lab program. In an actual report, you would fill in the blocks.

Appendix D: Program Listing

```
*****
Program Header
*****

*****
include statements
*****

*****
Equate Statements
*****

*****
RAM var defs
*****

.
.
    ORG $B600
    LDS #$1FF

    JSR  STARTUP    ;Initialize A/D and SCI,
                    ;init RAM vars
MAIN JSR  GETTEMP  ;Set RAM var TEMP to
                    ;current temperature
    JSR  SETDISP   ;Update LED display
                    ;Check to see if hot or
    JSR  TEMPCHK   ;cold message should be
                    ;sent
    BRA  MAIN      ;Repeat

.
.
*****
Subroutine Listings
*****

*****
Interrupt Service Routines
*****
```

References

Lineberry, Bob, "Computer Engineering Laboratories at Virginia Tech,"
<http://www.ee.vt.edu/cel> (Blacksburg, VA: ECE Department, 2004).

Spasov, Peter, *Microcontroller Technology: The 68HC11*, 3rd ed. (Englewood Cliffs, NJ: Prentice Hall, 2002).