|  |
| --- |
| http://www.intersectionconsulting.com/wp-content/uploads/2011/03/UBC-logo.png  University of British Columbia Faculty of Applied Science Department of Electrical and Computer Engineering EECE 281 – Design Project I |
| DESIGN AND DEVELOPMENT OF |
| AUTOMATED REFLOW OVEN CONTROLLER |
|  |
|  |
|  |

|  |  |  |
| --- | --- | --- |
| Date of Submission: | Monday, February 24, 2014 | |
| Submitted to: | Dr. Jesús Calviño-Fraga, Ph.D. | |
| Submitted by: | GROUP BB  Derek Chan  Nina Dacanay  Aleksander Dordzijev  Glyn Han  Jessica Hua  Kyujin Park | 12345678  21848122  12345678  21847124  12345678  24898132 |

**Table of Contents**

1.0 Introduction 1

1.1 Specifications 1

1.1.1 Software Specifications List of files & a brief summary of each file function: 2

1.1.2 Hardware Specifications 5

2.0 Investigation 7

2.1 Idea Generation 7

2.2 Investigation Design 7

2.3 Data Collection 8

2.4 Data Synthesis & Analysis of Results 9

3.0 Design 10

3.1 Use of Process 10

3.2 Need and Constraint Identification 11

3.3 Problem Specification 11

3.4 Solution Generation 12

3.5 Solution Evaluation 12

3.6 Detailed Design 12

3.7 Solution Assessment 12

4.0 Life – Long Learning 12

4.1 Software 12

4.2 Hardware 13

4.3 Management 13

References 14

Bibliography 15

Appendix iii

# 1.0 Introduction

Reflow soldering is a procedure commonly used to attach surface mount devices (SMDs) into printed circuit boards (PCBs). Solder paste, a mixture of solder flux and pellets, is used to temporarily secure the components onto the solar pads on the PCB, which is then placed in an oven for controlled heating. This method requires a strict control of time and temperature parameters to allow the flux to activate and the solder paste to melt without burning the PCB or damaging the sensitive electronics on the board. Given the complexity of manually controlling different parameters, including timing and temperature, which are involved in reflow soldering, the objective of this project is to design, build, program, and test a system controller that automatically controls a reflow oven for PCB reflow soldering while providing the user with a detailed user-interface and status control.

## 1.1 Specifications

The specifications for the project given to the group by the instructor of the EECE 281/282 course are as follows: Four selectable reflow profile parameters included: soak temperature, soak time, reflow temperature, and reflow time. These parameters can either be selected from preset values or can be manually adjusted with switches SW8-SW0 and verified by the pushbutton KEY1. Typically in a reflow soldering process, there are 5 reflow states and the profile parameters control these states. These states include: ramp to soak (1-3°C/sec), preheat/soak (60-120 seconds at ~150°C ± 20°C), ramp to peak(1-3°C/sec), reflow (~45-75sec past 217°), and cooling (2-4°C/sec). HEX7-HEX4 displays the total time while HEX3-HEX0 displays the elapsed time for each state. All device to user interactions are displayed through a 2x16 serial LCD display module.

### Software Specifications List of files & a brief summary of each file function:

|  |  |
| --- | --- |
| **main.asm** | Performs the state machine control:  Calls the appropriate functions from each of the files below  Displays the current state via the LEDs |
| **SSR.asm** | Controls the SSR Relay box:    Send the appropriate signal to the SSR (either ON or OFF) |
| **Serial\_Port.asm** | Takes and Reads the temperature from the K-Type thermocouple (connected to the MCP3004)  Displays the temperature to a Python Strip Chart through the Serial Port |
| **Buzzer.asm** | Interface for the Buzzer component |
| **Thermo2.asm** | Loads the variable Temperature\_Measured with the correct temperature in the oven, calibrated with the cold junction outside of the box. |
| **User\_Interface.asm** | 1. Settings Initialization (Settings\_Initialization) called at the beginning of the code, this is the user interface function used to get the temperature settings and stores them into registers.    1. Welcome\_message    2. Soak\_Temperature\_Input    3. Soak\_Time\_Input    4. Reflow\_Temperature\_Input    5. Reflow\_Time\_Input 2. Checking for other inputs (Check\_Inputs)Provides a check to the other user inputs used during the heating process (i.e. force stop / oven open) 3. Status Display / Value Display (Display\_board)Ability to write to the Hex Display / LCD Display and output set messages onto the LCD Display |
| **Door.asm** | Controls the SSR Relay box:  1 if door is open, SSR disable,  0 if door is closed, SSR enable |
| **LCD\_Display.asm** | Write messages to the LCD according to state |
| **Read\_sw5.asm** | Control switches |
| **Thermo\_Python** | Display graph chart on the computer |

Figure : Software Block Diagram

Thermocouple Temperature

(Thermo2.asm)

(

Serial Port

(serial\_port.asm)

(

Interface

(User\_Interface.asm)

(

Toaster Oven

ADC Converter

(adc\_converter.asm)

(

LM335

Room Temperature

Power Control

(SSR.asm)

(

LCD Display

(LCD\_Display.asm)

(

Check Door

(Door.asm)

(

Control Switches

(Read\_sw5.asm)

(

Temperature Chart

(Thermo\_python)

(

Display strip chart

on the computer

State-Machine

(main.asm)

### 1.1.2 Hardware Specifications

In order to measure the temperature inside the reflow oven with a reading range of 0°C and 280°C, a K-type thermocouple with cold junction compensation is used, with thermocouple outputs of 41µV/°C. To amplify the excessively small output, the signal is connected to an OP07 OP-AMP, with 20KΩ and 47Ω resistors. The resistance values for the resistors were chosen using empirical calculations with values obtained based on the thermocouple specification, and tested for accuracy inside the laboratory. The amplifier is then connected to (Talk about how the SSR & PULSE WIDTH MODULATION WORKS.ASK OTHER PEOPLE). The analog signals obtained through the LM355 temperature sensor and from the thermocouple wire are processed through an MCP3004 Analog to Digital Converter (ADC). Then, an NPN 2222A BJT is used to distribute a controlled signal of 5V to the SSR box from the pulse width modulation of the DE2 board. Additionally, serial port communication is used to provide data points for mapping on a strip chart of the temperature and time on a local computer.

Figure : Hardware Block Diagram

ICL 7660

Produces -5V

Altera DE2

Collect Data from Channel 0 and 1

MP3004

Analog data into digitalized form

OP 707

Amplifies the reading input voltage

PN 2222A

Transistor, helps to control SSR Box

K-Type Thermocouple Wire

LM335

OVEN

SSR Box

Provide power

Connected to Channel 1

Connected to Channel 0

Produce 5V Input

# 2.0 Investigation

## 2.1 Idea Generation

In order to efficiently generate ideas and develop a design prototype, our group held regular meetings to discus basic implementation techniques and technology. Prior to our first formal meeting, each member read all the files and watched all the videos given by the course instructor in the project folder. By doing this, we were informed of the project objectives and acquired time to formulate ideas and a working hypothesis of generally how the microcontroller should function and the key elements of operation. Additionally, each member provided an input on other relevant documents and resources that each group member should read prior to the meeting, mostly consisting of previous EECE 281/282 laboratory assignments as well as EECE 259 laboratory assignments and sample solutions / assembly scripts

During the initial meeting, our group first discussed the whole project in detail, outlining each specification and the required general implementation that it entailed. These were listed, and then broken down into subprojects with each subproject given more detailed specification and potential implementation design. Once the project was broken down, each member volunteered decided on subproject(s) that they would like to work on (based on their particular strengths and experience). Through this method, each team member is focused on subproject(s) that are appropriate to their expertise, which was appropriate in the project’s time-constrained schedule. Therefore, each member generated his/her own specific ideas based on the subproject, but the initial broad idea implementation was done on a group level. However our group maintained an open environment, and anyone was welcome to contribute ideas and suggestions to another member’s assigned task. Additionally, some subprojects (hardware design and soldering) were implanted by forming a pair of team members depending on size.

## 2.2 Investigation Design

The design of our reflow oven controller was established through careful research of the specific component data sheets and examination of the project files and lecture slides provided in the UBC connect website for EECE 281/282 course. In addition, we consulted the course materials for EECE 259 (lecture material and Using the MCS-51 Microcontroller) and supplementary files given in the course. Individually, we gathered information for our own subprojects and later discussed our specifications and implementation plan with other group members. Then, each member’s plan specifications was recorded into a file that was stored in an online tool called GitHub (please read section 2.3 for further details), which gave easy access to all the group members for reference. Once the group has heard and approved each proposal, we continued to work on the design until completion. As aforementioned, each design underwent an individual performance test and an analysis whenever it was completed in order to debug the operation and test for extraneous cases. During the testing phase, each member of our group was present to observe and allow individual suggestions and constructive criticism for the design to improve on. Again, these are all recorded and stored into GitHub.

The division of individual tasks definitely made information and data gathering for specific topics more effective. Specifically, Derek Chan was in charge of code organization, unification, state machine design, and integration and specification of subprojects; Nina Dacanay was mainly in charge of the software coding for the temperature sensor, the serial port, and the timers; Jessica Hua and Aleksander Dordzijev were in charge of the software coding for the user interface; while Glyn Han and Kyujin Park were mainly in charge of the hardware assembly. However, even though we were assigned specific parts, each member of the team was exposed to both software and hardware through soldering components onto the PCB and understanding all the codes used for the main program, and regular meetings were in place to ensure each members comprehension of each individual subproject.

## 2.3 Data Collection

Our group utilized a variety of resources to keep our project consistent, organized and efficient, while other tools were used to check the circuitry and debug a code. Such resources included: Github, Facebook, continuity buzzer, voltmeter, oscilloscope and various debugging tools that have been provided. Github is an online hosting code repository service specifically for software development. [1] Our group utilized this popular programming tool to incorporate version control and backups for our projects software portion via cloud based storage. This maintained all our code organized and clean from any data errors while transferring and preserved history of added code and deleted code. Additionally, it provided a convenient platform to track the progress of the project and each user’s input. Facebook, the popular social networking site is where our group had created a separate page where all our group members can see the latest updates on progress, next scheduled working sessions, and any type of issues that may arise from the project.

We used the voltmeter and the continuity buzzer to check the circuit every time we have soldered on specific sections of the circuit. By doing so this allows us to debug the circuit carefully by identifying issues before the entire circuit is assembled. Without testing the connections and the right voltages at certain locations it may cause room for simple errors that would take hours to identify and resolve. As for software debugging tools, our group used CrossIDE and Spyder. CrossIDE is a debugging program developed by our Professor, it debugs our 8051 assembly code. Spyder is an open source integrated development item (IDE) that was provided to us. We utilize it to debug our python codes for the serial port interfaces in order to display and print out our strip chart of the temperature and the time. [2]

## 2.4 Data Synthesis & Analysis of Results

Our group verified the validity of our results by testing the circuits and programs with their theoretical and assumed function. For software, each group member / team constantly debugged and verified that the function of the program was working the way he/she had designed it. Verifying the individual parts of the code made it much simpler to identify the location of the malfunction, and extensive testing before software/state machine integration reduced the issues faced when compiling the entire project. Additionally, a variety of test main.asm files were created to test function calling and integration of files. The validity of the programs we designed were all based on class knowledge and past lab experiences.

As for the hardware and circuitry, their validity was tested through the knowledge we have obtained through our EECE 253 course. We tested the circuits at specific locations and compared the physical results obtained with theoretical results that we had calculated prior to verify the validity of the different sections of the project. The validity of the programs and circuits were also tested by combining smaller parts of the program with the hardware. This allowed verification for the larger portions of the project with abovementioned techniques. Additionally for the end result, when we ran through the entire reflow soldering states, we compared it to the ideal theoretical temperatures and times. After the project was running the way we had planned and designed, we tested it with a PCB and we compared our result with online results and videos that were available to us.

# 3.0 Design

## 3.1 Use of Process

In the early design stages of the project, it was quickly evident that the complete oven controller would require many basic components interacting in a complex manner. Our group recognized the complexity in having many interacting parts and decided to simplify the project through modulation. This meant that we identified each feature of the oven controller and made it into its own module with detailed specifications. We planned to have clear specifications for each module so that when we finally needed to put all the components together, we would be able to easily troubleshoot issues and integrate the features of our modules. Each module would be abstracted so that only two or three functions would need to be used, and the complexity in implementation would be hidden while we combined all of our components together.

## 3.2 Need and Constraint Identification

Each feature that was required in the project was identified as an individual need. The needs we identified were:

1. ability to generate tones/beeps
2. ability to determine if the door was open or closed
3. ability to display text messages for the user via the LCD display
4. ability to transmit serial data to a computer
5. ability to turn the oven on and off using a solid state relay
6. ability to measure temperature in the oven
7. ability to measure time and display time to the user
8. ability to handle user inputs to modify parameters

## 3.3 Problem Specification

From the needs we identified, we created modules that met each need, each with their own specifications. The following list of modules are numbered respective to the needs listed above:

1. Buzzer
   * Generate a continuous tone
   * Generate multiple short tones
2. Door
   * Update a variable according to whether the door is open or closed
3. LCD
   * Write and display specific messages to the user
4. Serial Port
   * Transmit the current temperature through the serial port
5. SSR (Solid State Relay)
   * Enable and disable the oven attached to the SSR
6. Thermocouple Input
   * Measure the temperature in the oven
     + Determine the voltage generated by the thermocouple wire
     + Convert the voltage into a temperature value
     + Determine the ambient temperature (cold junction temperature)
     + Take a difference between the inside and ambient temperatures
     + Store the temperature as a variable
7. Timer
   * Keep a total time for the entire soldiering process
   * Keep a separate process time for separate processes
8. User Interface
   * Take in user 3-digit values for soak and reflow temperatures, and soak and reflow durations, or take the user’s chosen preset values
   * Display back to the users their desired values and times and allow confirmation of values
   * Provide a safety check for the inputted values and determine if they fit safety specifications

## 3.4 Solution Generation

## 3.5 Solution Evaluation

## 3.6 Detailed Design

Each individual component was required to provide a subroutine for initialization and for each specified feature. Please refer to Appendix B for each header file for our components, and Appendix C for the source code.

## 3.7 Solution Assessment

# 4.0 Life – Long Learning

## 4.1 Software

The most useful course was undoubtingly EECE 259 where we learned everything about microcomputers knowledge that we needed for this project. The hardware and software knowledge gained was paramount in completing the project, especially in hardware design and understanding specification, as well as Assembly language programming for the 8051 microcontroller. EECE 251 also played a large role in understanding hardware design specification and the underlying principles in circuit assembly

We spent most of our time and effort into integrating all the functions into one state-machine. The challenge was lack of experience and knowledge for managing complex software systems with multiple components. Additionally, working with a large group of people on a single task also presented management, logistical and integrating issues.

In order to overcome the problems, we learned to use GitHub where we could easily keep track of everyone’s work. We created a header file for each code file, defining its purpose, and its use of variables and registers. Additionally, we referenced the lecture notes which describe to write codes for a state machine. From this, we became more efficient programmers and learned the importance of commenting, specifications, quality control, and good coding principles.

## 4.2 Hardware

EECE 251 was a very helpful course for hardware because we gained an experience with soldering chips on the circuit board and we learned how to check the connection of soldered iron by using the voltmeter. Moreover, for EECE 251 labs, we already designed and built an amplifier and a BJT circuit which were the major part of the hardware. Additionally, the basics of circuit provided the foundation of understanding how the various circuit elements interacted in our hardware.

## 4.3 Management

The group consisted of six people and each member’s task modulated to specific sub-projects. In order to succeed as a group, we spent time creating a schedule and dividing our workload appropriately. Our group focus was time management. We knew in order to finish the project it would require a lot of our time but at the same time, we had other courses to keep up with. Also, we wanted to finish the project at least three days before the deadline. After we were aware of what our goals were, it was easy to be committed in order to reach our goal. We learned to manage our task appropriately to our conditions. Additionally, communication was of paramount importance in completing a large multi-process project. We learned to properly create and follow specification and project management skills, while honing our teamwork abilities and communication skills.

# References

|  |  |
| --- | --- |
| [1] | K. Finley, "Github Has Surpassed Sourceforge and Google Code in Popularity - ReadWrite," Say Media Inc., 2 June 2011. [Online]. Available: http://readwrite.com/2011/06/02/github-has-passed-sourceforge#awesm=~owugMKdmHVVJVl. [Accessed 20 February 2014]. |
| [2] | R. Grehan, "Spyder Review - Techworld.com," International Data Group, 8 September 2010. [Online]. Available: http://review.techworld.com/applications/3238833/spyder-review/. [Accessed 20 February 2014]. |

# Bibliography

|  |  |
| --- | --- |
|  | H. Han - Way, Using the MCS - 51 Microcontroller, New York: Oxford University Press, Inc., 2000. |

# Appendix