Rahil Patel

Introduction

For the final project, we had to write a program for the 6811 that simulated the popular game of hangman.

The program makes use of serial communication, a LCD, an infrared LED detector, a push button, and optionally a potentiometer. The LCD is used to show the word, number of guesses and letter selection. The infrared LED detector and push button are used for selecting a letter.

After the user starts the game, the user selects a letter. Then the 6811 sends the letter to a PC, which contains a complimentary application that determines the number of occurrences the letter has, and the positions in which the letters should be. The program must use this data to place the letters in the correct locations and keep track of the number of guesses. After the game is over, the user may play again.

Design

Most of the specifications of the game were provided or obvious, but the PC programmers and 6811 programmers had to collaborate for a few design components. I was part of the 6811 programmers (non-honors). The main debate was about how to select characters using the infrared LED detector. The professor suggested Morse code, but the students argued that scrolling through characters would be more efficient. Some students also suggested scrolling through characters quicker after a certain amount of time. There were few talks about difficulty and other non-essential elements, but since most of the specifications were given, the PC programmers already had a good process flow of the game.

Work

I started programming by using the homework5 solution as a base. I spent a night just reading, learning, and understanding the code in there. I did not use interrupts for serial communication in homework 5, so the logic took quite some time to understand. After many hours, I was able to transmit a '~' (start the game) and receive a black block character. I couldn't figure out why I was receiving this character, so the next night I proceeded by creating a separate code for scrolling through the alphabet. The scrolling code was completed timely, and I went back to trying to get the receive function to work. After a lot of useless tinkering, a friend found out that the PC sends decimals, and the black block was actually correct.

After passing that road block, it was plain frustrating programming for 15 hours. The following flow chart describes most of the program.

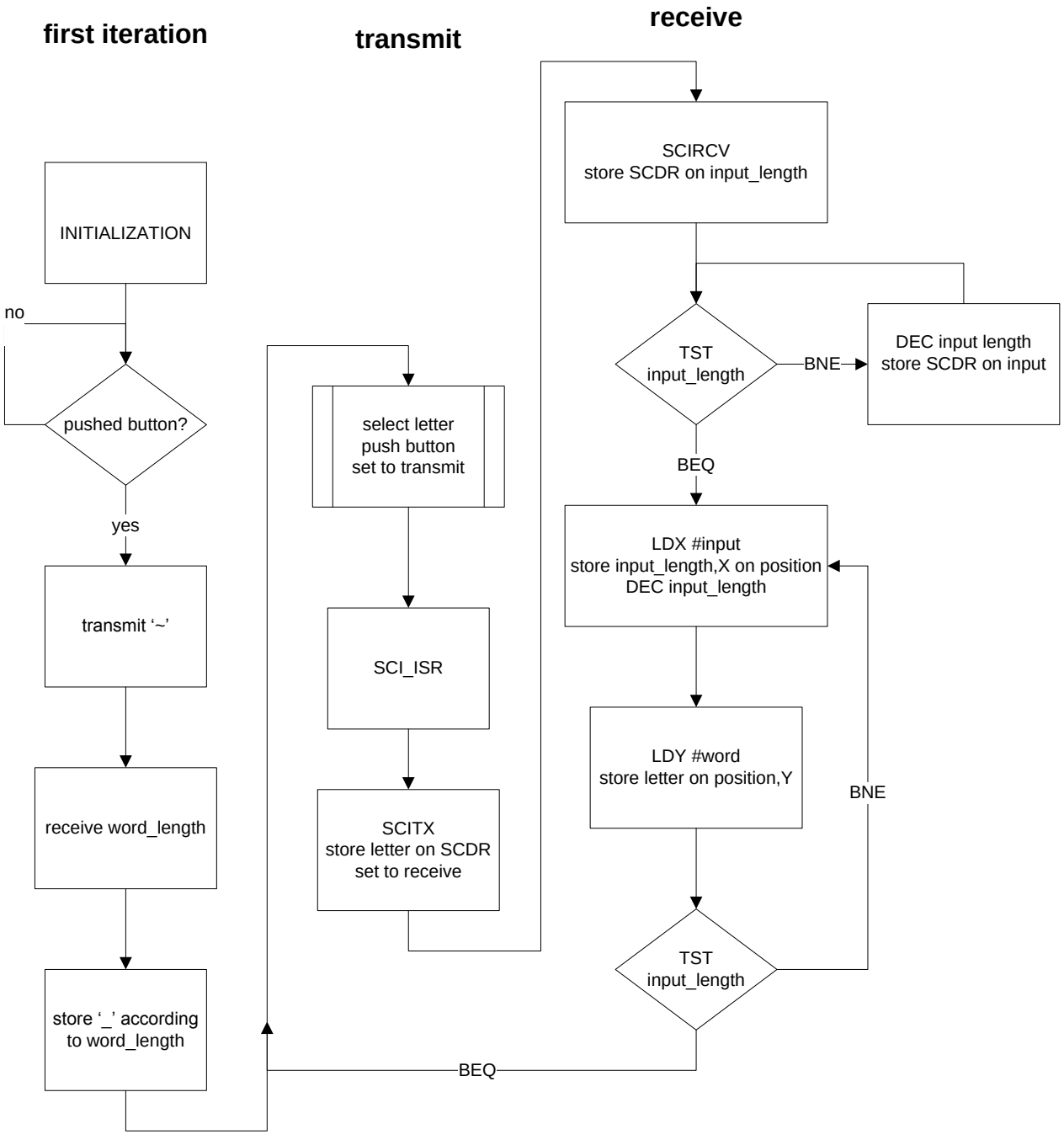

Figure 1. Flow Chart

The core interrupt flow is as follows:

1. push button sets to transmit
2. SCI_ISR checks RDRF/TDRE, goes to SCITX
3. SCITX transmits the character and sets to receive
4. wait for button and go to step 1

Tools

I used the same tools that I used for the rest of the programs, to build this program. This includes the 6811, MiniIDE and the instruction set section of the M68HC11E Series Programming Reference Guide. I tried using Wookie, but did not work well. In addition, I used notepad++'s compare plugin (diff/merge) to quickly compare different versions of my code.

User Guide

This section details the operational parts of the program and instructions to run it.

Operational Parts

As of the time this report was written, the following parts are operational:

- infrared/button letter selection
- transmission of letter
- display of word (blanks and guessed letters appear)

The word display has a bug and may seem random but it is not. The previous letter displays in the next guessed letter's position. I hope to fix this bug before the demonstration.

User Inputs

1. Push the button -> to start a new game (*Note: may have to press once more)
2. Type the secret word and press the enter key -> to send the word to the PC (*Note: if the PC prompts again, repeat this step)
3. Block the infrared beam -> to scroll through the alphabet.
4. Unblock -> to stop scrolling
5. Push the button -> to select the letter
6. Repeat steps 3-5, until you have guessed all of the letters

Observations & Conclusion

I wish I did not use interrupts. For whatever reason, a lot of code just does not work properly in interrupts. It spent fruitless time trying to get very simple code to work. I couldn't even get a Boolean value to work (using TST/INC). I also found out JSR commands crash in interrupts. I would have been much better off building off my own homework5, using pulling. It's less confusion and the code would work without the previously noted problems.

On top of the interrupt code troubles, there were a lot of random hardware and software problems. When I run same program, the outcome would change. One major problem was the USB-to-Serial cable, which would cause problems, making me redo my code several times. Another problem was difference of operating system performance. Windows Vista crashed less and work better overall, compared to Windows XP. This randomness of component failure made the work very frustrating, because there were so many factors that could go wrong. It was best to restrict all work to the Kaufman labs, using a serial-to-serial cable.

It should be noted I was not able to get EEPROM working. The baud rate change in the program has never worked for me. I have to set the baud rate in AxIDE to get the program to run in EEPROM.

Overall, it was terrible compared to the midterm. Nothing ever worked properly the first try. Every baby step had to be tested and debugged, by displaying it on the LCD. I'm guessing a lot of it had to do with the interrupts, but it could have been any of the factors mentioned before. Code just did not work as it did on the midterm.