The Digital Alarm Clock

Rajiv Varma, Maris Prieditis
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: rnvarma@oakland.edu, mmpriedi@oakland.edu

***Abstract*—The purposes of the project were to create a digital alarm clock with a "wave to snooze" feature and also learn more about interrupts and other components of the HCS12. Through hard work and intense coding sessions, we were able to synthesize our project on the Dragon12-Light board.**

I. INTRODUCTION

The scope of this project was to use various C and assembly functions to create a rudimentary digital clock with alarm functionality on the LCD (Figure 1). Additionally, this clock was to have an alarm and silence feature that utilized the HC-SR04 ultrasonic rangefinder (Figure 2) so the user does not have to actually press a button when the alarm is blaring.

This Final Project Report includes three sections: methodology; experimental setup; and, results. After this introduction, the methodology part goes step-by-step through the brainstorming and thinking behind solving the problem of creating a digital alarm clock. This section is segregated into various sectors that look at different aspects of the project beginning with pen-to-paper and ending with coding within CodeWarrior. The experimental setup subsection discusses the expectations and results of testing the project. It therefore touches on the hardware and software tools that we use to verify the stability and fidelity of the system. The results section simply lists all of the results we obtained. Multiple pictures and a video of the running project are included in this section. Finally, the conclusions section wraps up the entire report.

The motivation behind this project was two-fold: we wished to start with a simple project and build-up from it; and, we thought this project would utilize various hardware components from our lab sessions with software techniques that we have learned from lectures. Essentially, our motivation was to use what we learned in ECE 470 and apply it to a real-world problem.

With the success of this project, the implications are that everyone can code a working project with knowledge and learning. With plenty of work, it was possible for us to start with a simple digital clock, but then add complexity to it layer-by-layer. Each level built upon the previous one, with the lower levels acting as supporting pillars in the project. This means that our experience is non-unique: others can replicate our success. As such, it is a great learning tool, and both of us are more knowledgeable because of it.

The topics that we learned in class that applied to our project were the serial and parallel input/output ports, C language programming, interrupts, and timer functions [1]. Also, the lab on the ultrasonic rangefinder was very helpful in our setup [2].

On our own, we had to read through the textbook to find various bits of information on the HCS12 board. We also referenced the instruction manual PDFs and used the Internet to research how to create C functions within the main.c file. Using previous labs, we were able to create combination ASM/C functions suing main.asm. In reality, a lot of this project came down to debug sessions: e.g., testing and retesting until it works.

The ultimate applications of our project include home and business use. However, with further optimization and code-sculpting, this project could be adapted for commercial use as the underlying hardware and code behind a digital alarm clock that's sold to end users.

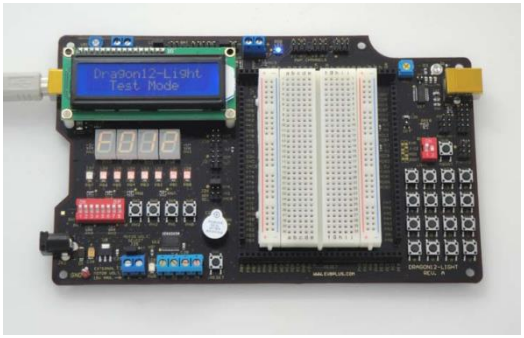

Figure 1: Dragon12-Light Board.

Figure 2: HC-SR04 Ultrasonic Rangefinder.

II. Methodology

This section explains how we designed our project.

A. Statement of Design

Our overall concept for this project was to use interrupts and the timer counter to implement an alarm clock. After connecting the required components to the board (Fig. 3), we used CodeWarrior to write the code. We ended up using multiple nested "if" statements and C functions to achieve success. Our delay function was written in combination ASM/C.

B. Hypothetical and Theoretical Considerations

From the outset, my partner and I thought an alarm clock was an interesting and useful application that could be implemented within the confines of the HCS12 processor and CodeWarrior. From Dr. Llamocca's lectures, we knew that operating and writing to the LCD screen was practical. We also knew about the interrupts and the timer counter from previous labs to use the rangefinder in concert with the LCD. Looking at earlier lectures, we found more information on using the DIP switches. It was actually pretty simple to get our thoughts down on paper and prepare a prototype.

C. Rangefinder

The HC-SR04 rangefinder was used to stop the alarm. We used the code from our Lab #4. The idea behind it was the same as in the lab – make the rangefinder constantly check for a returned distance value and compare that value to a hardcoded value. If the value was within our acceptable range, we would have the system snooze the alarm. In our case, we also made sure the system checked if the alarm was currently alerting the user. In this case, it would snooze; else, it would not report anything or perform any action.

D. LCD Screen

For the LCD screen, we used the provided C functions from Dr. Llamocca to write to and clear the screen. I ended up using a number of for-loops to cycle between hardcoded arrays of numbers and strings.

E. DIP Switches

For the DIP switches, I decided to use switch #1 for user control of the alarm. I also added 2 demo switches at #4 and #8. Switch #4 allows the user to initiate an alarm 5 seconds in the future, while switch #8 drastically speeds up the clock.

F. Overall Hardware Design and Considerations

For the overall hardware design, we figured that if we could start with a basic framework, we could then advance the project from there. Unfortunately, we were not able to implement many of our more imaginative ideas.

III. Experimental Setup

We verified the working condition of the project by implementing a demo switch to speed up the output on the LCD screen. This allowed us to visually show that the clock cycles through an entire day and continues from there. We also used CodeWarrior to debug the program on my Windows laptop. The oscilloscope was used to make sure the timer counter works.

When the system was run, we obtained our expected results. The system was able to trip the alarm 5 seconds after it started (hardcoded to ring at 1:00:05 am) if switch #1 was also "on". Switch #4 was able to change the setAlarmValue for 5 seconds into the future. Switch #8 correctly minimized the elapsed time. The RGB LED and LCD screen both functioned as expected throughout the many tests.

IV. Results

Our prototype board is shown in Figure 3. This includes some rough text on the LCD screen before we finalized it. We also cut down the length of the wires for cosmetic reasons.

Figures 4-7 show the operation of the digital alarm clock. Figure 4 has the clock in normal operation. In Figure 5, switch #1 is "on" so the alarm is set to run by the user. Figure 6 shows what happens when the alarm is alerting the user. Figure 7 shows the response of the system to the user waving their hand in front of the rangefinder.

I also created 2 videos – one of a time-lapse of 24 hours of clock time, and one showing the normal operation of the system.[1]

Overall, the results we obtained were expected from what we coded. We created a simple clock with an alarm component, and our system does in fact function correctly. Of course, that does not mean we are satisfied with our results. There are numerous areas that I wish we had been able to explore and incorporate into our design.

The main area I wish we had solved was the user input section. As it is, there is zero user input for the designated alarm time. I hardcoded the alarm to run 5 seconds after the system starts up. This is a poor implementation, and it is only good for testing purposes. The ideal method would have probably been to utilize the numeric keypad with a menu system on the LCD screen with feedback to the user. In fact, a scrolling LCD screen would have been a very interesting addition to our system.

Obviously, including a sound component into the system would have been optimal. I wish we had been able to at least use the speaker buzzer on the board. Unfortunately, I had trouble coding PT5 within the numerous for-loops, and the results I was getting during debugging were suboptimal.

Ideally, we would have fashioned a more-accurate clock. The reliance on the for-loops plus the functions that run every second of clock-time very likely contributed to a delay of time that accumulated over time. Also, if we had figured out a way to synchronize our clock with GPS, this issue would have been resolved.

Another very interesting idea would have been to add a communications link (likely SCI) between the board and a laptop, such that the user could input a designated clock and alarm time via keyboard or even potentially through the Internet.

---

[1] The videos are included in the Moodle upload folder.

Additionally, the connection of the board to an external light source (perhaps a table lamp next to a sleeping user) could have allowed the system to slowly turn the lights on in place of audibly alerting the user. Another application would potentially be to allow the user to set designated times for the system to turn on and off the lights in their household.

Lastly, since the board was dependent upon power either from the USB or the wall adapter, it would have been a nice challenge to make it self-sustaining, in as much as it could just run from a local battery. This would also stand as a proof-of-concept for a backup power source so that the clock would continue to function in the event of a power-loss.

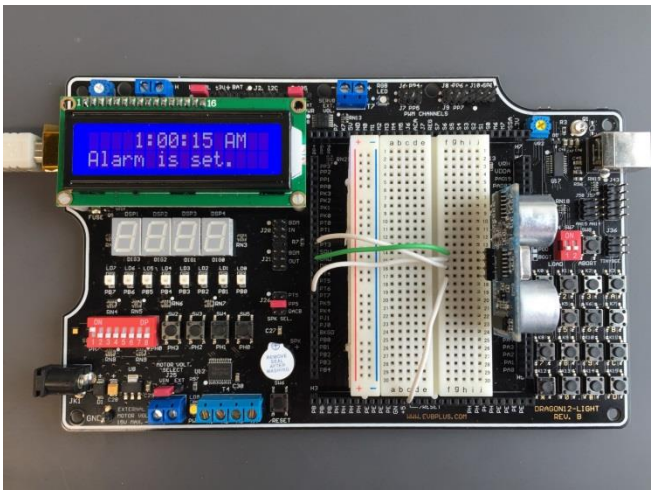

Figure 5: Alarm set via DIP Switch #1.

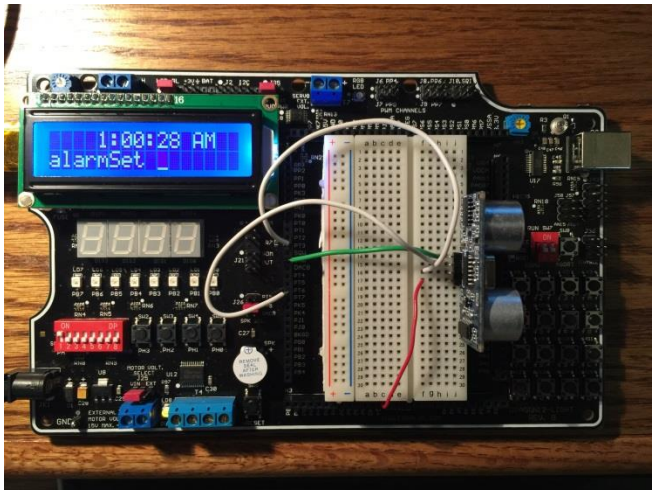

Figure 3: Preliminary clock and alarm.

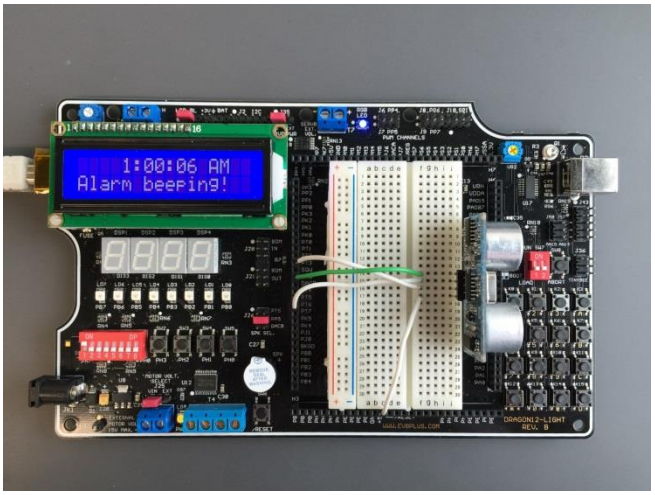

Figure 6: Alarm alerting user via text and RGB LED.

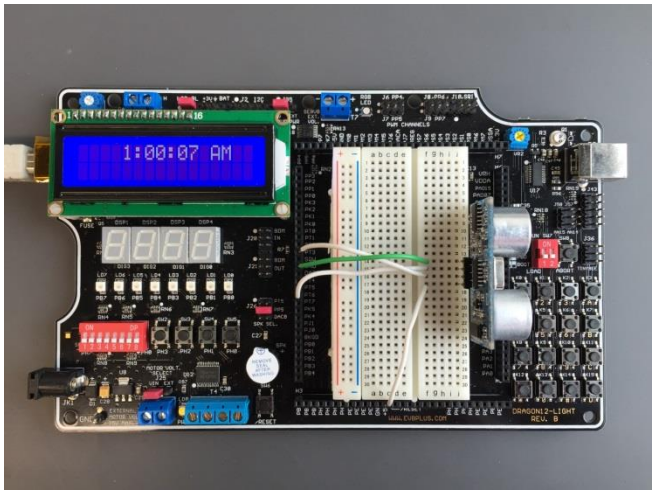

Figure 4: Normal clock operation.

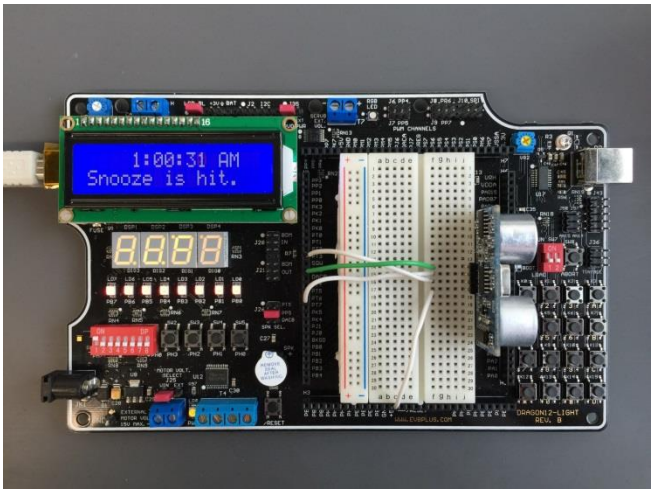

Figure 7: Snooze via rangefinder.

Conclusions

In conclusion, while we were able to produce a working version, there was much left to be done to fully complete the system. The menu system and numeric user input are on the top of that list, and without either of those features, our system is not commercially viable. Listed in the results section are all of our future goals for the project.

However, we did learn a lot about LCD screens, interrupts, and the importance of taking into account all factors of latency (both hardware and software) in a digital system.

In retrospect, though there were some obstacles and unanticipated challenges, I am glad I that I was able to work on this project, and I am happy that it aided me in advancing my education in microprocessor-based system design.

References

[1] ECE 470 Lecture Notes. Llamocca, Daniel. Fall 2014.

[2] ECE 470 Laboratory Experiment Notes. Lorenz, Lincoln. Fall 2014.