# Abstract

This report provides an overview of our Embedded Systems lab’s project. In this project we implemented a stopwatch that allows the users to set a start time. This was implemented on an 8051 Microcontroller. We wrote the program using the assembly language and tested the program on an actual 8051 micro controller after each phase.

# Introduction

Embedded systems are the magic that gives machines that extra technological functionality. For example, advanced cars (BMW, Porsche) braking systems (ABS), security systems, and power plants. These all use embedded systems.

In this project we’ll be using Atmel’s implementation of the 8051 micro-controller. An LCD is connected to port 1 on the chip. The LCD will be used to display the time and a message, which will prompt the user to enter a start time. A keypad is connected to port 2. Push button 0 is the button, which will invoke the procedures that will prompt the user to enter the start time for the stopwatch. Push button 1 will restart the stopwatch and flash a green LED. Note that we are using interrupt based programming. Meaning, both push buttons will send an interrupt signal.

# Hardware Design

The hardware that we used:

1. An Atmel Microcontroller chip.
2. A keypad.
3. An LCD display.
4. A green LED
5. 2 push buttons

This is all included in the **MicroTRAK/51-C2** kit, which we used.

# Hardware Testing

Hardware testing consists of two parts:  
 1. Testing the kit for power.  
 2. Displaying a string of characters on the LCD to make sure it is connected properly.

A kit was provided by the Engineer to make sure that the micro-controller chip is properly connected. The LCD should display a string if everything was connected properly. Also, if the string was displayed that meant the power in the kit is at optimal value and fully functional.

# SOFTWARE DESIGN:

The design of the software can basically be divided into 3 parts:

1. Designing the push buttons interrupt service routines to deal with the required functionality.
2. Designing the Timer interrupt service routine to update the stopwatch regularly. In other word, this will make it go “tick-tock”.
3. The main and the start of the program. This contains the initialization, and configuration of all of the required peripherals and components (LCD, Keypad, interrupt enable, etc.).

# SOFTWARE TESTING:

We tested the software in 2 ways. One of ways was using the micro-controller IDE. While this was inefficient, it was all we had. This was all we had when we were working on the project during the weekends. Unfortunately, the labs are closed on the weekends so we had no access to the micro-controller and we were left with nothing but the IDE. The other way was downloading our program onto the micro-controller. Of course this was when we had access to an actual micro-controller. Our main strategy was trial and error; try to get it to work until it starts working.

# FINAL TESTING OF THE HARDWARE, SOFTWARE AND INTERFACING

In the final testing we tested all the functionalities. Actually it is hard not to test them all when mostly all of them are connected. We pressed each button, one after the other and vice versa. Meaning we restarted the kit and tried to press them in reversed order.

# How to Run The Stopwatch

First, connect the kit to the power supply. Second, wait for the kit to initialize if needed this depends on the speed of the kit being used. Finally, you can just look at the LCD as it displays time or you can press push button 0 to restart the stopwatch or alternatively you can press push button 1 to set a new time.

# Conclusion

The assembly language is not best language to work with. Especially when the program is known to grow large in instructions. But using procedures and interrupt based programming as opposed to polling based programming will surely make your life and the task at hand easier. Also, thinking, planning, and using tools such as flowcharts before starting the programming stage will make the program shorter in lines of code and more readable. Furthermore, it will also make the debugging process easier

# Appendix A: Flowcharts

# Appendix B: Code