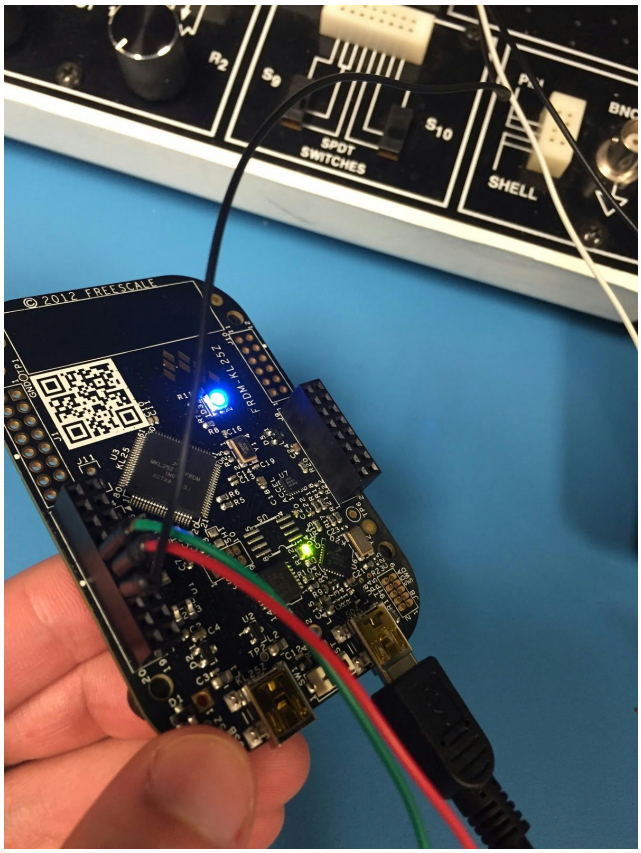


Platinum - Final Project

Project Report - Embedded Systems

CSE 336

Rafael de Oliveira
Tobin Dewey
Matt Amoroso

Table of Contents

1. Introduction

This project was designed for ECE 336 - Embedded Systems. A skeleton code was given to us to be modified and extended with additional functionalities. The IDE use for this project is called CodeWarrior v10.6, which was published by Freescale Semiconductor for editing, compiling, and debugging software for several microcontrollers, microprocessors and digital signal controllers used in embedded systems. The board used for this project was the KL25Z board provided in class and the program developed by our group was written in C.

1.1 Project Features

- Program keeps the chip clock at 48Mhz.
- Uses the UART0 module through TeraTerm to communicate with a PC.
- Sets the terminal for 115,200 baud and utilizes Xon/Xoff protocol.
- Through the use of the TPM module, the program is able to create pulses to run a servo for external control. These pulses are ranged from 1msec to 2 msec (the range of the servo) in width with a 18-20 msec repeat period.
- Program parses commands from the terminal program, which are coming from the PC. A command to report back the status of the measured pulse widths to the PC has also been added.

2. Description of the Program

2.1 Program Functionality

The project's goal was to control a servo tower by talking to the KL25Z chip through a terminal of our choice. We decided to use TeraTerm, since it was simpler to work with and proved to be a very effective choice. The program works by increasing and decreasing the modulus that sets the pulse width that is sent to the servo tower. This was accomplished by the use of a mixture of imperative style programming in C and embedded systems programming to parse commands from the TeraTerm terminal through the UART0 module. The KL25Z board would then be wired to a breadboard and the servo tower.

Two separate pulse widths were used to control both the X and Y axis on the servo tower. We also found it useful, even though it wasn't a necessary step, to wire an oscilloscope to the board so as to obtain a visual depiction to the pulses controlling the tower and, therefore, check our results with what was expected.

2.2 Using the Program

To make the servo tower look up, the user should press the button "1". To make it look down, the user should press the button "2". The user is also able to make the tower look right and left by pressing "Q" and "W" respectively. What those inputs are doing is raising and lowering the modulus by a specific amount to make the servo move 1 degree in the desired direction. The servo starts at 1 millisecond on each pulse width, which puts it all all the way down and all the way to the left. The included also checks to make sure the user does not try to push the servo past its limits, which could result in the servo breaking its own gears. A README file has been included with this project as a simple guide on how the user can use the program.

2.3 Next Steps

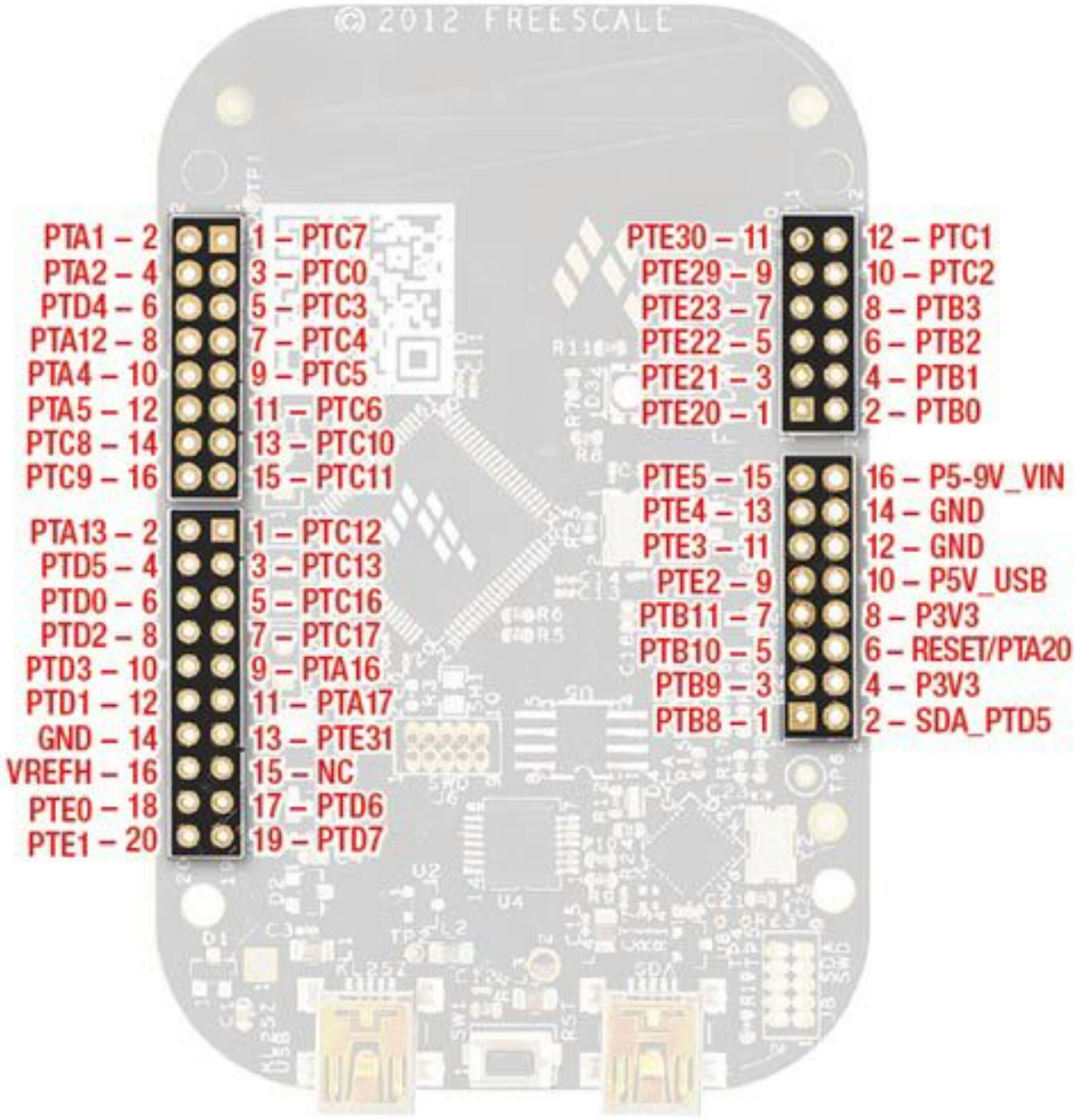
The next steps in this project involve a multitude of different processes:

The first step would be to make the laser working on the servo tower. This task would be done by using another pulse to turn the laser on or off when specified in the terminal.

The next step would be to improve the parsing algorithm and how the program is used. This step would involve in creating a more user-friendly interface, with the usage of widgets, such as buttons. Other routine specifications would also be added in the future, such as the specification of multiple locations in terms of the X and Y axis that the servo tower could go to. Then, the tower would wait for the user to give the command telling it where it would go. The tower would

travel to each of the specified locations and perhaps fire a laser after pausing at each spot for a user determined number of seconds.

The final step would test the project limits by trying to include a new peripheral on the servo tower, a good example would be a small cart with wheels that could move it around, or even a camera that would allow us to give the program the ability to choose where to aim the servo tower using image processing techniques. A good way to implement this would be using the OpenCV library available for C.



4. Summary

The process of adapting the platinum code and combining the KL25Z board with external hardware and software was very insightful for us. It helped build our understanding of embedded systems, interrupts, and parsing. We also got a more clear understanding of how to use terminal programs such as PuTTy and TeraTerm, how to manage memory, and how to use an oscilloscope.

Overall, this project gave us a wider perspective on embedded systems programming and also improved our C programming skills. Given more time, we would have liked to be able to implement more functionalities into our program, but with the time given, we managed to widen our horizons.