Report Outline : Feedback Controls

Final Project

Group Members: Ross Mitchell, Krista Stone, Matt Foster, Erin Wiles

Class: CSCI 255 Introduction to Embedded Systems

Date: 12/15/2014

“The final project will serve as the final exam and will be worth 15% of your total grade.

In order to complete the final project there are three deliverables: Each of the three portions mentioned 1/3 of your total grade for the project, so don’t forget something simple like sending in the code or the video.”

This is all due by Monday Dec 15th at 1:30 in Main 107

1. Video link that demonstrates that code works:

Since we have multiple videos, I have a document for this: Feedback\_Controls\_CSCI255\_Video\_Links

1. The code used to run your project: Feedback\_Controls\_CSCI255\_Code\_Examples

**Inside the document is each group member assignment.**

1. A final paper describing your project.

The document being read is the outline. (duh)

This is the draft: Feedback\_Controls\_CSCI255\_Report.

“The main focus of this paper should be to show someone how to complete what you have done. This also means that you could recreate this project if the paper were handed to you in five years. This being said, you should likely document all components that you used along with the schematic.

The final paper should be written as a document that could be used to recreate the work that you did.

Specifically it should contain wiring diagrams of enough detail as to easily duplicate the work you have

done.

Also you should pick one of the more technically difficult tasks that you overcame and detail the

problem, how you found the solution to that problem and the end result. For example someone had a

problem using the hardware interrupt because they were toggling the output pin connected to the

hardware interrupt in a different portion of their code.”

1. Explanation of the Scope of our project
2. Explanation of I2c
3. Explanation of UART
4. Explain LRF
5. Explain PWM
6. Explain how to set-up and wire components – include
7. Basic Troubleshooting – make a video showing how to do the following?
8. Cp2102: UART bridge to USB – able to debug and program MSP430 and code studio at the same time.
9. Right now we cannot debug code when I2C slave address is changed– oscilloscope was very useful. Sometimes code will not run must be unplugged and replugged.
10. Code composer depends on undetermined setting of computer – works better with CP2102.
11. CP2102, mouse acting crazy. Set-up putty without connecting pins. Once it is running, connect pins.
12. Unplugging and replugging in USB after loading program – we suspect this may be tied to the issue of changing the slave address mid-program, what we are calling dynamically.
13. Restart Code Composer Studio and/or computer – code which works, stops working. Time suck, frustration generator!
14. Jumper between P1.6 and LED – biggest time suck and deterrent of all!
15. Pause and reload issues debugging – what it looks like, unable to read, errors we receive.
16. Rotate jumpers 90° for UART RXD and TXD

Current Problems - Code to be merged with existing I2C slave write code. NOT SURE WE CAN DO ANYTHING WITH THIS BUT I DO THINK WE SHOULD ADDRESS IT. Include any example code we find.

1. PWM on slave –

#1 is clock speed.  Because the UART on the slave side is talking to the LRF at 115200 bps, we have that code running the clock at 16 MHz instead of 1MHz.  This means anything time dependent on the slave side has to be done with a 16 MHz clock.  The only thing time dependent that I know of is reading the PWM.  I did the calculation and a full pulse width on PWM should take 32000 clock cycles at 16 MHz.  That means you must use an unsigned integer to calculate the PWM rate.  It should have a value between 16000 and 32000 that you can send back over the UART.

1. LRF on slave –

The LRF code is in RJ’s computer.  It regularly checks the LRF for a new value using its own UART and stores that value as an integer.  The PWM should store its value as an integer as well, or maybe as two bytes in a character array.  In any case a buffered array should always be ready to dump back to the master if the request should arise.

1. TIME was our biggest issue, not knowing to remove the jumper between P1.6 and LED wasted 2+ weeks. Flakiness combination of I2C and UART, code composer.

Unresolved Problems

1. Troubleshoot: Change # of bytes depending on 1st byte from slave: 1 slave has 5 bytes other slave has 3 bytes to send.
2. Dynamically set/change slave address on Master – This problem seems trivial but it is not!!!!

Bryce said the following, this is not true but mention troubleshootin: I think we discovered why your multiple slave code did not work. The addresses must increase by 2 between each one because the LSb is the read write bit. You may have had conflict because two or more slaves were responding at the same time.

Try this? Also put a short delay between the time that the address has been changed and you request data from the slave.

We suspect it is an initialization problem, or rather some settings must be re-set.

1. Converting String to values after character-wise read in from UART.