Ueberschrift

Wenwen Chen, Christian Dietz,
Jan Seeger, Rainer Schoenberger, Julian Tatsch

August 2, 2013

Abstract

1 Introduction

This is time for all good men to come to the aid of their party!

Outline The remainder of this article is organized as follows. Section 2 gives account of previous work. Our new and exciting results are described in Section 3. Finally, Section 4 gives the conclusions.

2 Previous work

A much longer $\LaTeX\,2_\varepsilon$ example was written by Gil [1].

3 Results

In this section we describe the results.

3.1 Servo controller board

3.1.1 Schematics

3.1.2 Board

After the schematics have been created, the PCB board layout was designed. The components are placed on the board connected with the corresponding copper tracks. This process is also called routing. Routing wasn't a big problem, as we had enough room on the board. Almost all track fitted on one layer and only a few vias and additional wires were needed. This made the PCB manufacturing process easier, as only a one-sided PCB could be used. Figure **??** shows the finished board layout.

3.1.3 Communication protocol

The Servoboard is connected via the I2C or two-wire interface to a command unit. The command unit can either be the FPGA or the RaspberryPi.

The I2C protocol is not implemented in software, but the hardware I2C capabilities of the AVR Atmega8 microcontroller are used. This gives us faster response time and less overhead. It also unburdens the AVR CPU, as only one hardware interrupt is needed, each time a complede byte is received via I2C. Implementing I2C via software would have required many timer interrupts and might have turned out less stable.

We implemented our own communication protocol ontop of I2C. In the microcontroller's firmware, we used a simple state machine to realize this.

The communication protocol looks as follows: The Master first sends a preamble (0xff), which is used to synchronize both communication partners. This is necessary, because loosing packets or receiving corrupted data can lead to a situation, where the receiving unit ends up in an undefined or faulty state. Therefore, it might be that the communication gets stuck.

By sending a preamble, that cannot be part of the normal communication, this problem is solved: everytime the microcontroller receives a preamble-byte, the commonication state machine goes back to its initial state, no matter what the current state is. This way, if a transmission error occurs, only the current command is lost, but further communication will still be possible.

Nevertheless, our communication protocoll makes it possible to transmit 0xff inside a command using the following method: If values greater or equal than 254 are sent, they are split into two bytes. The first byte is 254 and the next one is the difference to the desired number. On the receiver's side, both values are then combined and their sum determines the acual number. For example if 255 needs to be sent, the two bytes 254 and 1 are sent.

4 Conclusions

We worked hard, and achieved very little.

5 Future work

References

[1] J. Y. Gil. LaTeX 2$_\varepsilon$ for graduate students. manuscript, Haifa, Israel, 2002.