Microcomputer Interfacing Chapter 1 - Notes

Mark Lipski

January 18, 2017

1 Introduction

It's rather difficult to explain the motivations behind microcomputer interfacing without first understanding what a micro controller is. Most people are relatively well acquainted with microprocessors, they're what power phones, or any personal computer. Micro-controllers on the other hand function as low power digital control systems and are found in a significant portion of your household electronics.

There are a number of differences between micro-controllers and microprocessors.

1.1 Microprocessors

Pros

- High Data Throughput
- Large quantities of memory
- Operating system

Cons

- High power usage
- High latency
- Restricted to whatever peripherals can be interfaced via usb through the motherboard
- Expensive

1.2 Micro-controllers

Pros

- Substantial number of different micro-controllers available, can usually find a micro that meets your system requirements

- Can be extremely low power
- Can interface a substantial number of peripherals

Cons

- Low data processing capacity

1.3 Hardware Differences

A microprocessor almost always has RAM, as well as a relatively long pipeline. Micro-controllers(MCUs) are typically single core processors with a relatively short pipeline between 1-7 stages, micro-controllers don't usually have RAM, and instead rely upon flash memory to store both data and program code. Most MCUs also have a substantially lower clock rate of between (10-500)MHz.

It should be noted that the classification of a processor isn't black and white, many of the characteristics of MCUs and microprocessors can be interchanged, for instance, there can be an MCU with a substantially higher clock rate of around 300MHz which is substantially better at processing data, however it still may not have RAM or an operating system, and would likely still be classified as an MCU.

These are all the superficial differences between MCUs and microprocessors, and essentially all you need to understand for micro computer interface.

2 Course Motivations

Now that you have an idea of what an MCU is, hopefully you can now derive some reasons for learning the information in this course.

MCUs are often used in control systems. They can be very digital in nature, or even replace analog control systems due to their diversity and how inexpensive they are. They can be used in DSP(Digital Signal Processing) applications, control a car, run your microwave, or any given number of applications.

Some more interesting examples of MCU use would be something like a fitbit. A fitbit is essentially an accelerometer hooked up to an MCU which then transmits the accelerometer data via bluetooth to your phone.//

Important topics in Microcomputer interfacing are:

1. How to write low level C code and assembly to code on an MCU
2. How to choose an appropriate micro for a given application
3. Understand and be able to utilize the standard peripherals on most MCUs
4. Get some practice reading technical documentation

5. How to physically wire and interface an MCU
6. How to properly debug an MCU using both software and hardware approaches.

Point number 3) is what will take up the bulk mass of the material in these notes, as there is are a substantial number of peripherals for MCUs.

Unfortunately point number 4) isn't something that can be taught and must be learned over time spent working with technical documentation.