Communicating Microcontroller Communications to Students

Brian Krug

Department of Electrical Engineering
Grand Valley State University
Grand Rapids, Michigan

Nabeeh Kandalaft

Department of Computer Engineering
Grand Valley State University
Grand Rapids, Michigan

Abstract --- Learning to create simple microcontroller based circuits can be interesting when dealing with blinking lights or moving parts. When dealing with controller to controller communications, the topic becomes much more abstract, especially to the majors not specialized in programming or electronics. This paper addresses two methods used to help maintain interest when teaching microcontroller communications. The first method focused on using the lecture classroom as a pseudo microntroller system, where the students themselves are the transmitters/receivers. The second method modified a lab normally dedicated for driving a motor, but instead used controller communications to drive the same motor. The second approach produces a greater satisfaction as students are not merely receiving confirmation by reading a register, but instead, an action confirms completed communication.

1. Introduction

Microcontroller communications is considered an advanced topic for undergraduate engineering and can be reserved for an embedded system design course. Recent advances have made easy to test and debug software [1]. One particular platform from Texas instruments is the 32 bit ARM microcontroller [2]. Not only is the ARM core popular in industry, but these controllers include all the basics including general purpose input/output (GPIO), timers, interrupts and analog to digital conversion. As systems become smaller, more interfaces to external devices are relying on serial communications. These require less hardware pins and are standardized throughout the industry. Two very common communication protocols considered in this paper are Inter-Integrated Circuit (I2C) and Serial Peripheral Interface (SPI).

These methods were conducted on two 300 level, 2nd semester embedded system design courses. Each class contained about 25 students consisting of mostly electrical and computer engineering majors.

2. Present Teaching Method

All microcontroller functions are typically first introduced in lecture, followed up with some examples, and finally a laboratory exercise to reinforce each topic. This paper will use this framework as a reference. There are many physical forms of microntroller communications from parallel to serial and from wired to radio. Each form can operate using a variety of protocols. Even one of the earliest forms of digital communication, Morse code, relied on a protocol, namely each character is comprised of a long dash or a short dot.

Although a microntroller class generally doesn't deal with Morse code, lectures can start with the widely used communication formats used in most electronic systems. This includes the Universal Asynchronous Receiver Transmitter or UART. The UART is interfaced using an RS232 pinout is shown in figure 1. Although this format will not be the focus of this paper, the improvised methods described later can be easily adopted for this format.

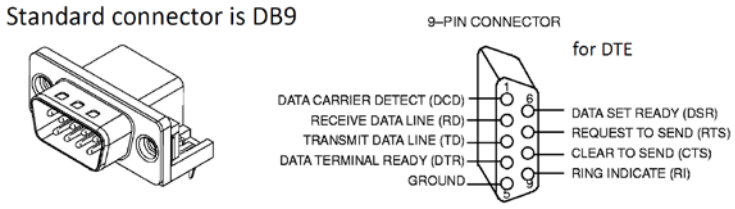

Figure 1: RS232 pinout

The next generation spawned several communication formats including I2C and SPI and will be used to compare teaching methods in this paper. These are much simpler to connect as they require fewer wires than RS232 but are much more complicated to program.

First, the concepts of I2C and SPI are introduced in lecture. This can be done with a power point presentation using a simple wiring diagram and a timing chart like the one shown in figure 2 for an SPI port.

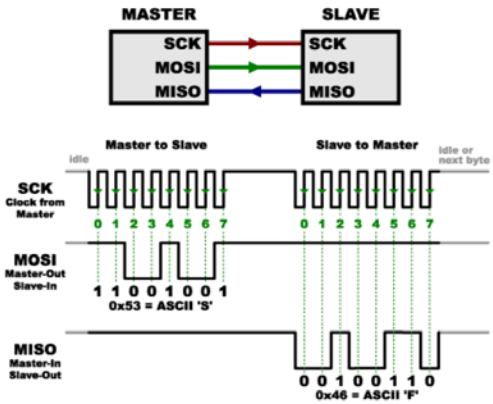

Figure 2: SPI timing

Unfortunately, there are no blinky lights, no whistles, no turning wheels or levers to move. This becomes a digital timing exercise. Going through the exercise only communicates a single letter to the receiver. I2C communication is not any better- and even more difficult to describe given its more complicated protocol [3](but simpler interface), shown in figure 3

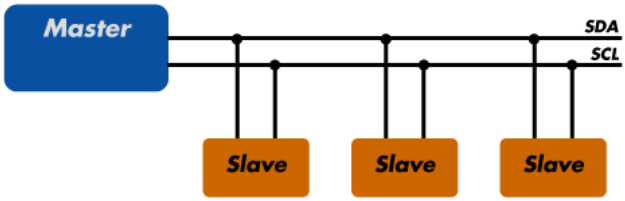

Figure 3: I2C architecture

So after walking through the interfaces in lecture, the student is presented with a lab exercise to attempt to communicate from a microcontroller through one or both of these interfaces.

Because some students may prefer to work alone, single ended communications is preferred, where only one microcontroller is needed to talk to a peripheral such as an LCD screen, real time clock (RTC) or a sensor. All of the mentioned peripherals can be purchased with many different interfaces, from standard GPIO to I2C or SPI to name a few. Once the student had achieved correct transfer of data, then the lesson was over. Because of the relative complexity, retention was low and interest appeared to be even lower.

3. Improvised Method

Two improvements are detailed here. The first targets an improvement in lecture and the second, an improvement in the lab exercise.

First, instead of stepping through the digital exercise for figures 1 and 2, let the class become the figures. This takes a little of prep work, but once the initial setup is created, it can easily be reused. The I2C bus will be used as an example. The main points about an I2C bus that should be understood are the following:

a. Any peripheral can communicate on the same line
b. No peripheral can damage another peripheral by "driving a driver". This is a problem with many communication formats
c. Deciding how the data is transmitted is just a matter of following a timing diagram.
d. Only two lines are needed (and ground) to achieve bi-directional communication.

The class becomes the I2C network. A long spool of two wires (ground and signal) is unwound through the classroom. A series of switches (maybe 10 or so- not every student will get one) is connected between the two wires, about every 10 feet or so. Only the "lucky" students will end up with a switch, which is part of the fun of this demonstration. The instructor's end of the two

wires will have a power supply, a pullup resistor and ground connection. The instructor can also have a switch connected similar to all of the scattered switches. A diagram is shown in figure 4

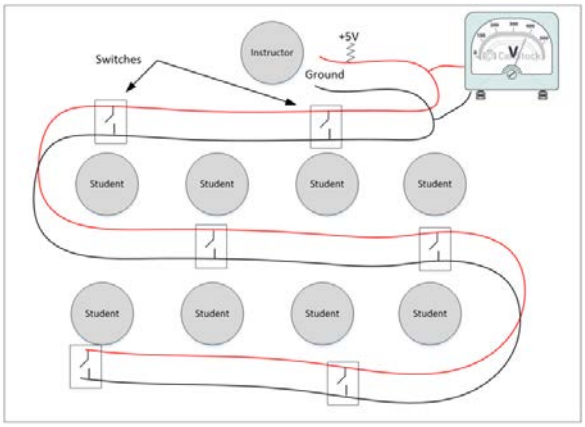

Figure 4: I2C classroom routing

The voltage on the non-grounded wire should be displayed for all to see (shown as a voltage meter on the diagram). If any student closes their switch, the meter will indicate zero volts. Each student is allowed in turn to control the meter. If any student leaves their switch closed, then no other student can affect the meter. Then the timing diagram can be analyzed, as specific students are addressed and asked to respond to the master (instructor).

Students not familiar with pullups/pulldowns really get a feel for why this method works and why this communication protocol is made like it is. The SPI bus can also be demonstrated this way, although one more wire is required.

The pitfalls of this demonstration is that it takes some time to setup. Routing wires through the classroom takes time, along with keeping students from losing focus. After the demonstration, there is cleanup. Overall, the whole exercise takes about 15 to 20 minutes of classtime.

Next, the lab was modified to encourage more creative thinking about how communications work. The normal communications lab was not changed, so students simply downloaded data from a real time clock interfaced via I2C. This is a fairly straightforward lab where a sequence of commands sent at correct timing intervals allows the user to read/write to get the current time and date, or set the time and date.

The next lab was normally just a stepper motor control exercise. There are a lot of features to stepper motors but the basic control is not much more complicated than interfacing simple GPIO. So this was changed to controlling the motor from a second controller. This is illustrated in figure 5

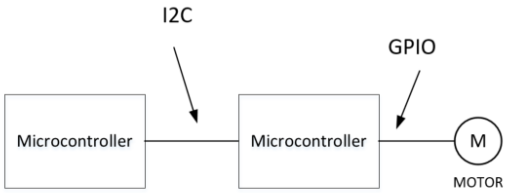

Figure 5

There are many examples available on I2C communications. The MSP432 is a popular microcontroller and the "Programmable Microntrollers" text provides detailed code to send characters back and forth between microcontrollers [4]. The students then need only to translate this code to signal the second microcontroller to control the stepper motor. The latter exercise was similar to the old lab. The following is an excerpt from the new lab exercise:

> **Part III: Controlling the direction and position of the motor remotely**
> EGR 326 Modify your C program from part one so that one MSP controls the motor, while the other is connected to your keypad and two pushbuttons. Each MSP is connected via an I2C communications port.
>
> •Pressing any digit (1-9) will store and display the digit pressed to the console
>
> •Pressing the first button will increment the motor the number of cycles (cycle= 4 steps) entered from the keypad.
>
> •Pressing the second button will decrement the motor the number of cycles (cycle=4 steps) entered from the keypad
>
> •You should also print to the console the present motor count (representing position). Demonstrate the working prototype to your instructor

4. Feedback results

At the conclusion of the semester, after these two method were incorporated into two classes, a blind survey was conducted to get the students impressions of the changes to the lecture and additional lab work. Fifty one students in all responded to the survey.

The survey broke down the changes as follows:

The students were asked to rate on a scale of 1 to 5 (5 = very useful, 1= not useful) the following methods to teach microcontroller communications:

1) Powerpoint presentation (very similar to previous semesters)
2) Class demonstration (as shown in figure 4)
3) Updated Lab (controller-controller-motor exercise)
4) Controller- RTC lab (very similar to previous semesters)

The results are shown below:

1) Power point: 3.67
2) Classroom demonstration: 3.78
3) Updated Lab 4.20
4) Old Lab 3.61

The results show that the old method of lecture and simple "one way communication lab" were somewhat effective at explaining microcontroller communications. But both the classroom demo and updated lab really helped reinforce what is viewed as a less than exciting topic. This first attempt at a classroom demonstration only included a couple switches. It might be more effective if at least half of the class were participants and thus improve its effectiveness. There were several anonymous comments on how the classroom exercise helped understanding and comprehension for the final exam.

5. Conclusion

Microntroller design is an exciting and growing area in research and industry. But there are interesting aspects to design and then there are the less than interesting areas. This paper targets microcontroller communications as one of those less than interesting areas that can be made interesting with two techniques. First the lecture was spruced up by incorporating the class as an actual communication network. This helps support the details of the protocols as many students don't realize what they don't know until they see a network in action. Second, the lab was updated to incorporate communications into other labs that are more "action" oriented. This paper presented incorporating communications with driving a stepper motor. Feedback indicated that both methods increased understanding over traditional teaching methods while not adding significant lecture or lab time.

References

1. Steve Hsiung, Feng Jao, Lijian Xu, Marjaneh Issapour, Collaborated Efforts in TI ARM M4/32Bits Microcontroller Curricula Developments and Assessments, 2018 ASEE conference, June 24.

2. Antonio Francisco Mondragon, The Embedded Development Tools You Did Not Have When Growing Up

3. NXP UM10204 I^2C Bus Specification and user manual, Rev 6, April, 2014

4. Cem Unsalan, H. Deniz Gurhan, M. Erkin Yucel, Programmable Microcontrollers: Applications on the MSP432 LaunchPad, McGraw Hill, 2017