2005

Implementing Simple Protocols In Multiple Processors Control Applications

Steve Hsiung
Old Dominion University, shsiung@odu.edu

Tyson McCall
Old Dominion University

Corinne Ransberger
Old Dominion University

Original Publication Citation

Hsiung, S., McCall, T., & Ransberger, C. (2005). *Implementing simple protocols in multiple processors control applications.* Paper presented at the 2005 ASEE Annual Conference and Exposition, Portland, Oregon.

Implementing Simple Protocols in Multiple Processors Control Applications

Steve Hsiung,
Tyson McCall,
Corinne Ransberger
Engineering Technology Department
Old Dominion University
Norfolk, VA 23529

Abstract

Using microprocessor/microcontroller in various control applications is not only one of the major topics in Engineering Technology curricula, but also of interest in industry applications. To accomplish it correctly the process of designing application programs starts from the individual module development through extensive testing, verification, and modification. Applying these developed modules in a useful manner requires the links and integrations that lead to the practical project implementation. Frequently, in students' senior project designs and faculty's research plans, the microprocessor/microcontroller resources become scarce or cause conflicts during the modules' integration stage.

To accommodate the shortfall of the resources and resolve any conflict state, several choices must be considered, such as the need to revise or totally rework the module, or apply the module with additional circuit design. This article presents a proven concept that implements the simple serial communication protocols in a multi-processor environment, which aims to keep the pre-developed modules intact with the least possible modification when they are integrated into the project.

I. Introduction

A project called Sparring Partner was implemented under a contract between a private company and Old Dominion University, Technology Applications Center in 2004. This project was to design and develop a training robot that is to be used in the boxing training exercises. Its original design relied on a single CPU (Motorola 68HC11) to control 8 DC motors, 8 position sensors, and some other peripheral and safety features.

After the prototyping and examination of the mechanical functions, it was determined that the control circuits had to be revised. The requirements for this 68HC11 had grown to 9 DC motors and 18 position sensors with the same safety features. Due to the limitation of the 8 bit 68HC11 CPU, the processor's resources were exhausted [6,7]. The mechanical designers desired to

have a total of 20 or more position sensors to gain adequate controls of the training model, but this list was forced to cut down to accommodate the CPU.

The 68HC11 is a microcontroller that has been in the market since 1980 [2] and Motorola has discontinued the manufacture of this product. It became difficult to find the supplier for this chip and its price is higher than expected. After a study of the specifications and potential applications on SPI [11], I^2C [8], and SMBus [12,13], an idea has surfaced to revise the designs on the electronic hardware and software to use multiple and cheaper CPUs, such as Microchip's 16F84A in a form of serial communication links [8,11,13]. The 16F84A is a popular 8 bit microcontroller in many places and the vender suppliers are plentiful [1].

The design initiative is to have multiple slave processors that everyone is in the same format, which uses one 16F84A as a dedicated CPU to control one motor. A single master 16F84A CPU controls and links all the slave CPUs. This design becomes a multiple processor environment that has a single master which takes the control commands from a user and passes the necessary control functions to an appropriate slave to perform the operations. With this design concept, there will be virtually no limit on the number of slaves in the system. The previous limitations on a single CPU design are automatically resolved.

Certainly, this approach requires a well planned software protocol design, and the hardware requirement becomes a fixed module that is less complex [8,13]. This paper focuses on hardware, software designs, and their implementation as a proof of concept of a multiple processor control application. The use of multiple PIC 16F84As in a system design is doable, low cost, and increases the system efficiency.

II. Software Design

Since there are multiple slaves and a single master in this control system design, two kinds of software are needed for this project. The major proof of concept in this project heavily relies on the software design. In order to better clarify the design of this proof of concept project, subsequent sections of this paper will describe the master, slave protocol, and communication.

1. Software on Master

The master is the controlling microcontroller which handles all the controlling sequences, such as taking interaction between a user and the system, making sure the right motors are running in the specified time and position. The master controller oversees major system components such as the keypad, LCD, and slave microcontrollers that run the motors.

In operation, the master starts with the keypad and LCD display module, handling interactions between the user's inputs and system's response. The keypad routine is a standard scanning, debouncing, and decoding of the four rows and four columns to detect the user's input. The LCD routine implements serial communication between the master CPU and display module via a 74164 shift register [14].

A major portion of the software design in this project is the communication between the master and the multiple slaves. All the communications are initiated by the single master. Once the master has processed an action selected by the user input, it determines which action was chosen and transmits the information/instructions to the appropriate slave using serial synchronous communication [11]. Both read and write on the master side are implemented in the

same subroutine. This routine is in charge of generating the clocks, and sending and receiving the bits of information. Since 16F84A does not have any hardware support for serial communication, the clock and data bits rely entirely on software bit banging. The time between the clock edges is preset at .2ms for the whole system. In order to control the multiple slaves, every slave has a unique address that is embedded in the master software. There are pre-defined four bytes commands that a master sends to all the slaves in the system. Two dedicated I/O pins on the master and slaves synchronize the intended action between the parties.

Every communication sequence consists of the master broadcasts five bytes (four command bytes and one 0XFF byte to read from the slave) on the shared bus lines [11,13] consisting of a clock (CLK), data out (DOUT), data in (DIN), and framing I/O bits. The first byte is the slave address, the second byte is a master read, the third byte is the speed of the motor, the forth byte is the motor direction, and the last byte is the motor running time period. Once the master receives the acknowledgement (ACK) from the intended slave, it sends the remaining three bytes. When that is done, it goes on to the address of the next action line of bytes that needs to be sent and continues on until the control sequences are finished. When all instructions are sent to the slaves, the master will return to start and wait for the next control sequence from the user through the keypad.

2. Software on Slave

The slave is in charge of executing what the master has sent. It does not start processing information until the master is ready to send. However, the slave has a few tasks it must complete before it is ready to start the communication.

To make the individual motor controller as a fixed modular design, it should have the same hardware and software but perform different action. A unique address has to assign to each slave to differentiate them. This becomes the only difference between the slaves in this system. To start, the slave first pulls its address from the EEPROM and stores it into its DRAM [9]. Once this is accomplished, it waits for the master to signal the start condition with a framing of "00" as an initiate of the sending information. The first byte is going to be the address byte. The address byte that is received is compared to the address of the particular microcontroller. If they are different, the microcontroller does nothing but waits/polls for the next action address. Once it receives the correct address, the slave waits/polls for another framing condition "01" and sends the ACK.

After the ACK, the slave waits/polls for a different framing condition of "10" to receive three more motor control bytes information. All the bytes will be stored in the predefined DRAM locations [1,9]. When the slave is finished receiving the rest of the instructions, it activates the motor accordingly.

The motor speed byte is used to determine the prescalor to the 16F84A TMR0 timer interrupt interval that is used to generate the PWM signal to regulate the motor speed [1,9]. The motor running period byte is used in conjunction of the sensors to determine when to shut down the gate of the PWM signal that eventually stop the motor. When that is accomplished, the slave is ready for the next set of information from the master.

3. The Protocol

There are basically three I/O lines (clock, data in, and data out) used in this

communication. These are all shared as a serial bus between a master and multiple slaves [8,13]. In each action of the serial communication bits streams, there are total of five bytes either transmit or receive between a master and any particular slave CPU. The pre-defined bytes are: (1) address byte, (2) slave acknowledge byte, (3) speed byte, (4) direction byte, and (5) time period byte.

There are several set of rules for this communication [8,11,13]: (1) only one master is allowed in the system, (2) only the master can generate the clock, (3) only the master can start/stop the communications, (4) only the master is responsible for the framing I/O bits, (5) there are multiple slaves allowed in the system, but each slave shall have a unique address recognizable by the master, (6) the slave can only monitor the framing I/O bit for communication responses, (7) the slave is required to respond to its address call by sending an ACK byte, (8) after the initiation of a start from the master, every slave has to read the address that master broadcasts, (9) the slave is not permitted to respond if its address is not called, and (10) the only time that the slave sends a byte is when it is required to ACK.

To ensure the safety of the system performance, an alarm condition is implemented in the event of violation of the protocol. The alarm condition is defined as: when the master sends a legitimate address to the slaves and does not receive an ACK or the ACK is not recognized for any reason. As soon as the master detects this alarm condition, it will disable the de-multiplexer that is used by the slaves to activate the motors.

4. The Communication

At the beginning of the protocols testing stage, it was difficult to keep the master and slave synchronized with each other. Therefore, the protocol rules mentioned above were introduced and the framing I/Os were added. These two additional I/O pins were developed to frame the states of the communication bytes to fix the problem. They are the states that the master controls and slaves poll during each action. The framing I/Os are inputs to the slave. They are used to indicate to the slaves that the master is ready to do the next set of instruction. There are four different states (00, 01, 10, 11) the master sends to the slaves. Since the slaves don't perform as much work as the master, these states let the slaves recognize where the master is in the communication sequence. "00" informs the slaves as a start that the master is getting ready to send the first byte (address). Following the first byte, the master sends a "01" to notify the slaves it is ready to receive the ACK. When the I/O pins switch to "10", the slave knows that the master has received the ACK and is about to send the last three bytes. Finally, the master will send an "11" through the I/O pins, indicating it is done with the transmitting action.

The synchronous serial communication shares the same clock [8,11,13], and every party relies on the clock edges to either read or write the bits. The framing implementation resolves the timing issues and differentiation of the start, end/stop, address, and command bytes. The presentation of the protocol bytes and associate framing I/O is presented in Figure 1.

III. Hardware Design

Although the hardware design appears complex, it has a lot of duplications due to multiple CPUs in the system. There are three major parts in this design: (1) the master that handles the user's commands and communications via keypad and LCD module, (2) the multiple slaves that actual

generate the PWM signals to drive the motors via DC motor interface board, and (3) the motor driver board circuit that handles the high current to activate the DC motors.

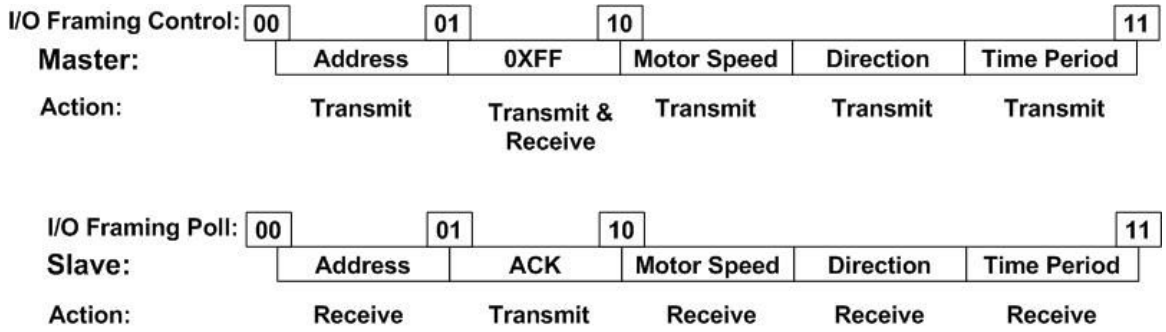


Figure 1. Master & Slave Protocols

1. Master Control Circuit

The master interface circuit has a standard 4x4 keypad, a LCD module, a shift register, and three software controlled I/Os for the serial interface buses to the slave CPUs.

The keypad is direct interfaced to PORTB RB0-RB8 with eight 10K pull up resistors [1,9]. RB0 (IO_1) and RB1 (IO_2) are dually used for the slaves' serial communication framing I/O controls. These two logic lines will generate four different states that are used as guidance to the slaves to follow the predefined protocols. RB2 (MA_E) is also used as a control of the slave's de-multiplexer enable that serves as an alarm condition for master to shut down all the motors when an emergency condition is encountered.

The LCD module is connected to a 74164 shift register in a parallel format, but its interface to the CPU is in a serial form. This is needed because of the limited number of available I/Os on the master CPU. RA2 (LCD_E) and RA3 (LCD_RS) are used for E and RS controls on the LCD display module [4].

The entire serial interface is done through PORTA. RA0 (marked as CLK) is used to generate the clock, RA1 (marked as DOUT) is the control data output from a master to the slaves, and RA4 (marked as DIN) is a return data line from the slaves to a master. RA4 is an open drain I/O, making it the best choice for this type of interface communication [1,9]. There are three serial communication lines that are not only used in master-slave, but also in CPU-LCD interactions. There is no unused pin on the master circuit. Figure 2 presents the master hardware circuit design.

2. Slave Control Circuit

The serial communication interface is implemented on PORTA where RA0 (CLK) is used to accept the clock signal from the master, RA1 (DOUT) is used to read the command bytes from the master, and RA4 (DIN) is used to send the ACK to the master. The PWM signal is generated from the TMR0 timer via interrupt control on RB0 pin [9]. It is used to control the de-multiplexer output that eventually is used to regulate the energy to the DC motor. The PWM signal is generated constantly since it is an interrupt driven event. To gate this PWM to a proper channel (either forward or reverse control of the motor), RB3 (marked as SLX_E) is used as an enable control. Both RB1 (marked as SLX_RB1) and RB2 (marked as SLX_RB2) are used as

channel select on the 74138 3-to-8 decoder that is functioned as a de-multiplexer. The X on SLX stands for the number of the motors in the circuit.

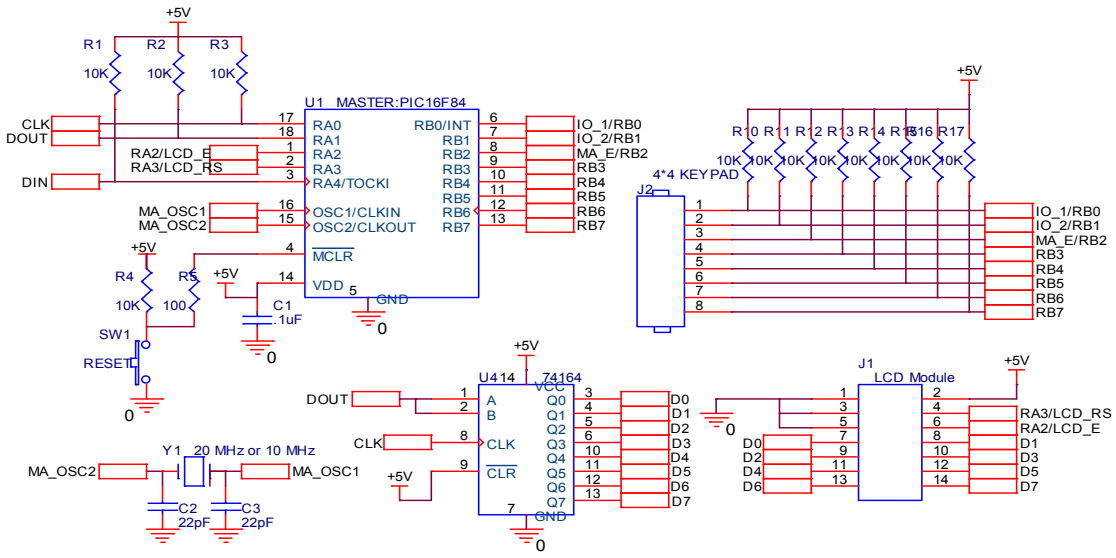


Figure 2. The Master Hardware Circuit

There are two position sensors on each slave. The signals are monitored on RB4 (SLX_SENSOR1) and RB5 (SLX_SENSOR2) pins to provide feedbacks on the motor's position. The framing logic states are monitored on RB6 (IO_2) and RB7 (IO_1), which controls the slave's communication protocols sequences. There are two unused I/O pins, RA2 and RA3 on the slave circuit design. The multiple slaves are a duplication of the following slaves' circuits as presented in Figure 3.

3. Motor Control Circuit

The motor driver circuit is a standard H-bridge design. The bridge on-off controls are made through an IRF530N power MOSFET that can easily handle 10A DC current [5]. The circuit can control a motor in either forward or reverse direction depending on the PWM signal that is coming in at its P_F_1 or P_R_1 terminal. The two position sensors have RS latch debounce circuit to produce a clean signal as a feedback to the slave CPU. The motor circuit design is presented in Figure 4.

IV. The Implementation

This multi-processor control application that uses simple protocols has been proven functional. The testing of this concept was carried out in one master and four slaves to control four different DC motors. The setup is presented in Picture 1.

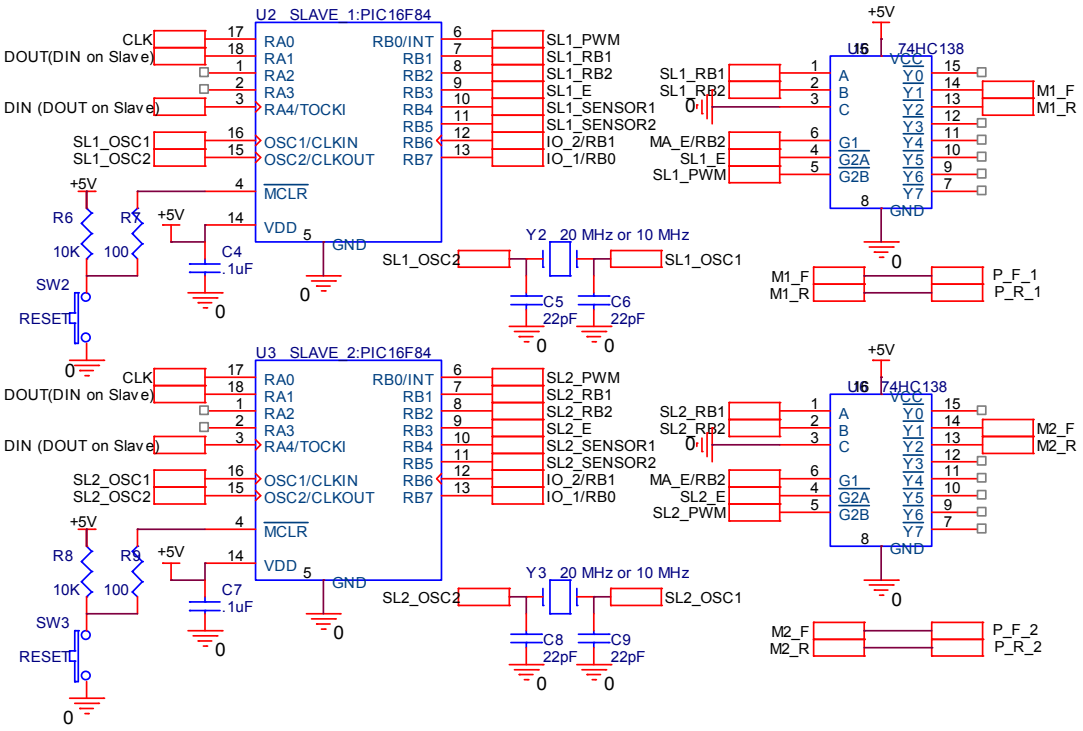


Figure 3. Two Slaves Hardware Circuit

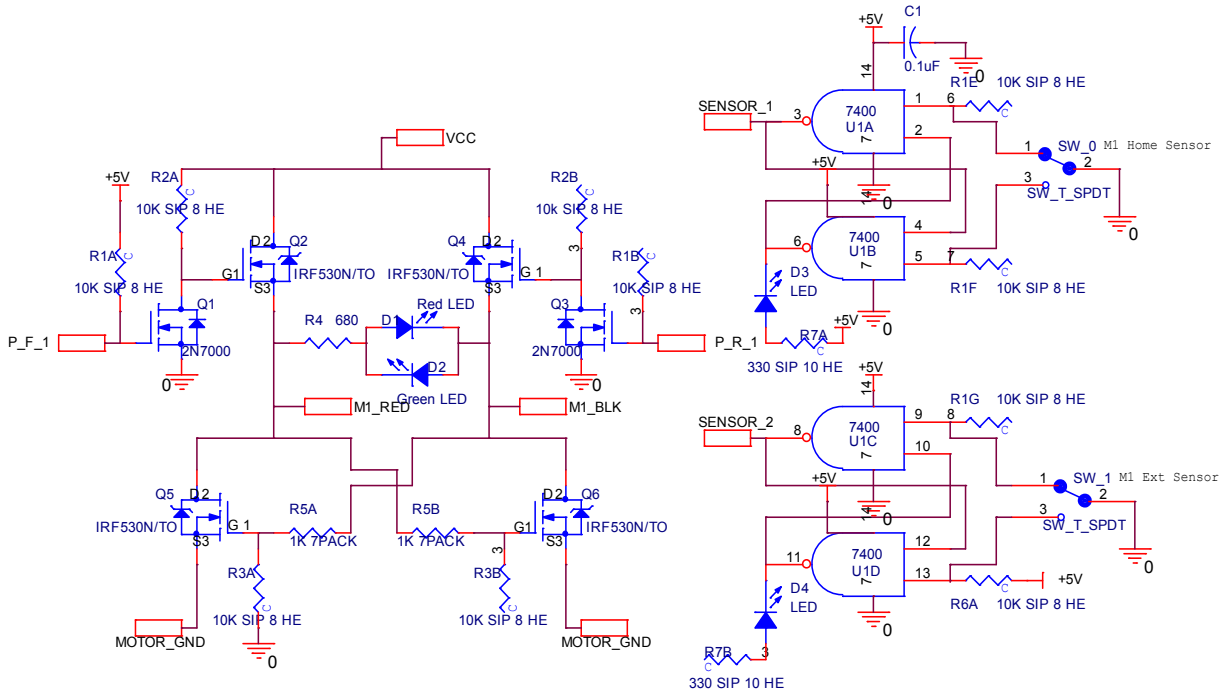
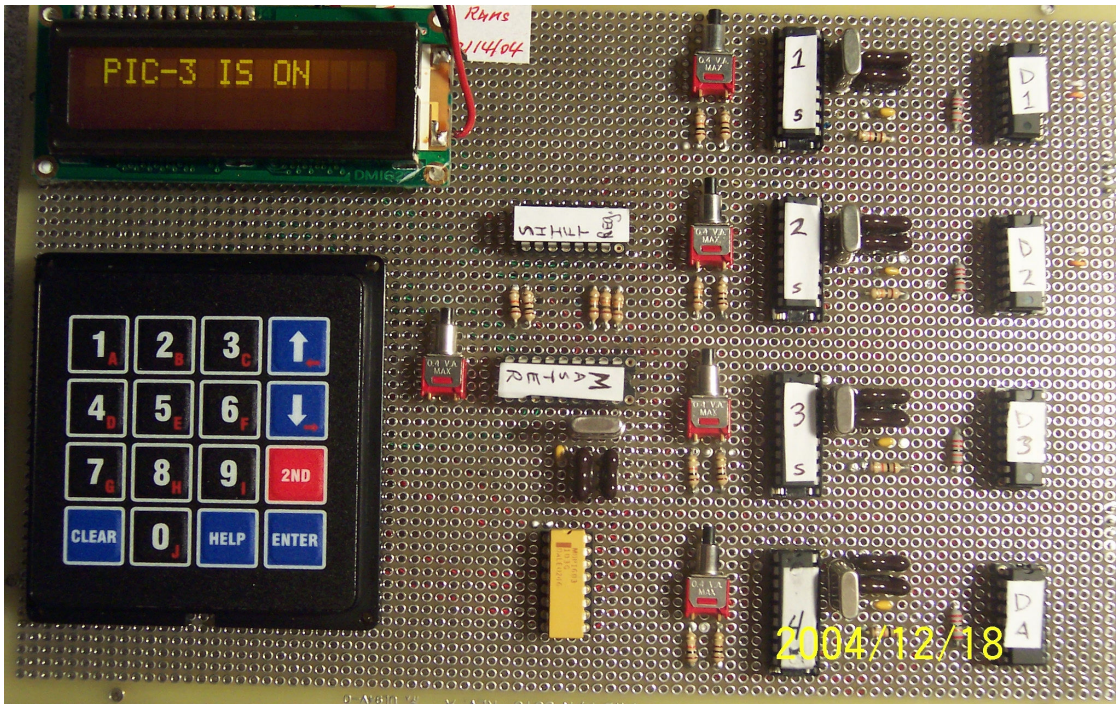


Figure 4. The Motor Control Hardware



Picture 1. One Master & Four Slaves Setup

The intended addresses for the 9 slaves are:

PIC 1 = 0X11 (Right Elbow)
PIC 2 = 0X22 (Right Shoulder up/down)
PIC 3 = 0X33 (Left Elbow)
PIC 4 = 0X44 (Left Shoulder up/down)
PIC 5 = 0X55 (Pivot Torso)
PIC 6 = 0X66 (Back/Forth Torso)
PIC 7 = 0X77 (Right/Left Torso)
PIC 8 = 0X88 (Right Shoulder Left/Right)
PIC 9 = 0X99 (Left Shoulder Left/Right)

The duty cycles (DC) for the TMR0 timer prescalor are:

Slow Speed = 0XD5 = 20% DC
Medium Speed = 0XD6 = 50% DC
Fast Speed = 0XD4 = 80% DC

The direction is defined as:

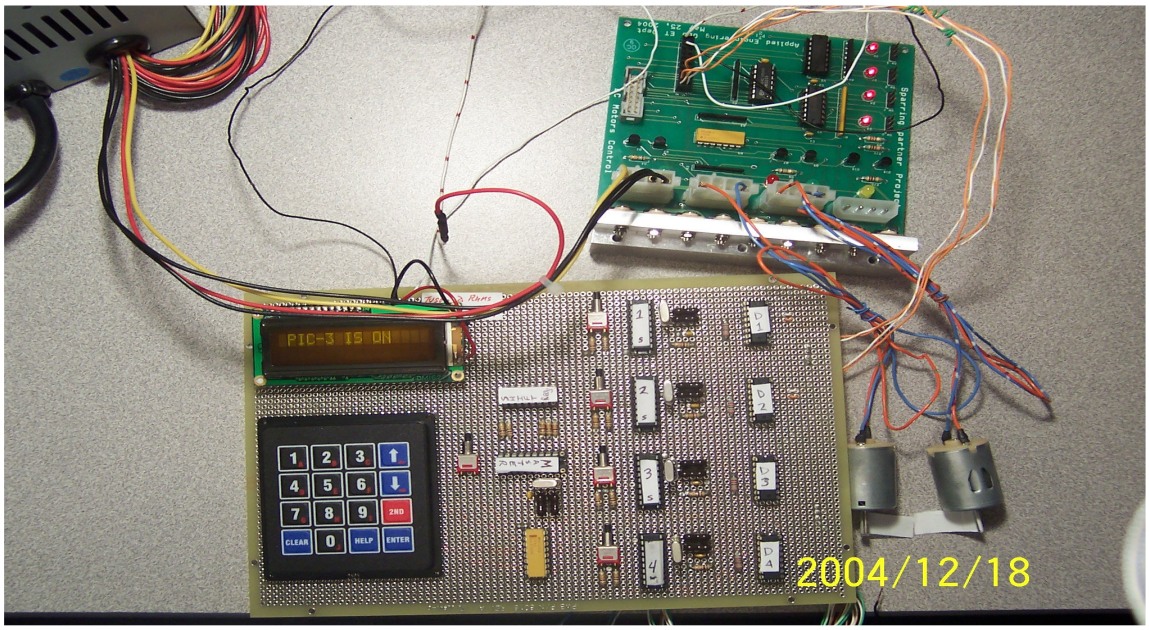
Motor Backward = 0X01
Motor Forward = 0X02

The time period is defined as:

Short = 0X20 = 2.04 Seconds
Medium = 0X33 = 4.02 Seconds
Long = 0X65 = 78.965 Seconds

The sample communication bits streams on clock (CLK), data out (DOUT), and data in (DIN) lines is presented in Picture 2. The prototype demo system that has one master and four slaves with two DC motors is presented in Picture 3. The overall operation of this demo system is presented in a block diagram format in Figure 5.

Picture 2. The Serial Communication Signals



Picture 3. Multiple-Processor Demo System

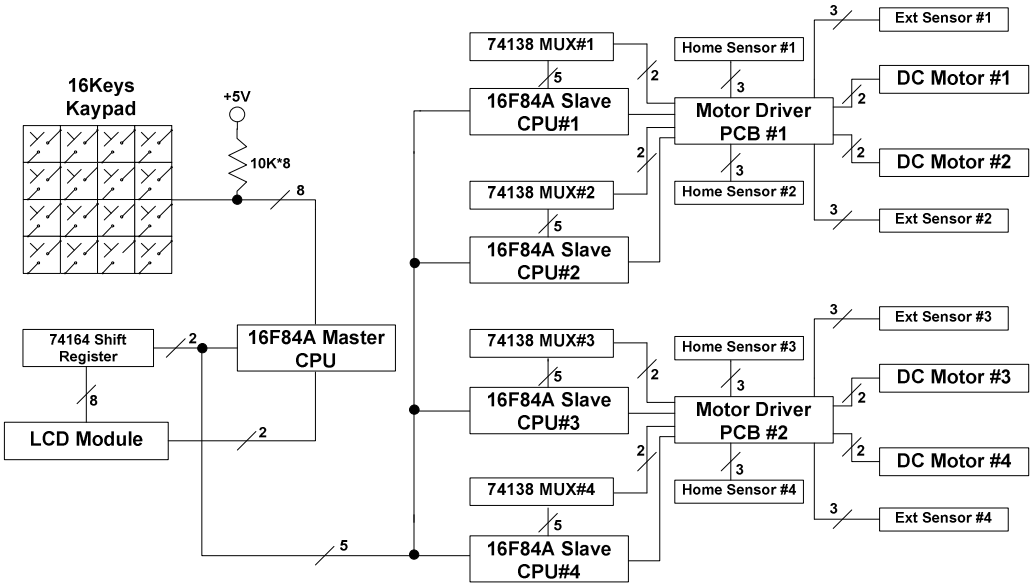


Figure 5. The Demo System Block Diagram

Table 1 represents the estimated cost of the multi-processors system that uses PIC16F84As is compared with the original design using a single MC68HC11 processor. The cost for both systems is very similar, but the flexibility that multi-processor system provides is beyond this assessment.

Single CPU System	Cost	Multi-CPUs System	Cost
MC68HC11 & EVB	$80	PIC16F84A*10	$40
Motor Driver PCB & Accessories	$100	Motor Driver PCB & Accessories	$100
Misc Components	$30	Misc Components	$50
Total	$210	Total	$190

Table 1. The Estimated Cost Comparison between Two Systems

V. Conclusion

This project was actually implemented in the students' (Tyson McCall and Corinne Ransberger) senior project design. The concept has been successfully proven to be suitable in real multiple motors control applications. There are several valuable lessons that were learned in this proof of concept project design.

The designed serial communication protocols with only byte address can have up to 254 slave processors (that exclude 0X00 and 0XFF). This can easily be extended to any number of slaves by adding multiple bytes of the address definitions. The three control bytes in the existing protocols can also be extended to fit any project needs. The two bits I/O protocol framing controls from the master may be eliminated by using the clock edge sensing interrupt to synchronize the communications. Certainly, an upgraded CPU such as the 16F877A that has a built in SPI hardware block would relieve the software bit banging load on the CPUs to improve communication efficiency, but would also increase the cost of the project design [10]. An implementation of the SLEEP in all the CPUs software will make the system more energy

efficient [1]. If security is a concern, adding the CRC-8 implementation in the protocols will be one of the solutions [3].

This capstone project brought the theory and protocol design [11,13] of the synchronous serial communication in the chip level into a practical application has made a good impact on the students' understanding of the potential in their ET career. Regarding their accomplishment, here is what the students said, "As the project nears to an end we have learned so many things. We learned how to communicate with each other as team members, which we think is important when we move on and start our careers. We also learned how to communicate with multiple microcontrollers, and how to design a project that can be useful to the environment. We were able to take our knowledge and apply it to one project. We were able to see that each one of us thinks completely different, which makes the project unique and different. The boxing robot no matter how simple it may sound, it was a great way for us to apply our knowledge of communication, electronics, and teamwork."

The concept of the serial communication is simple and has been utilized in different areas in the real world. This integration of the existing concept in a custom-made application that brings real-life applications into classes, has served one the important missions of the ET education: applied engineering. This proof of concept provides the student with an interesting application idea and a better understanding of the links between hardware and software along with their potential applications in the workplaces.

VI. Bibliography

1. Bates, Martin, "PIC Microcontrollers: An Introduction to Microelectronics", 2nd, Elsevier: Newnes, 2004.
2. Cady, Frederick, M., "Software and Hardware Engineering, Motorola M68HC11", Oxford University Press, 1997
3. CRC-8 Implementation White Paper, USAR System Inc., www.semtech.com, 1999.
4. "How to control a HD44780 based character LCD", http://home.iae.nl/users/pouweha/lcd/lcd0.shtml, 2004.
5. IRF530N, Internacional Rectifier, www.irf.com, 2004.
6. Motorola , "M68HC11 Reference Manual", Motorola, Rev 3, 1991
7. Motorola , "M68HC11 E Series Programming Reference Guide", 1991
8. Philips Semiconductors I^2C Specification, www-us2.semiconductors.philips.com /i2c/news/, 1997.
9. PIC16F84A Data Sheet, Microchip Technology Inc., http://www.microchip.com/, 2004.
10. PIC16F877A Data Sheet, Microchip Technology Inc., http://www.microchip.com/, 2004,
11. Serial and UART tutorial, Frank Durda, www.freebsd.org/doc/en_US. iso88591-1/articles/serial_uart/, Email:uhelm@freebsd.org, 1996.
12. System Management Bus (SMBus) Specification, Revision 1.1, Smart Battery System Specifications, www.sbs-forum.org, Email: battery@sbs-forum.org, 1998.
13. System Management Bus (SMBus) Specification, Revision 2.0, Smart Battery System Specifications, www.sbs-forum.org , Email: battery@sbsforum.org, 2001.
14. TTL Logic Data Book, Texas Instruments, 1988.

VII. Biography

STEVE C. HSIUNG

Steve Hsiung is an associate professor of electrical engineering technology at Old Dominion University. Prior to his current position, Dr. Hsiung had worked for Maxim Integrated Products, Inc., Seagate Technology, Inc., and Lam Research Corp., all in Silicon Valley, CA. Dr. Hsiung also taught at Utah State University and California University of Pennsylvania. He earned his BS degree from National Kauhsiung Normal University in 1980, MS degrees from University of North Dakota in 1986 and Kansas State University in 1988, and PhD degree from Iowa State University in 1992.

TYSON J. McCALL
Tyson McCall completed his Bachelor of Science in Engineering & Technology with an emphasis on Computer Engineering Technology at Old Dominion University, in December 2004. He is currently a member of the National Society of Black Engineers (NSBE) and is past Telecommunications Chair of the ODU chapter. He is a former member of The Institute of Electrical and Electronics Engineers (IEEE) and has twice been recognized for academic achievement Dean List's at Old Dominion University.

CORRINE A. RANSBERGER
Corinne graduated from Old Dominion University in 2004. She currently has a BS in Electrical Engineering Technology will continue her advanced education in Electrical Engineering. The last year of her college career, Corinne had the opportunity to work as a teacher assistant for a few of the teachers in her department.