Journal of
INDUSTRIAL
TECHNOLOGY

Volume 23, Number 3 - July 2007 through September 2007

The Use of PIC Microcontrollers in Multiple DC Motors Control Applications

By Dr. Steve C. Hsiung

Peer-Refereed Article

KEYWORD SEARCH

Computer Programming
Electricity
Electronics
Research

Dr. 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. Steve can be reached at shsiung@odu.edu.

The Use of PIC Microcontrollers in Multiple DC Motors Control Applications

By Dr. Steve C. Hsiung

Abstract

This article is an applied engineering example of using microprocessor/microcontroller in various controls of the senior project designs in Electricity, Electronics, and Computer Technology (EECT) concentration in Industrial Technology curricula, but also of interest in industry applications. This project reflects the Industrial Technology curricula as a combination of hands-on and minds-on approach on real-world applications. The design of a microprocessor/ microcontroller control system involves 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.

This development demonstrates the accommodation of the shortfall of the resources and resolves any conflict that may force the designers to: (1) revise the module design, (2) rework most of the design, or (3) add additional circuit to the module otherwise. This use of multiple PICs presents a tested research concept that implements simple serial communication protocols in a multiprocessor environment, which aims to keep the pre-developed modules intact with the least possible modification during project implementation.

Introduction

The focus of this project is how to expand the role of existing small scale PIC microcontrollers into a multiple-microcontrollers system to resolve limited resource issues in meeting complex project needs.

A project was implemented under a contract between a private company and Old Dominion University, Technology Applications Center in 2004; this project, designed to develop a teaching robot, was 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 (Motorola, 1991). 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 (Cady, 1997) 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 (Serial UART Tutorial,

1996), I²C (Philips, 1997) and SMBus (SMBus1.1, 1998; SMBus2.0, 2001), an idea 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 (Philips, 1997; Serial UART Tutorial, 1996; SMBus2.0, 2001). The 16F84A is a popular 8 bit microcontroller in many places and the vender suppliers are plentiful (Bates, 2004). From an economic point of view, it is the one of the best choices of all the available low-end microcontrollers.

The selected design is to have multiple slave processors that everyone is in the same format and uses the 16F84A as a dedicated CPU to control one DC motor. A single master 16F84A CPU controls and links all the slave CPUs. This design is to modularize the 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 limitations in the previous design on a single CPU approach 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 (Philips, 1997; SMBus2.0, 2001), and thus the following section on hardware, software designs, and their implementation as a proof of concept of multiple processors in multiple DC motors control applications in an applied research on the use of multiple PIC 16F84As in a system design is convinced, low cost, and efficient.

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. To clarify the design of this concept, subsequent sections of this paper will describe the master, slave protocol, and communication.

Software on Master

The master is the controlling microcontroller which handles all the controlling sequences, such as the interface 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 (TTL, 1998).

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 (Serial UART Tutorial, 1996). 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 0.2 ms for the whole system. To control the multiple slaves, every slave has a unique address that is embedded in the master software. There are predefined five bytes protocols that a master sends to all the slaves in the system. Two dedicated I/O (Input/Output) pins on the master and slaves synchronize the intended action between the parties.

Every communication sequence consists of the master broadcasting five bytes (four information bytes and one 0XFF byte to read from the slave) on the shared bus lines (Serial UART Tutorial, 1996 & SMBus2.0, 2001) 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.

Software on Slave

The slave is in charge of executing what the master has commanded. It does not start processing information until the master is ready to send. However, the slave has a few tasks of its own before it is ready to start the communication. To make the individual motor controller 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 various slaves in the system. To start, the slave first pulls its address from the EEPROM and stores it into its DRAM (PIC16F84A, 2004). Once this is accomplished, it waits for the master to signal the start condition with a framing I/O 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 I/O condition "01" and sends the ACK. After the ACK, the

slave waits/polls for a different framing condition of "10" (framing I/O) to receive three more motor control bytes information. All the bytes will be stored in the predefined DRAM locations (Bates, 2004; PIC16F84A, 2004). 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 (Pulse Width Modulation) signal to regulate the motor speed (Wilmshurst, 2007; PIC16F84A, 2004). The motor running period byte is used in conjunction with the sensors to determine when to shut down the gate of the PWM signal that will eventually stop the motor. When that is accomplished, the slave is ready for the next set of information from the master.

The Protocol Design

There are three basic 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 (Philips, 1997; SMBus2.0, 2001). In each action of the serial communication bit 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 ACK (acknowledge) byte, (3) speed byte, (4) direction byte, and (5) time period byte.

There are several set of rules for this communication (Philips, 1997; Serial UART Tutorial, 1996; SMBus2.0, 2001): (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.

Table 1. Framing I/O Controls

Framing I/O	Mater Control	**Slave Response**
00	Start Transmission of Address	Ready to read the address
01	Start Receiving of ACK	Ready to send the ACK
10	Start Transmission Command Byte of Motor Speed	Ready to read the Motor Speed Control
10	Start Transmission Command Byte of Motor Direction	Read the Motor Direction control
10	Start Transmission Command Byte of Motor Run Time	Read the Motor Run Time Control
11	Signal End of Transmission	Recognize End of Transmission

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 outputs from a master and inputs to the slaves. These signals 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 summation of the framing I/O control is presented in Table 1.

The synchronous serial communication shares the same clock (Philips, 1997; Serial UART Tutorial, 1996; SMBus2.0, 2001), and every party relies on the clock edges to either read or write the bits. The framing I/O 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.

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.

Master Control Circuit

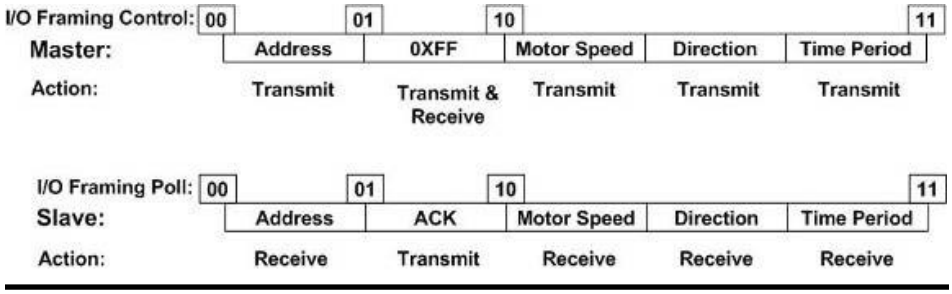
The master interface circuit has a standard 4x4 keypad, a LCD module, a shift register, and three software controlled I/Os for the serial interface buses to the slave CPUs.

The keypad is directly interfaced to the PORTB RB0-RB7 with eight 10K pull up resistors (Bates, 2004; PIC16F84A, 2004). 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 when following 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 (LCD, 2004).

The entire serial interface is accomplished 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. Its required pull up resistor (10K) connection makes it the best choice for this type of interface communication (Bates, 2004; PIC16F84A, 2004). There are three serial communication lines that are not only used in master-slave control, but also in CPU-LCD interactions. All the I/O pins on the master circuit are utilized. Figure 2 presents the master hardware circuit design.

Figure 1. Master & Slave Protocols



Slave Control Circuit

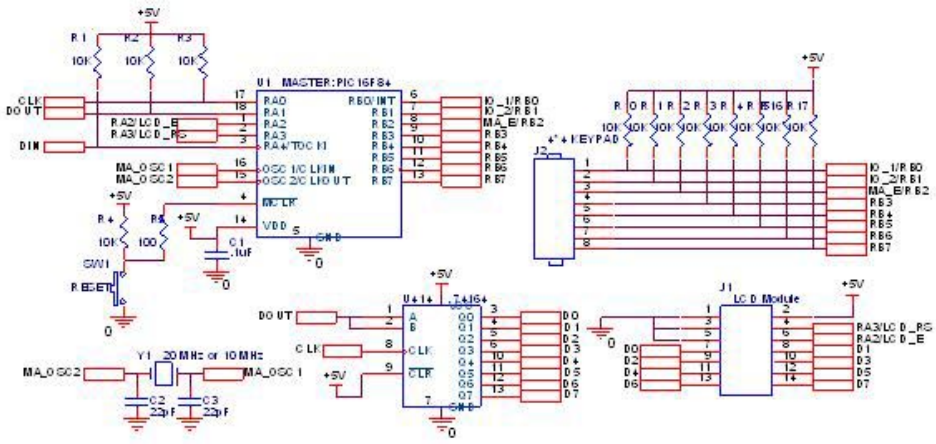
The serial communication interface is implemented on PORTA where RA0 (CLK) is used to accept the clock signal from the master, RA1 (DOUT) is used to read the command bytes from the master, and RA4 (DIN) is used to send the ACK to the master. The PWM signal is generated from the TMR0 timer via interrupt control on RB0 pin (PIC16F84A, 2004). 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) I/O pins 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 motor in the 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/RB1) and RB7 (IO_1/RB2), which controls the slave's communication protocols sequences. The slave circuit design has two unused I/O pins: RA2 and RA3. The multiple slaves are a duplication of the following slaves' circuits that have two slaves as presented in Figure 3.

Motor Control Circuit

The motor driver circuit is a standard H-bridge design. A +5V logic voltage source has to be applied before $+V_{CC}$ in this circuit to make it function properly. These bridge on-off controls are made through an IRF530N power MOSFET that can easily handle 10A DC current (IRF530N, 2006). The circuit can control a motor in either forward or reverse

Figure 2. The Master Hardware Circuit



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 a RS latch debounce circuit to produce a clean feedback signal to the slave CPU. The motor circuit design is presented in Figure 4.

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.

The addresses distributions and their control functions of the 9 slaves are presented in Table 2. The predefined command bytes and their associated functions are found in Table 3. Picture 2 shows the real sample communication bit streams on clock (CLK), data out (DOUT), data in (DIN), and framing I/O (I_O1 & I_O2) lines that were captured by a logic analyzer. 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 research demo system is presented in a block diagram format in Figure 5.

Table 2. Slaves' Addresses & Functions

Slave #	Slave Name	Address	**Function Control**
1	PIC 1	0X11	Right Elbow
2	PIC 2	0X22	Right Shoulder up/down
3	PIC 3	0X33	Left Elbow
4	PIC 4	0X44	Left Shoulder up/down
5	PIC 5	0X55	Pivot Torso
6	PIC 6	0X66	Back/ Forth Torso
7	PIC 7	0X77	Right/Left Torso
8	PIC 8	0X88	Right Shoulder Left/Right
9	PIC 9	0X99	Left Shoulder Left/Right

Figure 3. Two Slaves Hardware Circuit

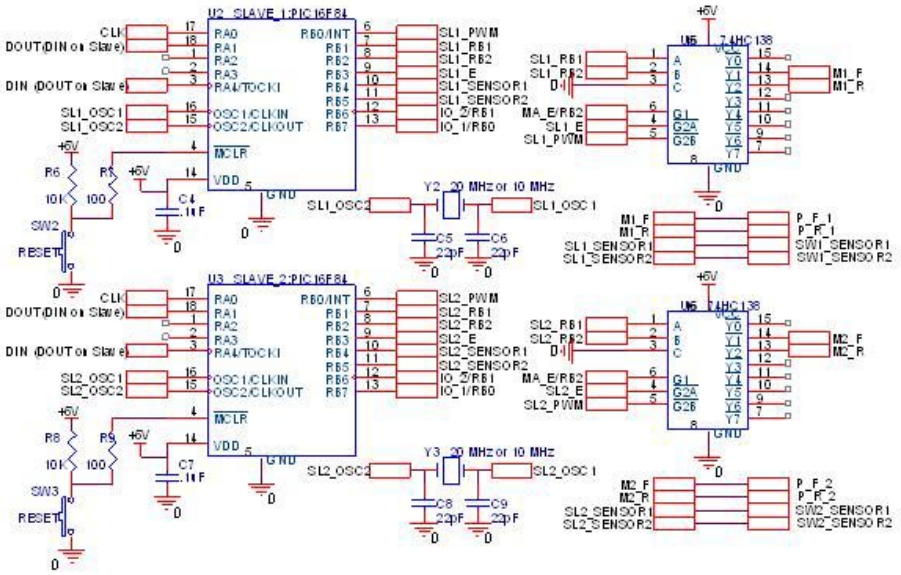
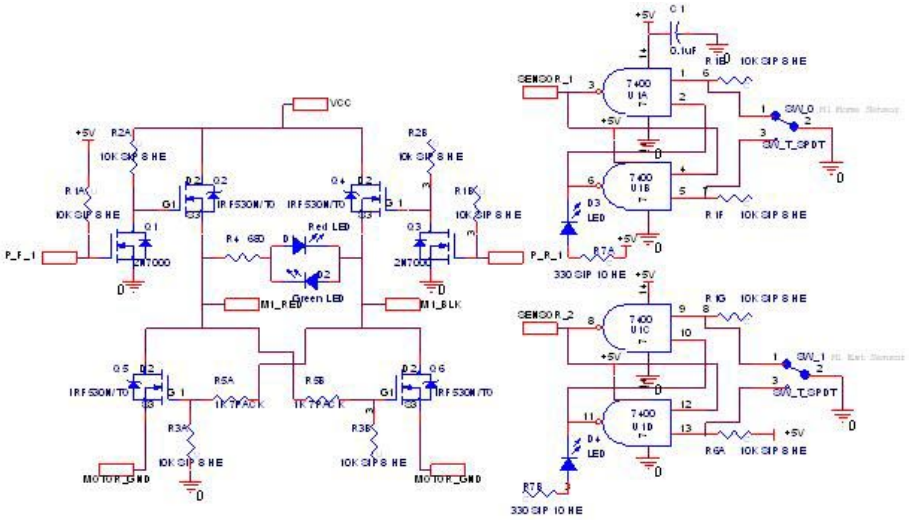


Figure 4. The Motor Control Hardware



Picture 1. One Master & Four Slaves Setup

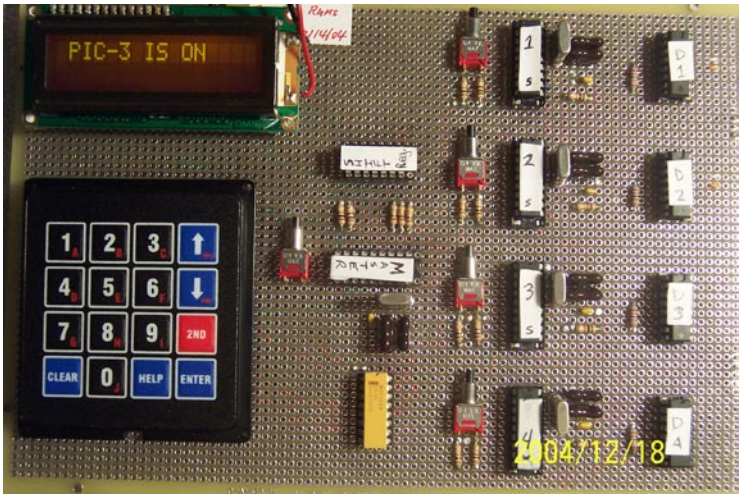


Table 4 represents the estimated cost of the multi-processors system that uses PIC16F84As is compared with the original design using a single MC68HC11 processor. The cost for both systems is very similar, but the flexibility that multi-processor system provides is beyond this assessment.

Conclusion

The designed serial communication protocols with only byte address can have up to 254 slave processors (normally 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 16F88 (same as 16F84A, a 18 pin package) or 16F877A (a 40 pin package) 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 (PIC16F877A, 2004). An implementation of the SLEEP (one of the PIC's instructions) along with proper interrupt wake up strategy in all the CPUs software will make the system more energy efficient (Wilmshurst, 2007). A proper managed time out period of 20 ms to 40 ms can be added to the existing protocol designs to increase the system sensitivity on a miscommunication situation between a master and multiple slaves (Philips, 1997; SMBus2.0, 2001). If security is a concern, adding the CRC-8 implementation in the protocols will be one of the solutions (CRC8, 1999).

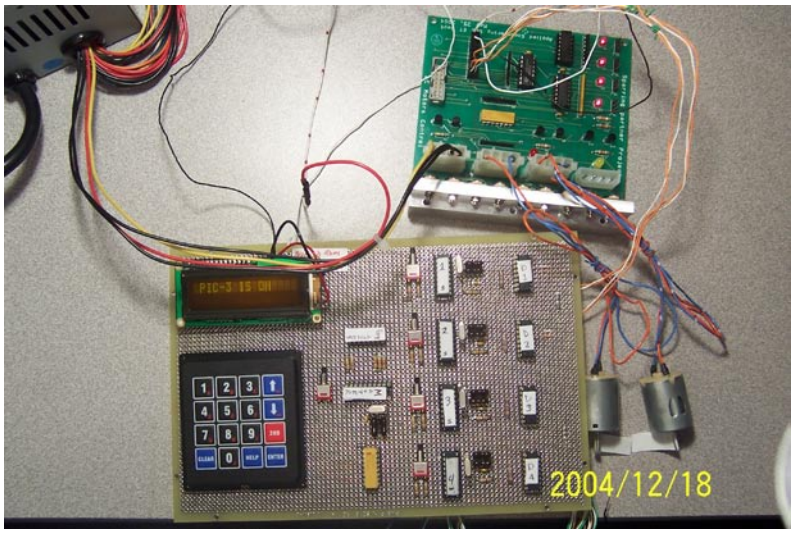
The theory and protocol design (Serial UART Tutorial, 1996; SMBus2.0, 2001) of the synchronous serial communication in the chip level into a practical application enhanced the students' understanding of the potential in their EECT centered career. Here is the summary of what the students have

Table 3. Command Bytes & Functions

#	Command Byte	**Control Function**
1	0XD4	Fast Speed Motor Control, 80% Duty Cycle (PWM)
2	0XD5	Medium Speed Motor Control, 50% Duty Cycle (PWM)
3	0XD6	Slow Speed Motor Control, 20% Duty Cycle (PWM)
4	0X01	Motor Direction Control: Moving Backward
5	0X02	Motor Direction Control: Moving Forward
6	0X20	Time Delay Between Motors Action: Short = 2.04 Seconds
7	0X33	Time Delay Between Motors Action: Medium = 4.02 Seconds
8	0X65	Time Delay Between Motors Action: Long = 78.965 Seconds

Picture 2. The Serial Communication Signals

Picture 3. The Multiple-Processor System Setup



learned on what their accomplished: (Tyson & Ransberger, 2004)

1. As the project approached an end, the students learned how to communicate with each other as team members, which is important when they move on and start their careers.
2. The students designed the communication protocols with multiple microcontrollers, and implemented a project that can be useful in the environment.
3. The students were able to take their knowledge in EECT and apply it to real world application problems.
4. The boxing robot, no matter how simple it was presented, was a great way to enhance understanding of communication, electronics, and teamwork.

These serial communication protocols in multiple PICs were actually implemented in the students' (Tyson McCall and Corinne Ransberger) senior project design, and successfully proven to be suitable in real multiple motors control applications.

The concept of the serial communication is simple and has been utilized in different areas in the real world, such as automation controls (Roy, 2006) and PC networking (Zheng, 2002). Integration of the existing concept in a custom-made design that brings real-life applications into classes, has served one the important missions of the Industrial Technology education: Applied Engineering. Proof of concept research provides the students with an interesting application idea and a better understanding of the links between hardware and software along with their potential applications in the workplaces.

Figure 5. The Demo System Block Diagram

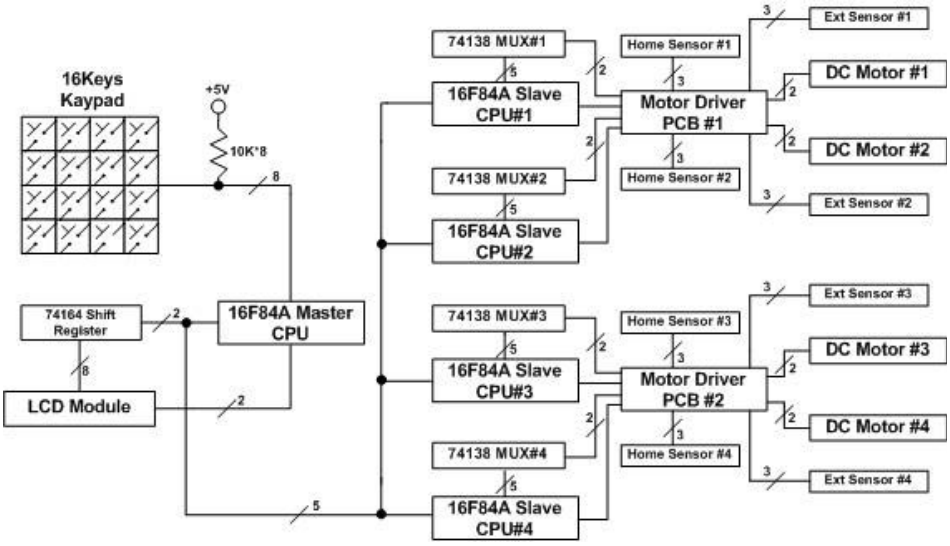


Table 4. The Estimated Cost Comparison between Two Systems

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

References

Bates, M. (2004). *PIC Microcontrollers: An Introduction to Microelectronic* (2nd ed.). Burlington, Massachusetts: Elsevier: Newnes.

Cady, F. M. (1997). *Software and Hardware Engineering, Motorola M68HC1.* New York: Oxford University Press.

CRC8 (1999). *CRC-8 Implementation White Paper, USAR System Inc.* Retrieved May 15, 2005, from http://www.semtech.com.

LCD (2004). *How to control a HD44780 based character LCD.* Retrieved May 15, 2005, from http://home.iae.nl/users/pouweha/lcd/lcd0.shtml.

IRF530N (2006). *International Rectifier.* Retrieved April 11, 2006, from http://www.irf.com.

Motorola (1991). *M68HC11 Reference Manual*. Motorola, Inc., Rev 3.

Motorola (1991). *M68HC11 E Series Programming Reference Guide*. Motorola, Inc.

Philips (1997). *Philips Semiconductors I²C Specification.* Retrieved January 23, 2005, from http://www-us2.semiconductors.philips.com /i2c/news/.

PIC16F84A (2004). *PIC16F84A Data Sheet, Microchip Technology Inc.* Retrieved May8, 2006, from http://www.microchip.com/.

PIC16F877A(2004). *PIC16F877A Data Sheet, Microchip Technology Inc.* Retrieved May 8, 2006, from http://www.microchip.com/.

Roy, Niladri (2006) Connecting Industrial Automation Control Networks *Xcell Journal, Xilinx, Inc.* Second Quarter, 30-33

Serial and UART tutorial (1996). Frank Durda. Retrieved March 21, 2005, from http://www.freebsd.org/doc/en_US. iso88591-1/articles/serial_uart/, Email:uhelm@freebsd.org.

SMBus1.1 (1998). *System Management Bus Specification, Revision 1.1, Smart Battery System Specifications*. Retrieved April 11, 2006, from http://www.sbs-forum.org, Email: battery@sbs-forum.org.

SMBus2.0 (2001). *System Management Bus Specification, Revision 2.0, Smart Battery System Specifications*. Retrieved April 11, 2006, from http://www.sbs-forum.org , Email: battery@sbsforum.org.

TTL (1998). *TTL Logic Data Book*. Dallas, Texas: Texas Instruments.

Tyson, Mc. & Ransberger C. (2004). *Serial Communication: Multiple Processor Control Environment*. Norfolk, Virginia: ODU Senior Project Report.

Wilmshurst, T. (2007). *Designing Embedded Systems with PIC Microcontrollers: Principals and Applications*. Burlington, Massachusetts: Elsevier: Newnes.

Zheng, Y. & Akhtar, S. (2002) *Networks for Computer Scientists and Engineers* New York: Oxford University Press.