Int. J. Appl. Engng Ed. Vol. 7, No. 1, pp. 40–50, 1991
Printed in Great Britain.

0742–0269/91 $3.00+0.00

Microcomputers in the Mechanical Engineering Microprocessor Laboratory at Georgia Tech—Part II*

C. UMEAGUKWU†
M. CHOUCHANE‡
George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332, *U.S.A.*

This paper describes a microprocessor laboratory in mechanical engineering and the corresponding set of students' final group projects. The chip used in this course is the MC68HC11 microcontroller made by Motorola. The course teaches students in the 68HC11 microprocessor instruction set and assembly language programming, the fundamentals of each of the many microcontroller subsystems, and the basics of some electronic components used to interface the microcontroller with external devices. In addition, students gain hands-on experience and the opportunity to apply what they have been taught by completing five experiments and final group project (three students per group). Five examples of students' group projects are presented in this paper. The experiments used for teaching a microprocessor course were presented in Part I.

INTRODUCTION

THE majority of engineering schools have incorporated microcomputer and microprocessor related courses into their curriculum, and have developed the corresponding microprocessor laboratories [1–9]. Most existing microprocessor laboratories are still using 8-bit microprocessors, such as the 8085 [9], or the 6502 [10]; however, newly developed laboratories are beginning to use more advanced 16-bit microprocessors, such as the MC68000 and the Intel 8086 and 286, as well as more advanced microprocessor components [12–14].

This paper describes the theory and design of the microprocessor laboratory and a set of five projects. The laboratory is designed to be taken concurrently with a three credit quarter hour senior elective microprocessor applications course. The prerequisites for this course are: (1) an introductory course on microcomprocessors which covers Boolean algebra, assembly language programming, d.c. motors and digital arithmetic; and (2) control.

The present setup of the lab has evolved from an earlier microprocessor course based on MC6801. The experience gained there was invaluable in setting up this laboratory. In this earlier microprocessor course, the microcontroller used was the earlier generation of MC6800 series, MC6801 [15–17]. The chip was purchased by itself and distributed to the students. The students were then required to install the chip on a breadboard and perform all the wiring necessary for the chip to be operational. The minimum system for the 6801 required a clock generator, a serial communication interface, reset logic, and a host terminal. The versatile Commodore VIC 20 computer was used as the host terminal for communication with the 6801's on-chip software monitor. The monitor program enabled the user to develop software on the 6801.

The earlier laboratory setup proved to have many disadvantages. The breadboard wiring required by the student to create a minimum system for the 6801 was fragile, blew fuses and damaged chips too frequently. The minimum system utilized only the on-chip memory of the 6801 which proved to be too small for anything but the most modest of programs. Expansion of the system memory required the addition of a separate memory chip. Lastly, the VIC 20 setup provided only the means to communicate with the 6801's on-chip monitor program. The setup was unable to provide the capability to write assembly language programs on the VIC 20, assemble the programs into machine language and download into 6801 memory. The shortcomings made the process of software development slow and hindered the implementation of the overall objective of the microprocessor course.

A new microprocessor lab setup was sought which addressed the shortcomings of the previous hardware arrangement. The intention was to avoid the students themselves having to perform the basic wiring necessary to operate the microcontroller. This ruled out the purchase of individual

* Paper accepted 31 May 1990.
† Charles Umeagukwu is Assistant Professor.
‡Mnaouar Chouchane is graduate Research Assistant.

chips. What was needed was a ready-to-go microcontroller system contained on a single circuit board and requiring a minimum of setup to operate. We also wanted a system where students could move easily between the stages of writing, assembling and debugging microcontroller programs. Motorola sells a low-cost single board 68HC11-based system for microcontroller software development called the M68HC11EVB [18–21].

The EVB board is relatively cheap. It is a versatile product which contains 8K of RAM for program storage, ROM-based monitor software to facilitate software development, and a built-in RS-232C communications interface. The communications allow the board to be connected to an industry standard IBM-PC or compatible host computer. The hefty 8K or RAM provides more than enough storage for user programs. Add to this system text editing software, 68HC11 cross-assembler software, and communications software for the PC and the result is a powerful stand-alone 68HC11 development system. With such a system, software can be written in assembly language on the PC, assembled into a machine language program, and downloaded into the RAM on the EVB board via the communications interface where it can then be debugged or executed via the monitor software. Motorola offers 68HC11 cross-assembler software and communications software to purchasers of the EVB board free of charge. However, the authors were able to acquire a student-written program to form the software component of our system. The software is capable of handling the PC to EVB communications, assembly of user programs, and downloading of resulting machine language files to EVB memory all in one integrated package. The software package fully meets our objective of allowing the students easy movement between the stages of writing, assembling and debugging microcontroller programs.

Selection of the Motorola EVB board allowed us to contruct a relatively low-cost software development system for use in the Microprocessor Laboratory. The complete hardware and software system allows us to realize the capabilities that were unavailable with the previous setup. This paper demonstrates the philosophy of this laboratory through the description of five students' final projects.

DESCRIPTION OF THE FINAL PROJECTS

Students are required to demonstrate an in-depth understanding of both the theoretical and hands-on aspects of the microprocessor course by completing a final group project. They are split into groups of three students during the first week of class. They are asked to use the knowledge gained in the class lectures and the laboratory exercises to come up with their own unique project. The project is due during the last week of classes, and it is accompanied with formal group presentations. Each group is required to present a working project. Of course, there is initial informal presentation by each group to the instructor who advises the group whether the project is too ambitious or too frivolous. The description of five of these projects, including the title of the projects, interface procedures, and pertinent drawings are presented below.

Watchdog Security System

The objective of this project was to use a Motorola MC68HC11 as a controller for an alarm system. A program for the 68HC11 was written to monitor the prototype alarm circuitry built in the laboratory as shown in Fig. 1. The program flow chart is shown in Fig. 2. The program will allow an individual to input a security code, enter and exit delays to arm the system; monitor doors, windows and other sensors; and indicate the location where security was breached.

This watchdog security system consisted of a monitor attached to the microcontroller. Sensors were strategically placed throughout the circuitries that simulated doorways, windows, hallways, stairways and storage rooms as shown in Fig. 3. Anytime there was a disturbance, the microcontroller set off an alarm while the source and location of the disturbance was displayed on the monitor.

Regardless of the sensor type (magnetic, vibration or sonar), its output must be high when there is no disturbance. When there is a disturbance, its output goes low and the buzzer sounds. The advantage of this configuration is that if the sensor

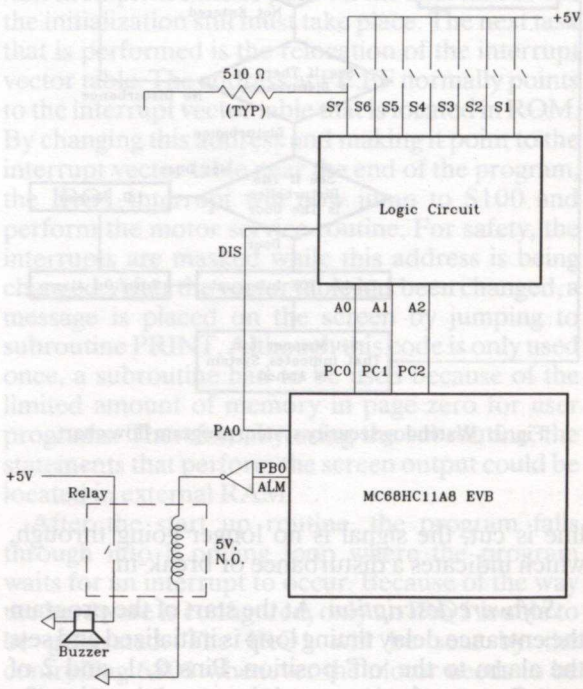

Fig. 1. Prototype alarm circuitry.

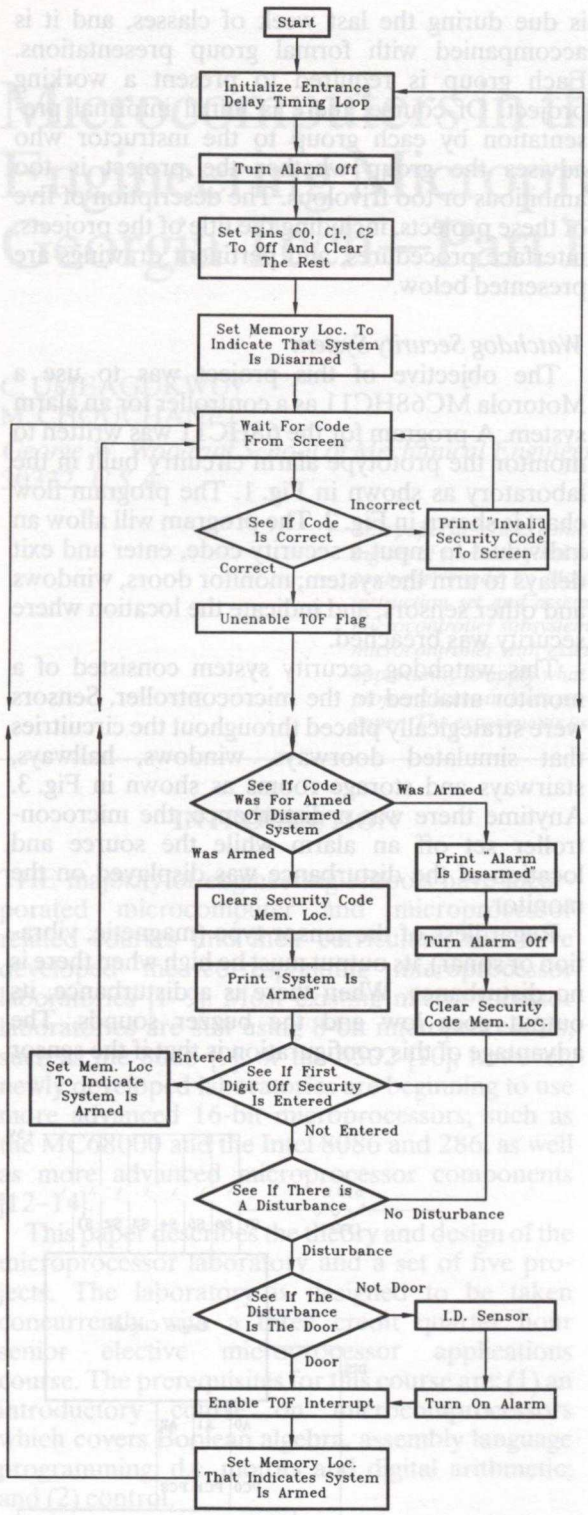

Fig. 2. Watchdog security system program flow chart.

line is cut, the signal is no longer going through, which indicates a disturbance or break-in.

Software description. At the start of the program the entrance delay timing loop is initialized and sets the alarm to the 'off' position. Pins 0, 1, and 2 of port C are set for input and the remaining pins (3–7) are grounded to keep them always clear from

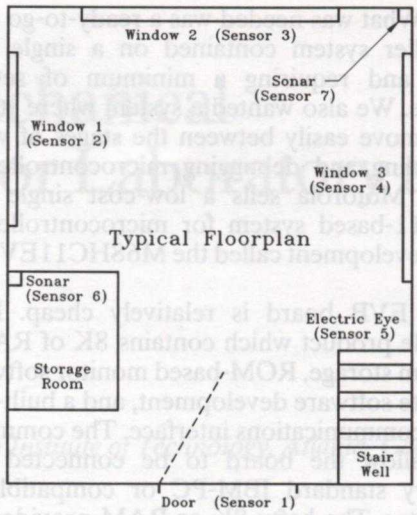

Fig. 3. Typical floor plan and sensor locations.

any uncontrollable disturbance. Next, port C is set to indicate that the controller is initially disarmed.

The controller now waits for the security code to be input into the computer. If the wrong code is entered the computer prints on the screen 'INVALID SECURITY CODE'. The correct code causes the timer overflow flag to clear and checks if coded entry was for arming or disarming the system. When the coded entry is for disarming, the software clears the security code memory locations while printing on the screen 'SYSTEM IS DISARMED' and jumps back to the start of the program and waits for arming security code. For arming, the program first checks to see if perimeter is secure, i.e. all doors and windows closed. If not secure 'PERIMETER NOT SECURE' is printed on the screen and arming sequence begins over again. When the system is secure an exit delay begins. During the delay routine the program continually checks for sensor trips and the alarm will sound for any disturbances except the door opening. If the door is still open when the exit delay runs out, the program goes into extended exit delay. Once again any disturbances other than the door will set off the alarm. Upon the door closing the extended or normal exit delay, the computer will print 'SYSTEM IS ARMED' on the screen.

When the program is armed, it is continuously checking to see if there is either a disturbance, or if first digit of the security code is entered. The security code is 1-2-3-4, so the program tests the keyboard to see if a '1' has been entered. If other than a '1' is entered, it ignores the entry and stays in its loop. However, if a '1' is entered, the program jumps to the 'INPUTS SECURITY CODE FROM KEYBOARD' section of the program. The program will jump back to the 'SYSTEM IS ARMED' section if the rest of the security code is entered incorrectly. The program reads the code as a disarming code and shuts the alarm system off. If a disturbance in sensors occurs then the program checks whether the door was opened. If it was not

opened the system prints to the screen the sensor number and turns on the alarm. When the alarm is set the program waits for the correct disarming code. If the door was opened, the timer overflow interrupt mask is enabled which begins the entering delay in the interrupt vector. The interrupt vector first clears the timer overflow flag and begins incrementing a counter memory location while checking whether or not maximum count was reached. While the interrupt vector is counting, the individual who opened the door enters the security code to turn off the alarm. If the person successfully enters the code before the time runs out (15 sec.), the control system shuts off and 'SYSTEM IS DISARMED' is printed on the screen. If the person unsuccessfully enters the security code the alarm will sound and the interrupt will be disabled. Once the alarm is sounded, the program waits for the correct security code to turn the alarm off. When the system is disarmed, the program goes to the beginning and begins the process over again.

Automated Positioning and Capping of Bottles

This project is about the application of a microprocessor controlled stepper motor for the automated positioning and capping of a bottle. The microprocesser used in this project is MC6801. Specifically, a robot places the bottles on a round table that is rotated by the stepper motor. In this project a human hand was used to simulate the robot manipulator. Bottles, on a rotating table are sequentially positioned under a mechanism which will cap them. An angular interval of 30° separates each bottle. However, in this case the capping mechanism cannot be activated until the next bottle to be capped has been properly positioned. For the project, one MC6801 was used to simulate the capping mechanism while another was used to control the positioning of the table, and hence the bottle to be capped. The capping 6801 must wait until the positioning stepper motor has successfully positioned the bottle under the capping mechanism. A ready state is signified by a high level on a line running from an output pin on motor 6801 to an input pin on capping 6801. When the ready signal is received, the capping 6801 enters a 3-second delay signifying the capping of the bottle. At the conclusion of the delay, an interrupt signal is generated by creating a low level on a line running from an output pin of capping 6801 to the IRQ2 interrupt pin on the 6801 that controls the motor. This interrupt causes this 6801 to enter a service routine which sets the ready line to low level, signifying that it is busy, and begins sending the required stepping sequence to the stepper motor to step it 30° thereby positioning the next bottle to be capped. The low level on the ready line tells the capping 6801 that the bottle is being positioned. When the 6801 has stepped the motor the required 30° it sets the ready line high again signifying that the bottle is now ready to be capped. Reception of the ready signal by the capping 6801 tells it to begin capping the bottle and the process begins all over again. The circuit that was designed for this project is shown in Fig. 4.

A stepper motor is a d.c. machine which has four field windings wound in such a way as to create a number of stable poles evenly spaced around the circumference of the outer motor housing. Simultaneously, energizing the four windings in a specific way results in the rotation of the rotor so as to align with the next stable pole. The stepper motor used in the project has 48 poles, so the smallest step angle is $360°/48 = 7.5°$. Continuous rotation of the stepper motor shaft is achieved by energizing/deenergizing all four motor windings following a sequential pattern which is given in the product literature. For the stepper motor used in the project, the sequence for clockwise rotation is as follows:

Motor Winding	1	2	3	4	
Condition:	off	on	off	on	clockwise
	on	off	off	on	
	on	off	on	off	
	off	on	on	off	
	off	on	off	on	

Software description. This program has two major parts; a start-up routine and an interrupt service routine that turns the motor when requested.

The start up routine, executed only once when the program is first started, takes care of configuring the 6801 for its task. Ports 1 and 4 are configured for all output. Port 1 is configured for output because the chip will be communicating with the controlling 6801 via pin P17. Port 4 is configured for output so that port 4 can be used as an address bus for the added RAM. In extended non-multiplexed mode port 4 is the address bus but the initialization still must take place. The next task that is performed is the relocation of the interrupt vector table. The address at FE:FF normally points to the interrupt vector table that is located in ROM. By changing this address and making it point to the interrupt vector table near the end of the program, the IRQ1 interrupt will now jump to $100 and perform the motor service routine. For safety, the interrupts are masked while this address is being changed. After the vector table has been changed, a message is placed on the screen by jumping to subroutine PRINT. Although this code is only used once, a subroutine had to be used because of the limited amount of memory in page zero for user programs. Therefore, by using the subroutine, the statements that perform the screen output could be located in external RAM.

After the start up routine, the program falls through into a polling loop where the program waits for an interrupt to occur. Because of the way the hardware is configured, only an IRQ1 is able to be generated. The IRQ1 will be sent by the controlling 6801 whenever the motor needs to be turned. The first thing that occurs in the loop is line P17 is set high without altering the rest of port 1.

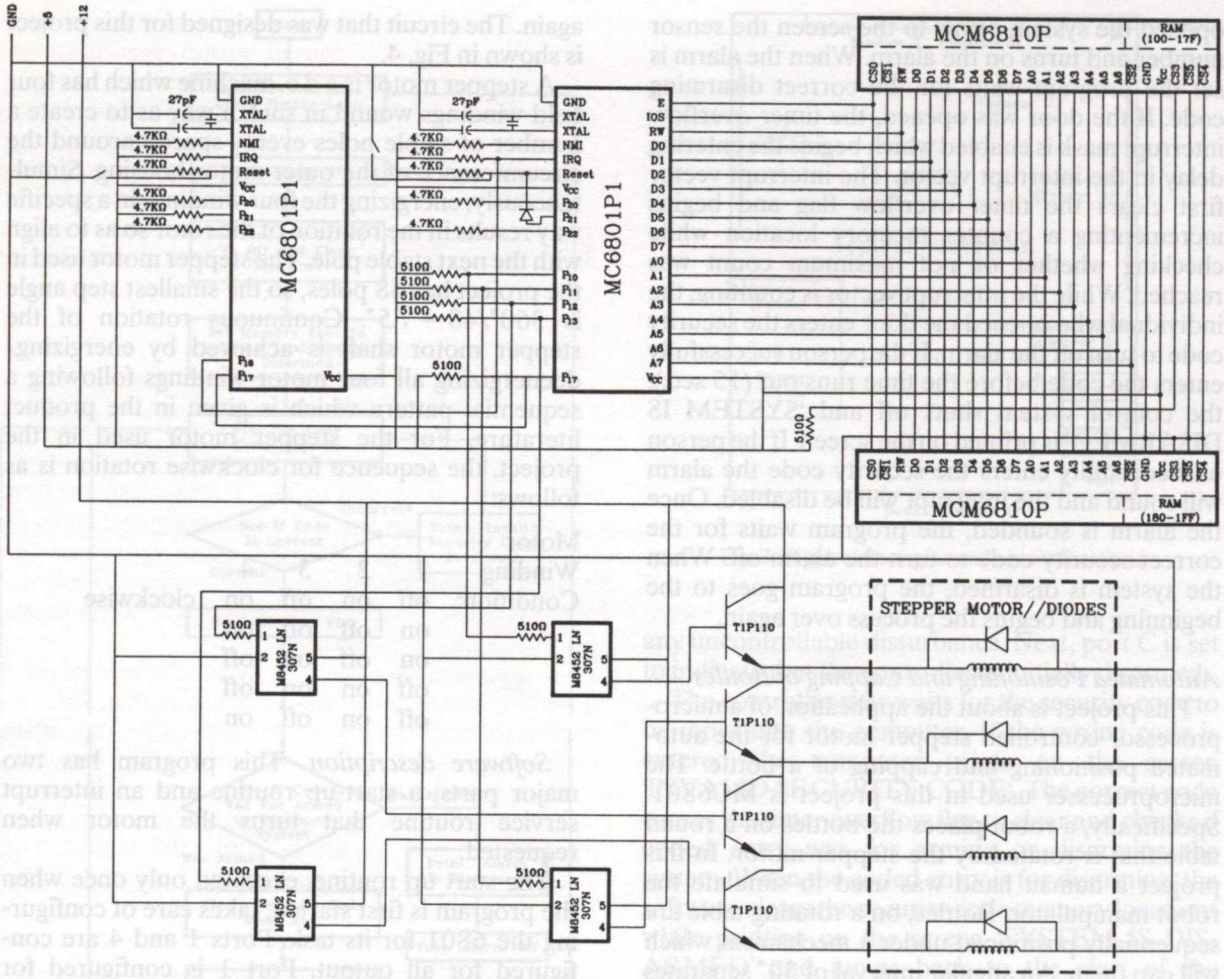

Fig. 4. Circuit diagram for automated positioning and capping of bottles.

This had to be done because if the codes that are sent to the stepper motor are changed, it is possible that the motor will only oscillate instead of turning. Line P17 serves as a ready line which tells the controlling 6801 that the motor controller is ready to turn the motor. After line P17 is set high, maskable interrupts are enabled and then the processor waits for an interrupt. When the processor executes the wait for interrupt command, the machine state is dumped to the stack and then monitors the interrupt lines until an interrupt occurs. When an interrupt does occur, the processor jumps to location FFF8:FFF9 to obtain an interrupt vector. However, this vector points to a subroutine in the Lil'bug monitor that uses the vector table pointed to by FE:FF to find the desired interrupt handling routine. The use of two interrupt vector tables, one at FFF1:FFFF and another which is pointed to be FE:FF, is done so that if the user does not wish to reset interrupt vectors, then the 6801 is already configured with one in the Lil'bug monitor ROM. This happens because on REST the address at FE:FF is FFC8, the starting address of the ROM vector table. Once the interrupt service routine is through it branches back to the beginning of the polling loop.

The other part of the program is the interrupt service routine which is used to turn the stepper motor whenever an IRQ1 is received. This routine starts at address $100 and is pointed to in the new interrupt vector table. The first thing to occur is interrupts are masked so that it is not possible to receive an interrupt during interrupt servicing. This is needed because it is possible to overflow the stack and crash the program if many interrupts were received simultaneously. The next step that occurs is pin P17 is set low without altering the rest of port 1. Once again, this had to be done to prevent possible motor oscillation. The next part of the program is a loop which will send four codes to the motor to make it turn 30°. The loop is performed by decrementing a counter variable and the motor codes are accessed by indexed addressing. The codes are changed by decrementing the x register, thus moving up one location in the motor code table. After a code is obtained, it is sent to port 1. A delay routine is used to send codes to the motor at the rate of 200 pulses per second. Once the four codes have been sent to the motor, the interrupt routine is finished and returns to the polling loop.

Elevator Control

The object of this project is to control an elevator (or a similar mobile vehicle) with two inexpensive

d.c. motors. The first motor drives the elevator in the upward direction, while the other drives the elevator in the downward direction. The final goal is to make the elevator respond to passengers' requests in a logical manner and also to provide a smooth ride. The Motorola MC68HC11 is used as the core of the elevator control system. It is programmed to drive the elevator motors so that the elevator performs its expected function and behaves according to the required specifications.

Figure 5 shows a simplified layout of the elevator driving system. An emergency braking system is included in order to keep the elevator under control when both piniono are unloaded. This corresponds to a failure of one or two of the driving gears to transform the action of the motor elevator. The elevator is then slowly driven to the ground floor. Two d.c. motors are used in this project but in an actual application, a larger reversible motor will have to be chosen. The choice of the motors depends mainly on the maximum load of the elevator and the required speed. Figure 6 shows the circuitry of the interface board. Two of these circuits were built; one circuit for each 6811.

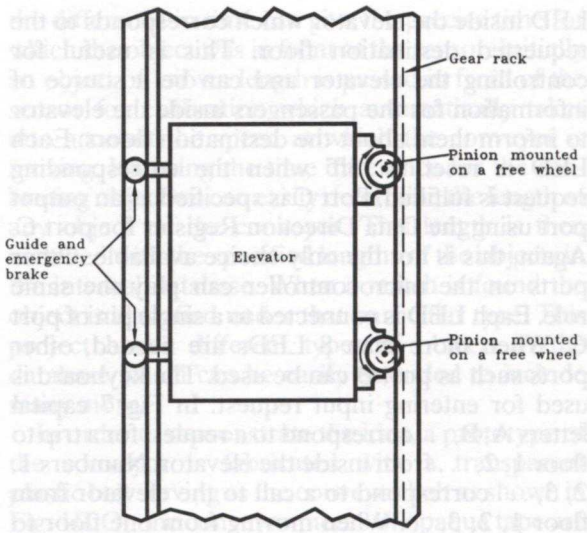

Fig. 5. Elevator driving diagram.

Description of the Circuitry. In addition to the circuitry used to control the motors, the wiring circuitry includes two light emitting diodes (LEDs) for each floor of the building which uses the elevator. Each floor has an LED inside the elevator which is turned on when a trip to that floor is requested from inside the elevator. A second LED is located at each floor near the elevator door, and it is turned on whenever the elevator is called for that floor. In addition, the requests generated from a call to the elevator from one floor turns on the

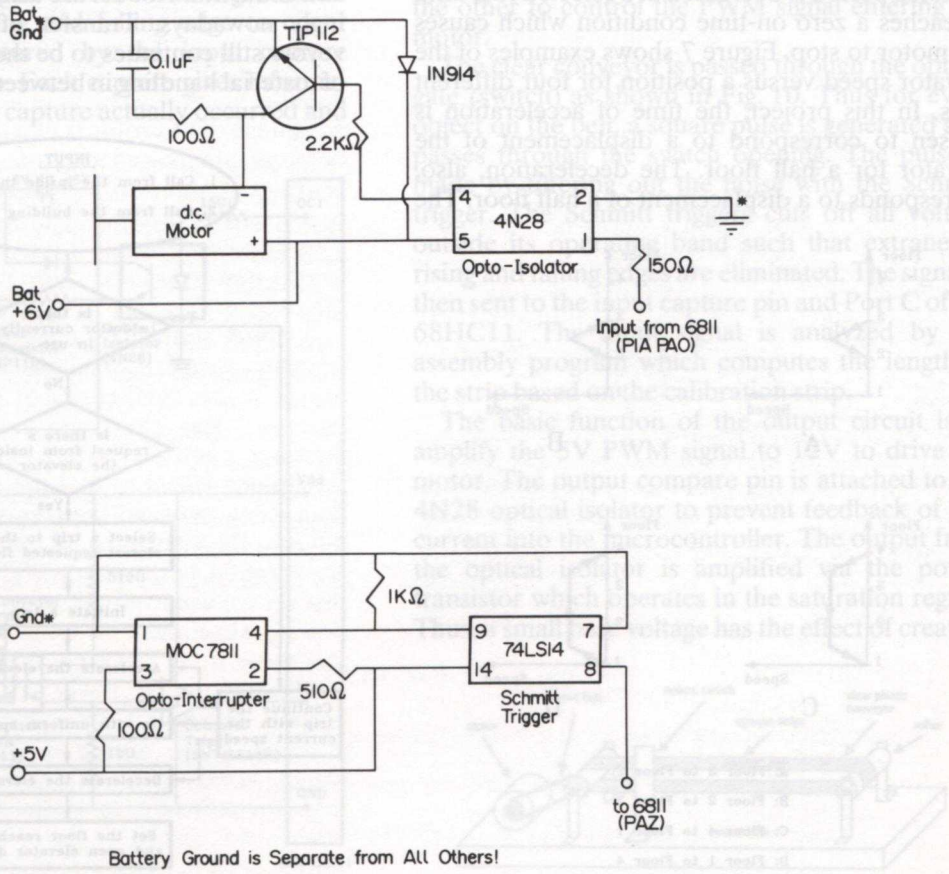

Fig. 6. Circuit diagram for elevator control.

LED inside the elevator which corresponds to the requested destination floor. This is useful for controlling the elevator and can be a source of information for the passengers inside the elevator to inform them about the destination floors. Each LED is reset to 'off' when the corresponding request is fulfilled. Port C is specified as an output port using the Data Direction Register for port C. Again, this is not the only choice available. Other ports on the microcontroller can play the same role. Each LED is connected to a single pin of port C. When more than 8 LEDs are needed, other ports such as port B can be used. The keyboard is used for entering input request. In Fig. 7 capital letters A, B, ... correspond to a request for a trip to floor 1, 2, ... from inside the elevator. Numbers 1, 2, 3, ... correspond to a call to the elevator from floor 1, 2, 3, ... When moving from one floor to another, the elevator should provide a smooth ride. Hence, the elevator should undergo three phases of speed: (1) accelerate uniformly for a certain time, (2) move with a uniform speed toward the destination floor leaving a given distance for deceleration (3) Decelerate uniformly until the elevator reaches a full stop which should correspond to the destination floor.

The acceleration of the motor is achieved by increasing the on-time of the square wave voltage fed to the motor at a chosen rate. In the uniform speed, the on-time voltage is maintained constant. In the deceleration mode, the on-time voltage of the fixed period is decreased at a selected rate until it reaches a zero on-time condition which causes the motor to stop. Figure 7 shows examples of the elevator speed versus a position for four different trips. In this project, the time of acceleration is chosen to correspond to a displacement of the elevator for a half floor. The deceleration, also, corresponds to a displacement of a half floor. The time of acceleration and deceleration can be modified for convenience. Similarly, the maximum speed can be controlled as desired.

The project was completed for a three-story elevator. This limitation was necessary in order to validate the control method and to finish the project in the limited time assigned for a class project. The extension of the project to an elevator serving any number of stories was kept in mind during the writing of the control program. Hence, we expect that the extension of the project, for more than a three-story elevator, should be accomplished with an addition of a limited amount of code and without a major additional effort. Also, the elevator is currently configured to respond first to the lowest floor among the requested floors. This is of course not the correct technique to control the motion of an elevator, but this step was appropriate for the early development of the system and the limited time. Similarly, correcting this deficiency is expected to present limited difficulty. The program flow chart is shown in Fig. 8.

Automated Material Handling and Identification System

Material handling operations are most commonly achieved by using conveyors, cranes, hoists and industrial trucks. With an increased emphasis on automation, Automated Guided Vehicles (AGV), Automated Storage and Retrieval Systgems (ASRS), Tote Stackers, Carousels, Miniloads and Industrial Robots are also being used increasingly nowadays. Transfer of materials on conveyors still continues to be the most popular form of material handling in between these systems. This

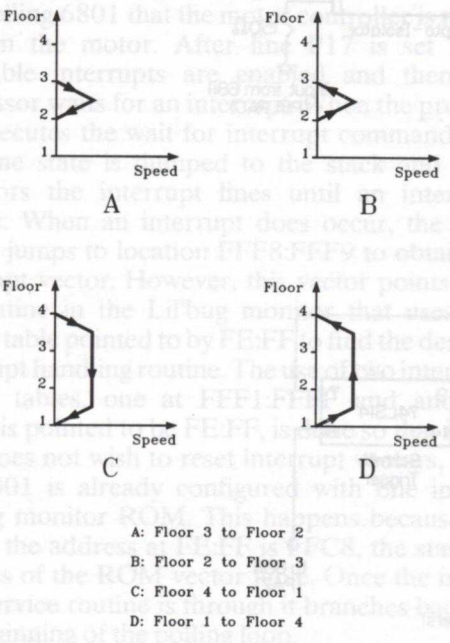

Fig. 7. Elevator speed versus position for four different trips.

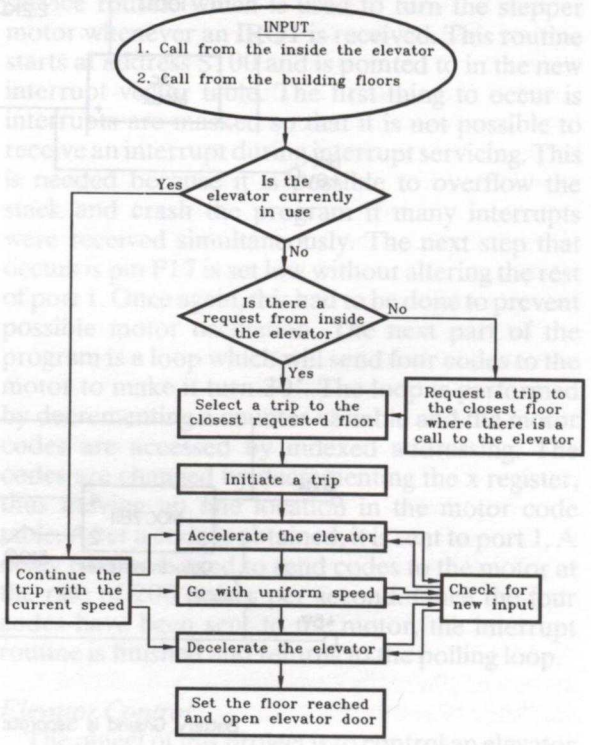

Fig. 8. Program flow chart for elevator control.

method of transfer is most commonly seen on assembly lines where hundreds of components are continuously moved to the assembly points, get assembled and continue further on to the next assembly point as an assembled part. A similar picture can also be seen in the receiving/shipping sections where the incoming and outgoing goods, in the form of packages, are lined up on conveyors. With hundreds of components passing on these lines in a haphazard manner, it is extremely important to keep track of the types and the number of components or packages on them. Sophisticated and expensive vision systems are being utilized by several companies to achieve this function. Bar code readers are also extensively used to identify the packages.

This project proposes a very simple approach to achieve a fast and dependable status control of materials. The cost of implementation is cut down to a negligent minimum in contrast to the complicated devices being presently employed. This project will give the system the desired flexibility that is important in today's dynamic workplace. The timer function of the Motorola's MC68HC11 single chip microcontroller has been shown to enable rapid and accurate counting of components on assembly lines or packages on shipping/receiving lines. In order to achieve counting and classification of objects this method utilizes the timer resource on the microcontroller. The objects on a conveyor line can be detected by passing through an optical gate as shown in Fig. 9. Each object causes a rising and falling edge to occur which can be captured by the input capture function on the chip. Each capture gives information about when the capture actually occurred and the difference in time can give the precise time for which the object was in front of the sensor. Initially an object of known length is passed in front of the sensor for calibration which automatically takes into account the speed at which the conveyor is moving. Knowing the time difference for the calibrating object we can easily determine the length of any object on the conveyor. This length is then matched with the specified lengths of the objects in the internal database. When a match is found the object is classified under that particular type. This project has 3 different types of objects in the database which can be easily extended to include many more.

In order to demonstrate the idea, a prototype of the conveyor is fabricated with a transparent plastic belt serving as the conveyor belt as shown in Fig. 10. On the conveyor pieces of opaque tape are fixed which represent objects on the conveyor. The conveyor is driven using a d.c. motor which in turn is driven by the program using pulse width modulation (PWM). This involves applying a square wave to the motor and by varying the period of the wave, the average voltage applied to the motor can be varied. The motor is driven using an open loop control scheme. A closed loop control scheme can also be easily implemented if deemed necessary.

Description of the Circuitry. The motor control circuit shown in Fig. 9 is composed of two circuits, one to measure the width of the opaque strips and the other to control the PWM signal entering the motor.

The clear conveyor is passed through the optical gate (switch) as shown in Fig. 10. Thus for every object on the belt, a square pulse is generated as it passes through the switch opening. The pulse is made by filtering out the noise with the Schmitt trigger. The Schmitt triggers cuts off all voltage outside its operating band such that extraneous rising and falling edges are eliminated. The signal is then sent to the input capture pin and Port C of the 68HC11. The input signal is analyzed by the assembly program which computes the length of the strip based on the calibration strip.

The basic function of the output circuit is to amplify the 5V PWM signal to 12V to drive the motor. The output compare pin is attached to the 4N28 optical isolator to prevent feedback of any current into the microcontroller. The output from the optical isolator is amplified via the power transistor which operates in the saturation region. Thus, a small base voltage has the effect of creating

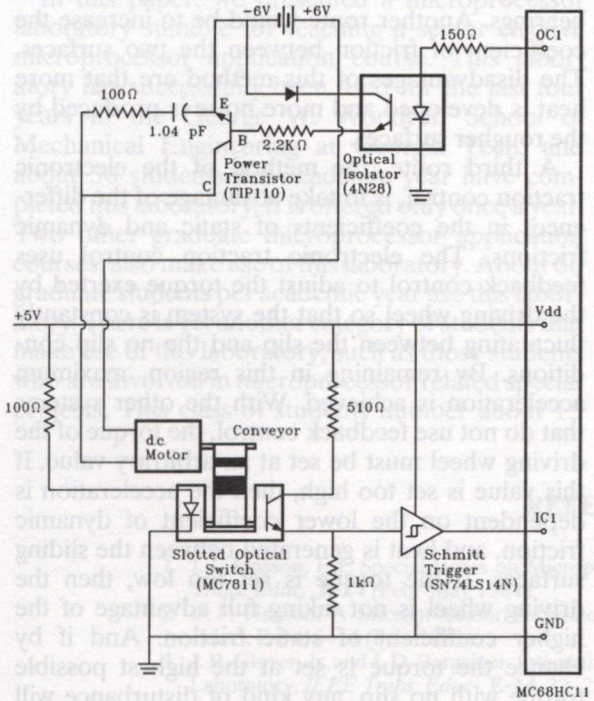

Fig. 9. Motor control circuit for automated material handling.

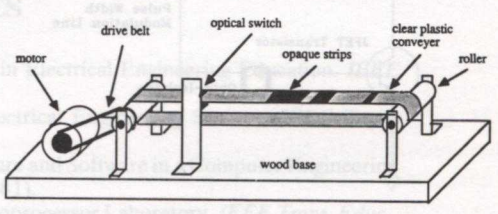

Fig. 10. Experimental setup for prototype conveyor system.

a short circuit between the collector and emitter. The result is that the circuit is closed and a voltage is sent to the d.c. motor. The capacitor is included to add damping to the system, while the diode allows the capacitor to discharge during the off time.

This project gives an idea of the usefulness of the microcontroller as a powerful tool for controlling different devices. It provides a foundation as to how the microcontroller can be efficiently used to control all kinds of motors which are the drivers for robots, automated material handling systems, numerically controlled machine tools and a wide variety of other applications. Another important aspect of the microcontroller is, as the name suggests, its small size and weight which enables it to be incorporated into any device without requiring much space. The high speed of the microcontroller can also be efficiently used for certain applications like counting of components on an assembly line, packages on a conveyor etc. by making use of the ideas gained from this lab.

An Electronic Traction Control System

An electronic traction control system was developed which utilized the Motorola MC68HC11 microprocessor and some other components as shown in Fig. 11. This system monitors the slip between a driving wheel and a driven wheel and adjusts the torque of the driving wheel to stop any slipping that may occur. The system can be utilized wherever slip between two contact surfaces is destructive.

Many situations exist where maximum acceleration of a driven wheel by driving wheel is needed.

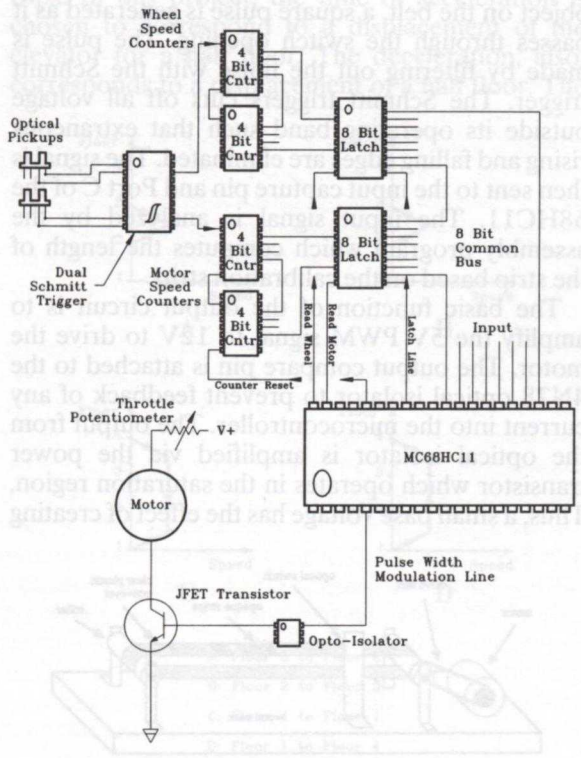

Fig. 11. Control circuitry for electronic traction control.

The upper limit of this acceleration is dependent on the frictional force developed from the wheel to wheel contact. This frictional force is dependent on two factors: the normal force imposed on the wheels as shown in Fig. 12, and the coefficient of friction between the two surfaces. The normal force on the wheels is dependent on how tightly the wheels are pushed together. The coefficient of friction is dependent on the properties of the contact surfaces.

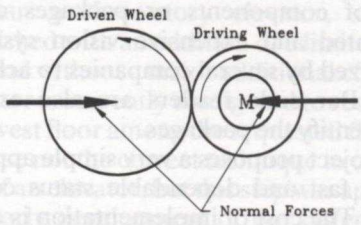

Fig. 12. Normal force on the driving and driven wheels.

To accelerate a wheel by another wheel, several force transmissions must occur. First, an external device must produce a torque, usually by some type of motor, and this torque must be transmitted to the driving wheel via a drive shaft or some other equivalent device. Second, the torque of the driving wheel must be transmitted from the surface of the driving wheel to the surface of the driven wheel. The driven wheel is then accelerated by this force.

To maximize the acceleration of the driven wheel, two routes can be pursued. The normal force can be increased or the frictional force can be increased. Increasing the normal force has been the practice since it is easily accomplished. The disadvantage of this route is that the load on the wheel bearings is increased which decreases the life of the bearings. Another route would be to increase the coefficient of friction between the two surfaces. The disadvantages of this method are that more heat is developed and more noise is produced by the rougher surfaces.

A third route, the method of the electronic traction control, is to take advantage of the differences in the coefficients of static and dynamic frictions. The electronic traction control uses feedback control to adjust the torque exerted by the driving wheel so that the system is constantly fluctuating between the slip and the no slip conditions. By remaining in this region, maximum acceleration is achieved. With the other systems that do not use feedback control, the torque of the driving wheel must be set at an arbitrary value. If this value is set too high, then the acceleration is dependent on the lower coefficient of dynamic friction, and heat is generated between the sliding surfaces. If the torque is set too low, then the driving wheel is not taking full advantage of the higher coefficient of static friction. And if by chance the torque is set at the highest possible torque with no slip, any kind of disturbance will throw the system into a slip condition.

Description of the Circuitry. The general hardware for the simulation of an actual system consists of five components: a Motorola MC68HC11 chip on an EVB board, a motor driving a small light plastic wheel simulates the drive mechanism of the system, a large heavy wheel simulates the driven wheel whose desired acceleration is at a maximum for the given conditions, a potentiometer simulates an accelerator which controls the voltage sent to the motor, a circuit counts the rotations of the wheels which is very similar to what would actually be used.

Electronic Traction Control is a system that would be very advantageous if applied to automobiles. Frequently, automobiles can encounter surfaces with reduced traction, such as icy or wet roads. If an automobile was equipped with Electronic Traction Control, wheel spin would be eliminated allowing the automobile to accelerate at a maximum for the given conditions.

There are two different ways that the motor speed can be controlled: throttle control or ignition control. The throttle control would involve placing a solenoid online with the throttle linkage. When the solenoid is energized the operator would have direct control of the throttle. But as soon as the back wheels begin to spin faster than the front wheels the ontime of the solenoid would be decreased until the speeds matched. The alternative method of control would be to cut the ontime of the ignition system until the wheel speeds matched. The program flow chart used in this project is shown in Fig. 13.

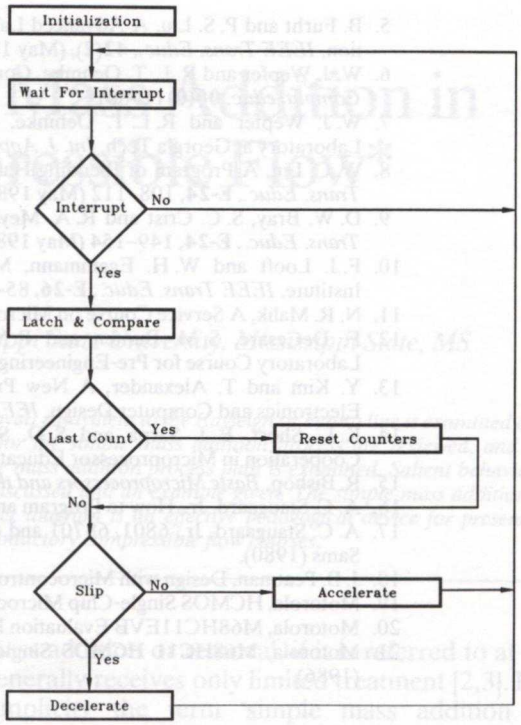

Fig. 13. Program flow chart for electronic traction control.

CONCLUSION

In this paper, we presented a microprocessor laboratory suitable for teaching a senior elective microprocessor application course. This laboratory has successfully been used for the last four years in the George W. Woodruff School of Mechanical Engineering at Georgia Tech, and about 30 students per academic year have completed this laboratory. It is offered only once a year. Two other graduate microprocessor application courses, also make use of this laboratory. About 60 graduate students per academic year use this laboratory. There is yet another category of students that make use of this laboratory, such as those students who are involved in microprocessor related special projects. This class of students number about 15 per academic year. The feedback from those who have completed this lab along with the elective microprocessor applications design course, especially those who took industrial positions, has been both positive and encouraging.

Five projects using the MC68HC11EVB microcomputer have been described in this paper. These projects introduce various aspects of microprocessor-based design and the students get familiar with various types of interfacing techniques. It is important to note that this lab and the course get high reviews from students. This is because they are exposed to the types of course work and projects that are generally associated with electrical engineering, but this time with mechanical applications.

Acknowledgements—The authors would like to express their appreciation to all the students who contributed in one way or the other in the projects presented here. We would also like to thank SME foundation, and CIMS program at Georgia Tech for providing equipment grant and curriculum development funding for this laboratory. Finally, we are grateful to Motorola Corporation for donating the MC68CH11EVBs and MC6801 used in these projects.

REFERENCES

1. J. J. Jonsson, Ed., Special Issues on Microprocessors in Electrical Engineering Education. *IEEE Trans. Educ.*, **E-24** (Feb.–May 1981).
2. D. F. Hanson, A Microprocessor Laboratory for Electrical Engineering Seniors. *IEEE Trans. Educ.*, **E-24**, 8–14 (Feb. 1981).
3. J. R. Glover, Jr. and J. D. Bargainer, Integrating Hardware and Software in a Computer Engineering Laboratory. *IEEE Trans. Educ.*, **E-24**, 22–27 (Feb. 1981).
4. S. V. Iyengar and I. L. Kinney, A Design-Oriented Microprocessor Laboratory, *IEEE Trans. Educ.*, **E-24**, 43–46 (Feb. 1981).

5. B. Furht and P. S. Liu, A Advanced Laboratory for Microprocessor Interfacing and Communication, *IEEE Trans. Educ*., **42**(2), (May 1989).
6. W. J. Wepfer and R. L. T. Oehmke, Computer Based Data Acquisition in the Undergraduate Lab, *Comput. educ*. **00(0)** (1986).
7. W. J. Wepfer and R. L. T. Oehmke, Computers in Mechanical Engineering Instrumentation Laboratory at Georgia Tech, *Int. J. Appl. Engng. Ed.* **1**, 415–421 (1985).
8. W. C. Lin, A Program of Blending Hardware/Software on Microprocessors in Education, *IEEE Trans. Educ*., **E-24**, 108–112 (May 1981).
9. D. W. Bray, S. C. Crist and R. A. Meyer, A Microcomputer Laboratory and Its Courses, *IEEE Trans. Educ.*, **E-24**, 149–154 (May 1981).
10. F. J. Looft and W. H. Egginmann, Microprocessor-based Projects at Worcester Polytechnic Institute, *IEEE Trans. Educ.*, **E-26**, 85–91 (Aug. 1983).
11. N. R. Malik, A Service Course on Microcomputers, *IEEE Trans. Educ.*, **E-27**, 6–13 (Feb. 1984).
12. F. DeCesare, S. M. Bunten and P. M. DeRusso, Microcomputers for Data Acquisition—A Laboratory Course for Pre-Engineering Students, *IEEE Trans. Educ.*, **E-28**, 69–75 (May 1985).
13. Y. Kim and T. Alexander, A New Project-Oriented Computer Engineering Course in Digital Electronics and Computer Design, *IEEE Trans. Educ.*, **E-29**, 157–165 (Aug. 1986)
14. K. L. Short, E. J. Sarpa and S. D. Shapiro, An Innovative Program of University/Industry Cooperation in Microprocessor Education, *IEEE Trans. Educ.*, **E-29**, 111–114 (May 1986).
15. R. Bishop, *Basic Microprocessors and the 6800*, Hayden (1985).
16. A. C. Staugaard, Jr., How to Program and Interface the 6800, Howard W. Sams (1985).
17. A. C. Staugaard, Jr., 6801, 68701 and 6803 Microcomputer Programming and Interfacing, H. W. Sams (1980).
18. J. B. Peatman, Design with Microcontrollers, McGraw-Hill, New York (1988).
19. Motorola, HCMOS Single-Chip Microcontroller, Motorola Semiconductor Technical Data (1988).
20. Motorola, M68HC11EVB Evaluation Board, User's Manual (1986).
21. Motorola, M68HC11 HCMOS Single-Chip Microcomputer, Programmer's Reference Manual (1986).