Session 2559

Teaching and Curriculum Development of Microprocessor Classes

Roman Stemprok
University of North Texas

Abstract

This paper addresses teaching and curriculum development for several microprocessor classes in the Engineering Technology Department at the University of North Texas. Fundamentals of computer hardware and assembly language were presented in undergraduate and graduate courses with emphasis on a processor to control external devices. Students studied microprocessor structure, became proficient in assembly language programming techniques, developed basic microprocessor interfacing techniques, designed simple memory systems, and investigated basic data communications. Special care was taken in organizing labs for these hands-on undergraduate and graduate courses. Students were assigned projects of increasing complexity from a simple control circuit to “Digital Pet” powered by the Motorola microprocessor (a semester project). Successful student teams demonstrated working hardware models at the end of each semester.

Introduction

This paper describes projects and laboratory assignments for courses in the Electronics Division of the Engineering Technology Department. After completion of the digital logic introductory course students learned to utilize microcontroller technology through “hands-on” assignments. Class curricula integrated the Motorola 68HC11 microcontroler and the xx86 Intel processor, which are available on PCs. Courses involved included Introduction to Microprocessors, Digital Systems, and Embedded Controllers.

The MC68HC11A8 microcontroller was used in a variety of lab assignments and projects at the undergraduate and graduate levels to interface output commands from the microcontroller to relays, motors, displays and ICs. The Motorola microcontroller offers high-speed in control-related embedded operations. The 8K bytes of on-chip memory (ROM), 256 bytes of RAM, 512 bytes of EEPROM and an 8-bit analog-to-digital converter were used with several I/O ports [1]. Students designed process controls using, for example, the A/D converter input from a variety of sensors or transducers.

System I/O design issues were taught in the upper division course where a PC interface, using an ISA expansion slot, was built by each student team. The Intel xx86 processor was programmed using assembly code (MASM compiler), or the DOS Debug program. The wire-wrap PC bus extender accessed the signals through the ISA expansion slot. This PC card extender was used to interface devices such as converters, sensors, motors and actuators.

Laboratory Assignments using MC68HC11

Students were introduced to microcontoller technology in the lower division undergraduate course utilizing the Motorola EVB (Evaluation Board) with the MC68HC11A8 processor [2,3]. Students studied Motorola assembly language programming and connecting I/Os. The basic evaluation board (MC68HC11EVB) was connected to a PC through a communication software (Procomm) program. The basic laboratory sequence is shown below [3]:

1. Block move.
2. Parallel output an 8-bit value that will turn ON/OFF LEDs.
3. Input a binary code through the Port C and output the code to 8-bit LEDs.
4. Attach the A/D “on-board” converter with a temperature sensor through an interface.
5. Output compare the square wave generation.
6. Measure a signal frequency utilizing the "on-board" programmable timer..
7. Generate sound using a speaker attached through an interface to a port of HC11.
8. Incorporate EEPROM programming.
9. Initiate event counting using a pulse accumulator.

Project Applications using MC68HC11

The graduate course included three semester projects constructed by each student. Students in the lower division undergraduate course were assigned end-of-semester projects using the HC11 microcontroller. The EVB, with a 60-pin header, was used to interface the required sensors, actuators and other I/Os through conditioning circuits. Appendix I includes a programming sample of an EEPROM that was used for these projects. Project applications included:

- Collision Alert System Project
- Gasoline Pumping Station
- Key Card Reader
- Digital Pet or Moving Roach
- Temperature Room Control
- Traffic Lights Control System

Figure 1 shows the Collision Alert System project. Recent studies indicate that most traffic accidents are a result of poor driving habits. Using an on-board microcontroller (M68HC11), accidents can be avoided by determining distance and speed. This project includes sensors connected to the car’s accelerator, brake, and speedometer. The indicator light is lit when these sensors are activated and the traffic situation calls for a control action (accelerating, breaking and measuring speed.) The project included recommendations to avoid hazardous situations.

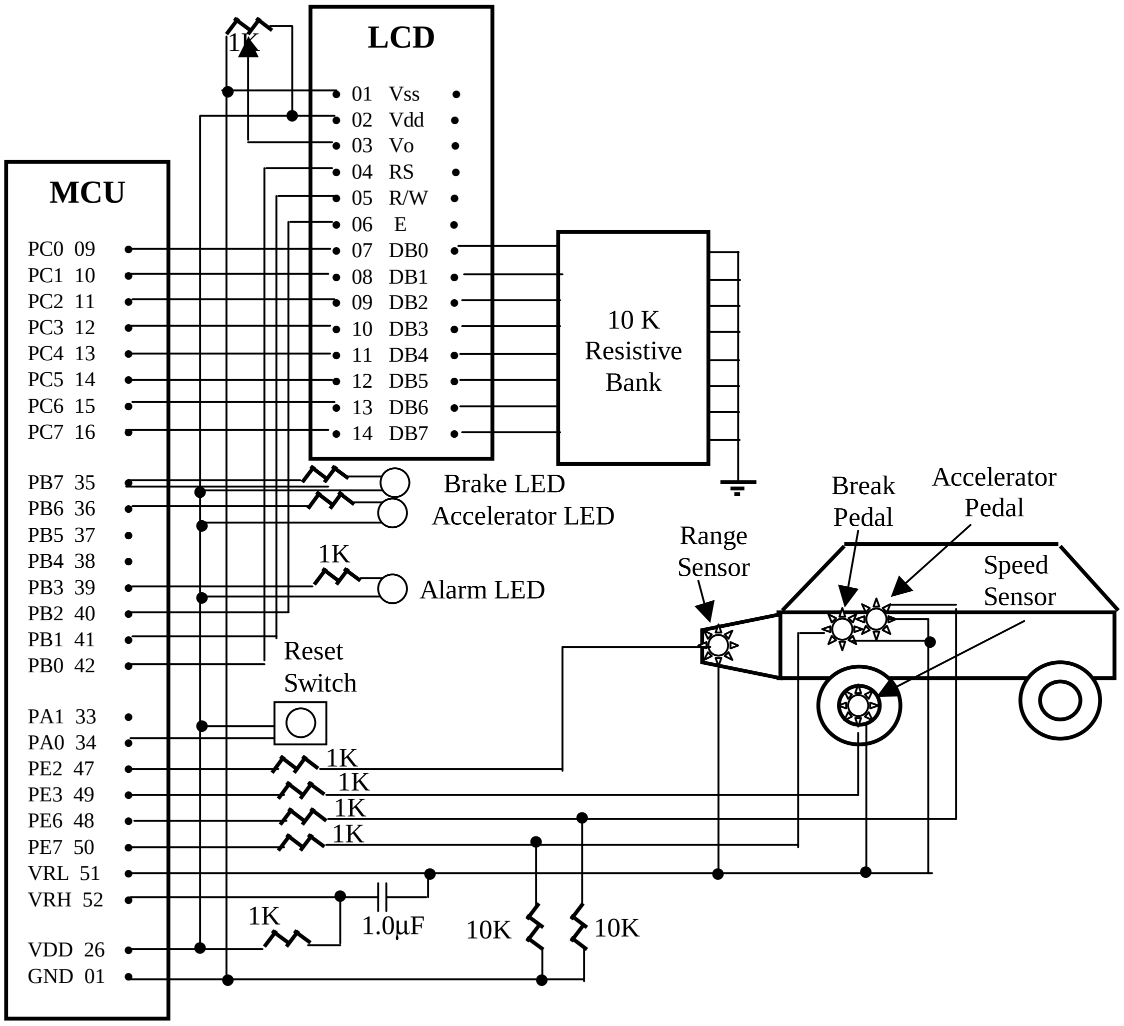

Figure 1: Car Control Peripherals and MCU

Digital pet was a successful design completed by several independent student groups. The animal body was constructed from LEGO parts with actuators powered by servomotors. The pet had “eyes” (sensors), its tail moved, and the animal emitted a sound when pulled. Some designs were able to navigate around objects.

A room temperature control project utilized the D/A converter included in the HC11 microcontroller. Outputs controlled the blower, air-conditioner and heater.

Laboratory Assignments using x86 Intel

Modified laboratory assignments, listed in Mazidi's lab manual [4], were given to students in the upper level undergraduate course to learn the general parts. They wrote assembly codes in Debug, performed data area calculations, and explored the BIOS functions of an Intel manufactured processor involving parallel and serial ports. The laboratory sequence, each including several lab sessions, is shown below [5]:

1. Assembly language programming using MASM and Debug
2. Assembly language application and program computations
3. PC system programming such as the mouse, video and memory
4. Serial and parallel port interfacing.

The next level of assignments was to connect the hardware I/Os to a computer. A test board [6] was connected through an ISA PC slot at 300H – 31FH address. Lab assignments included the following:
1. Output - Intel 8255 to control LEDs
2. Output - Intel 8255 to control stepping motor
3. Input/output from keypad to a monitor
4. Output – Intel 8253 to play music

Project Assignments using x86 Intel

An interface card using a ISA PC bus was constructed by each two-member student team. An empty JDR prototype card was supplied to each group [6]. The 16-bit prototyping board, with I/O decode logic, is for students who like to design their own cards. This card has plated through holes, silk screen legends, and gold plated edge card connectors and is compatible with soldertail or wire-wrap sockets. The examples use conventional addresses in the hex 300 through 31F range, and are expected to be 16 bits wide.

The card was stuffed with buffers to access the bus signals. Figure 2 is a design of such an ISA card interface with I/Os. Students used the 8255 Programmable Peripheral Interfacing chip. When a dynamic change was not required on the JDR card, a hardwired 74LS373 and 74LS244 chips were used.

The projects were:
1. Automatic Guitar Tuner
2. Constant Tire Pressure Regulator
3. Music Player

Conclusions

The laboratory assignments, leading to semester projects using the computer hardware and assembly language, were presented. Emphasis was on a processor that would control external devices. Students studied microprocessor structure, became proficient in assembly language programming techniques, developed basic microprocessor interfacing techniques, designed simple memory systems, and investigated basic data communications. Special care was taken in developing labs for undergraduate and graduate level courses. Students enjoyed these “hands-on” courses and were assigned the building of projects of increasing complexity from a simple control circuit to a

“Digital Pet” semester project powered by the Motorola microprocessor. Successful student teams demonstrated their working hardware models at the end each semester.

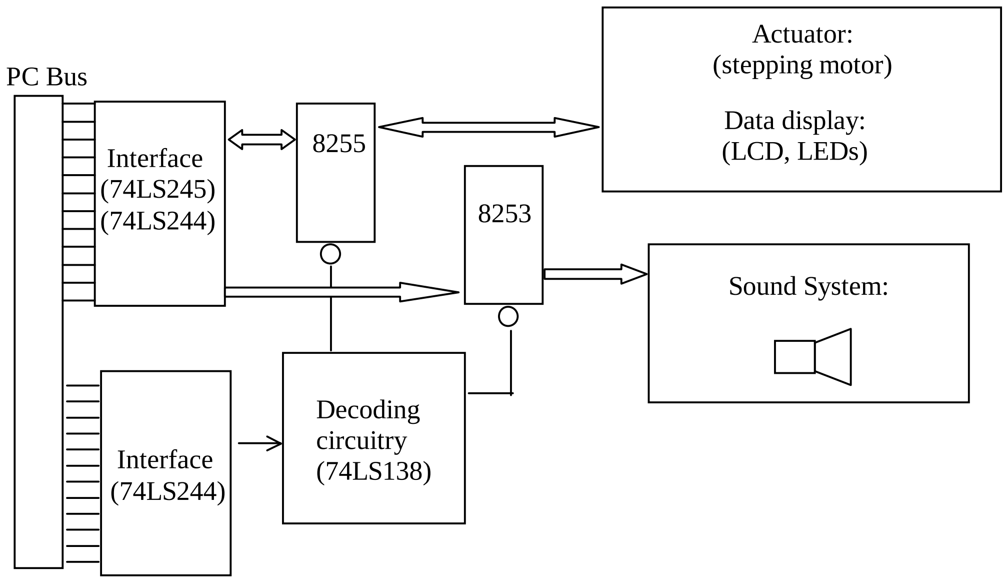

Figure 2: The 80x86 ISA PC Bus Controller

Bibliography
1. Driscoll, Coughlin, Villanucci, “Data Acquisition and Process Control with the M68HC11 Microcontroller,” Prentice Hall, 1994.
2. Motorola, Inc., “M68HC 11 Reference Manual,” 1991.
3. Motorola Inc, “Motorola Technical Training MC68HC11 Course,” 1993.
4. M.A Mazidi, “The 80x86 IBM PC and Compatible Computers,” Volumes I & II, Prentice Hall, 1998.
5. M.A Mazidi, “Lab Manual for The 80x86 IBM PC and Compatible Computers,” Volumes I & II, Prentice Hall, 1998.
6. JDR Microdevices, “16 Bit Prototype Board”, JDR-PR10, user manual, 1998.
7. Jefferey Royer, “Intergface Tutor, Student –user Manual V1.0”, 1989.

Roman Stemprok

Roman Stemprok is an Assistant Professor of Electronics Engineering Technology at the University of North Texas. He also serves as Director of the Center for Transportation Study. Dr. Stemprok is actively involved in research for industry. Dr. Stemprok received a M.E. in Electrical Engineering from McGill University in Montreal in 1989 and a Ph.D. from the Department of Electrical Engineering at Texas Tech University in 1995.

Appendix I

```
*         Example of using MC68HC11 for a control operation
*         Concept of EEPROM programming that was used in various projects
*
PPROG           EQU    $103B              ;index for PPROG register for EEPROM control
                ORG    $C100              ;start of FCC memory away from user code memory
NAME            FCC    'code to burn'     ;this is a code/data of your choice
                FCB    0                  ;form constant byte, fill that with zero
                FCB    0                  ;form constant byte, fill that with zero


                ORG    $C000              ;starting address of the user code
*
*         ROW ERASE MODE – step one
*
                LDAB   #$0E               ;set register $103B to ROW erase ($0E)
                                          ;EELAT, ERASE, ROW = 1
                STAB    PPROG             ;store ACCB at $103B
                LDY     #$B600            ;pointer to start of EEPROM array, EEPROM location is
                                          ;$B600 to $B7FF (512 bytes)
                STAB     0,Y              ;Store 'Don't Care' Data
                LDAB    #$0F              ;turn on program voltage EEPGM = 1, EELAT = 1,
                                          ;ERASE = 1
                STAB    PPROG             ;write the contents of accumulator B to address $103B, turn
                                          ;charge pump on, can not read from EEPROM
                BSR     DELAY             ;make a 10 ms delay
                CLR     PPROG             ;turn-off program voltage, enable READ – turn off latch,
                                          ;ERASE = 0
*
*         PROGRAM MODE, ROW into EEPROM Starting at $B600 – step two
*
                LDX      #NAME            ;load X as pointer to NAME table
                LDY      #$B600           ;load Y as pointer for EEPROM location
PRB             LDAA       0,X            ;load data from table
                BEQ        EXIT           ;exit PRB loop upon detecting 0 in NAME table
                LDAB    #$02              ;set EELAT bit in PPROG register
                STAB      PPROG           ;store ACCB at $103B
                STAA       0,Y            ;store data to LATCH
                LDAB     #$03             ;turn-on EEPGM bit - voltage on, EELAT = 1
                STAB      PPROG           ;store ACCB at $103B
                BSR       DELAY           ;delay about 10 ms
                CLR       PPROG           ;turn-off EEPGM voltage, READ enable
                INY                       ;increment index register Y by one
                INX                       ;increment index register X by one
                BRA       PRB             ;run loop
*
*    SUBROUTINE DELAY about 10 ms = #2860
*
DELAY           LDD      #2860
LOOP            SUBD     #1               ;subtract one from accumulator D inside the loop
                BNE       LOOP
                RTS
*
EXIT            STOP
                END
```