Distance Computer Architecture Laboratory

Saeid Moslehpour, Patrick Keene, Thomas Eppes and Peter Schuyler
University of Hartford

Abstract

Working in a laboratory environment is vital for students to master the technological concepts in science and engineering. Besides re-enforcing what is covered in lecture, lab time allows students to engage in experience-based learning. The educational community largely uses onsite experimentation for electronics/computer engineering laboratory experiments. How can we offer distance laboratory activities in computer engineering technology? The objective of this project was to investigate the use of the Freescale Semiconductor Microcontroller Student Learning Kit (MCUSLK) in combination with National Instruments Educational Laboratory Virtual Instrumentation Suite (NI ELVIS). We used Code Warrior development studio.

Introduction

This paper investigates the Freescale MCUSLK in a distance laboratory setting. The purpose of the project was to understand how to use the Freescale kit and the NI ELVIS system over the Internet. This would lead to a virtual lab environment where students could create, upload and test microcontroller programs remotely.

First, we wrote and simulated computer programs using the Code Warrior development studio. Next, we uploaded the program to the microcontroller and ran it. We then evaluated the results by connecting to the NI ELVIS instrument suite via LabVIEW and used the NI ELVIS system to perform additional debugging.

We also evaluated the effectiveness of the MCUSLK as a learning tool. Much of today's curriculum is found on the web and is distributed on electronic blackboards or from faculty websites. The Freescale MCUSLK and NI ELVIS provide a useful educational environment for students who are not available for onsite laboratory.

Components Evaluated

Freescale Microcontroller Student Learning Kit – The Freescale Microcontroller Student Learning Kit is more than just a microcontroller trainer.[1] It consists of an MCU project board complete with a microcontroller development module and a Motorola MC68HC908QY4P 16-pin dual in-line package (DIP). The kit comes with all necessary hardware including a power supply, serial cable, CodeWarrior Development software and technical training materials.[2]

MCU Project Board - The MCU project board supports several microcontroller development modules. Using a standard MCU port, these modules can plug directly into the board or can be connected by ribbon cable. The project board has useful tools that aid in quick easy prototyping of electronic circuits.

Support tools include on-board LEDs with current limiting resistors as well as push button and slide switches. These devices can be connected via ports on either side of the breadboard. In addition, banana ports and a BNC connector are located on the project board. These ports allow easy connection to outside tools such as a multi-meter, oscilloscope and function generator. These tools make the board useful for all types of electronic prototyping and make it easy to replace any breadboard currently in use.

CSM12C32 Development Module - This module has an RS-232 serial port and onboard power input. It can be used by itself or plugged directly into the MCU port of the MCU Project Board. This development module houses a Motorola MC9S12C32 microcontroller unit with 32 KB of EEPROM and 2 KB of RAM. This module provides an onboard analog to digital converter and supports a serial communications interface (SCI) and a serial peripheral interface (SPI).

Packaged with the development module is CodeWarrior Development Studio from Metrowerks. This software is a development suite designed to interface with the microcontroller. It includes a full microcontroller simulator as well as an interface that enables the user to work directly on the microcontroller. Standard libraries are included for common development modules in C, C++ and assembly languages.

Development Using the Development Module

Before all devices were connected, we followed the instructions to verify all power jumper settings. The project board came with jumpers pre-installed; however, we followed the guide to verify that they were in the proper locations. We next connected the development module into the MCU port of the MCU project board. A ribbon cable was connected at one end to the BDM port and to the BDM OUT port on the project board. The serial cable was connected from a laptop serial port to the COM port of the project board.

Next, we launched the Metrowerks CodeWarrior development software. After installation, the software showed how to setup a new project. The software asked for information on the development module being used, language (C, C++, and Assembly) and other options. This created an IDE specific to the software was being developed.

After this initial run of CodeWarrior, we became familiar with the Integrated Development Environment (IDE) interface. The primary tool that proved to be useful was the debugger window. It provided a view of the source code in both the original language and the compiled assembly language. The debugger also contained a window that showed the contents of the CCR's. A memory map was also present with a display of all memory values in hex form.

Sample Code

Sample code was written for the CSM-12C32 development module in assembly language. Figure 1 shows a simple program that adds two numbers. Adding two numbers was an excellent and simple way to exercise many of the CCR's.

Figure 1. Sample Assembly Code

```
CLRA
LDAA  #01
2  ADDA  #02
3  STAA  $0101
4  LDAA  $0101
```

By running the software in the debug mode, we were able to single step through the assembly instructions. Execution of the first instruction in line 0 caused the Z flag to go high. When the LDAA instruction was executed in line 1, we saw the contents of the A accumulator change from $00 to $01. When we single stepped the instruction at line 2, we observed the value $02 was added to the contents of accumulator A with a result of $03. A similar program was written in C++. Because C++ is a high-level language it needed to be compiled into assembly code for the MCU. Figure 2 shows the C++ code and the compiled assembly instructions generated by CodeWarrior.

Figure 2. C++ Program and the Compiler Assembly Instructions

C++ Code	**Assembly Instructions**
0 int a=2; 1 for(int x=5;x>0;x--) 2 {a++;}	LDX #5 INY DEX CPX #0

NI ELVIS

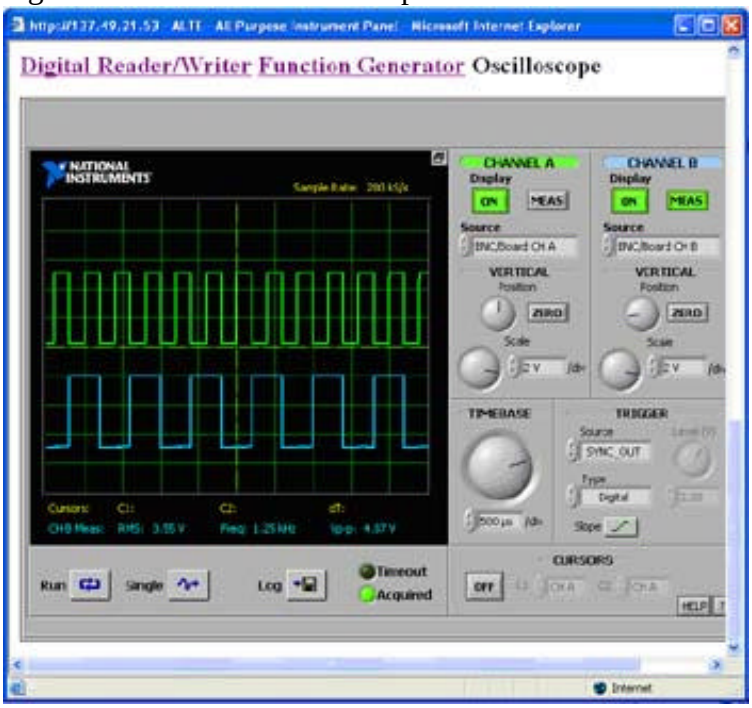
NI ELVIS is an integrated hardware and software platform that makes it easy to construct a test circuit, connect it to onboard instruments and interface those instruments remotely via LabVIEW. Each NI ELVIS unit comes with a number of internal instruments (e.g. multimeter, function generator, oscilloscope, bode plotter, digital reader, digital writer).

Setting up an NI ELVIS lab station to support a distance lab was a relatively easy process. After the test circuit was built in the breadboard area, the desired signal inputs and measurement points were connected. The connections points were well labeled so the test circuit can be wired on a lab bench then mounted to the NI ELVIS base unit. Lastly, the NI ELVIS virtual instruments needed to perform the lab were incorporated into a web page to be served by LabVIEW. Figure 3 is a picture of the complete NI ELVIS unit with a test experiments built on the breadboard.

Figure 3. NI ELVIS Unit

NI ELVIS comes with an interface cable to a VISA data acquisition card that must be installed in a PC running a development version of LabVIEW. NI ELVIS units are supplied with a suite of virtual instrument panels (VIPs) that must be loaded into LabVIEW. To simplify the user interface, we integrated all of the VIPs onto one browser window. Using bookmarks, we were able to switch from one instrument to another. Figure 4 shows a screen shot of the oscilloscope VIP.

Figure 4. NI ELVIS Oscilloscope Virtual Instrument Panel



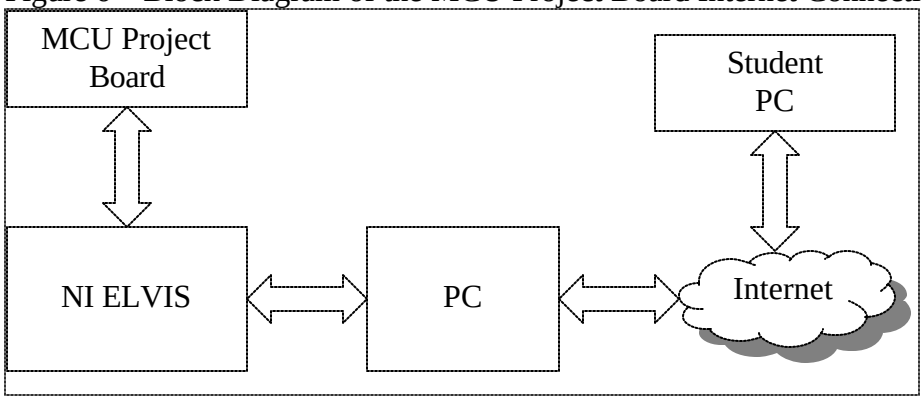
The NI ELVIS unit was located in a network closet as shown in Figure 5. The NI ELVIS unit was connected to a dedicated PC running LabVIEW 7.1 that served web pages containing the embedded VIPs.

Figure 5. Photograph of the NI ELVIS Unit and Lab Station PC

Learning Remotely Over the Internet

Figure 6 shows a block diagram of the MCU project board with its remote connection to the Internet. The MCU Project board, mounted on the NI ELVIS unit, was connected to the internal virtual instruments. Using the VIPs, we were able to access the MCU Project board via a simple web browser.

Figure 6 – Block Diagram of the MCU Project Board Internet Connection



Benefits and Limitations

The Freescale MCUSLK uses the latest technology and is conveniently packaged for students. The kit closely resembles what is used in industry making the transition an easy one for students.

The kit has many troubleshooting tools not offered in other MCU trainers. Connections between the board and external components are simple with standard input and output connectors for most devices.

The major limitation is not being able to upload software to the MCU over a remote connection. The software must be loaded locally by connecting directly to the PC's serial port. Without this ability, it can't be used to perform true distance experiments. Although there is no built-in feature that supports software uploading, one could develop this capability.

LabVIEW is a very comprehensive and versatile software package with the ability to support distance labs via the Internet. So, it is possible to program LabVIEW to send the necessary BDM commands to the MCU to upload pre-compiled instructions.

Procedures and Implementation

The MCU can be adopted for courses such as Microprocessors and Computer Architecture; however, this would be made easier if more training materials were available. Any implementation with NI ELVIS is limited by the onboard instrument suite, e.g. oscilloscope, function generator, digital multi-meter, digital reader/writer and LEDs. This is not sufficient for most microprocessor courses.

Conclusion

NI ELVIS is being deployed by some higher-education institutions in their first and second year electronics laboratories. It provides an inexpensive workstation for students compared with the cost of outfitting a laboratory with traditional test and measurement equipment. These savings allow a school to invest in more equipment for third and fourth years.

There are alternatives for laboratory experiments, most experiments on Microprocessor's Heathkits Educational Systems books could be adopted for the Freescale MCU. Some other experiments could be converted to Freescale MCU as well.

The Freescale MCU Project board does provide microprocessor functionality to NI ELVIS. However, the training material does not clearly present the fundamentals of microprocessors or how to interface them with external circuits. Therefore, it is best suited for third and fourth year students who have already mastered these fundamentals. In these cases, it does represent a useful tool to teach current microprocessor technology.

This experiment involved one senior graduation student over one semester. The laboratory equipment performed fairly well. The main concern with the student was lack of supporting examples and programs of diverse features of those boards. In this study we did not make any comparison among the off campus and on campus students.

There is a lack of training material on how the MCUSLK works. Although many white papers have been published, they are not effective learning tools for most students. This burden falls on the instructor; a time consuming task that may discourage widespread adoption. Freescale

indicated that they intend to publish a collection of training material produced by instructors. These materials would be made available to any for use in their courses. Provided that additional materials are developed, the investment on these boards is worth the investment.

Bibliography

1. Montañez, E & Ruggles, S. "Getting Started with the Microcontroller Student Learning Kit (MCUSLK)", *Freescale Semiconductor, Inc*., August 2004.
2. "MCU Project Board, Prototyping Board with Microcontroller Interface", *Axiom Manufacturing*, Garland, TX, June 2004.
3. "Motorola Semiconductor Technical Data, CPU12 Reference Guide", *Motorola Inc.*, 1998.
4. "MCU Project Board Quick Start Guide", *Axiom Manufacturing*, Garland, TX, April 2004.
5. "CSM-12C32 Development Module for Motorola MC9S12C32", *Axiom Manufacturing*, Garland, TX, April 2004.
6. Schematic. "MCU Project Board AXM-0331", *Axiom Manufacturing*, April 2004.
7. Gardner, G. "Schematic – CSM-12C32", *Axiom Manufacturing*, March 2004.
8. "MC68HC912BC32 – Technical Summary, 16-Bit Microcontroller", *Motorola Inc.*, 1997.
9. Website of Jonathan Valvano. "Programming the Tech Arts 6812 boards", *http://www.ece.utexas.edu/~valvano/metrowerks/*, October 2004.
10. Website of Freescale Semiconductor at *http://www.freescale.com*.
11. Website of CodeWarrior, "CodeWarrior for Freescale HC12", http://www.metrowerks.com/MW/Develop/Embedded/HC12/Default.htm.
12. Valvano, J., "Programming the Tech Arts 6812 boards", www.ece.utexas.edu/~valvano/metrowerks/, October 2004.
13. www.freescale.com/webapp/sps/site/overview.jsp?nodeId=06X0FzYcTY6167, Freescale reference material.

Biographies

PATRICK KEENE was undergraduate student at the University of Hartford majoring in Computer Engineering Technology. His areas of interest are software engineering and microprocessors He graduated on fall of 2004.

SAEID MOSLEHPOUR is an Assistant Professor in the Electronic and Computer Engineering Technology Department in the College of Engineering, Technology, and Architecture at the University of Hartford. He holds PhD from Iowa State University and BS MS and EdSP degrees from Central Missouri State University. His area of interest is logic design and FPGAs.

THOMAS EPPES is an Assistant Professor and Chair of the Electronic and Computer Engineering Technology Department in the College of Engineering, Technology, and Architecture at the University of Hartford. He holds BSEE and MSEE degrees from Texas A&M University and a PhD in ECE from the University of Michigan. His area of interest is distance experimentation and intelligent interactive software agents.

PETER SCHUYLER is an Assistant Professor in the Electronic and Computer Engineering Technology Department in the College of Engineering, Technology, and Architecture at the University of Hartford. He holds a BS degree in Bioengineering from the Syracuse University, an MSEE degree from the Syracuse University and is a doctoral candidate in Higher Education Administration at the University of Massachusetts.