AC 2008-1156: INNOVATIVE LAB STATION USING THE FREESCALE 'HCS12 MICROCONTROLLER AND DRAGON DEVELOPMENT BOARD

Christopher Carroll, University of Minnesota-Duluth

Christopher R. Carroll earned his academic degrees from Georgia Tech and from Caltech. He is Associate Professor and Assistant Head of Electrical and Computer Engineering at the University of Minnesota Duluth. His interests include special-purpose digital systems, VLSI, and microprocessor applications, especially in educational environments.

Innovative Lab Station Using the Freescale ‘HCS12 Microcontroller and Dragon Development Board

(author information omitted for review)

Abstract

The Freescale ‘HCS12 sixteen-bit microcontroller is a popular vehicle for teaching introductory microcomputer system design. Freescale’s Dragon development board is often the foundation for lab station implementations. However, the Dragon board usually is used with a dedicated personal computer at each station, which is expensive and which leads to a cluttered lab environment. Individual personal computers are unnecessary, and are overkill for the needs of the lab. What *is* required is a *central* computing facility on which students can create, edit, and assemble assembly language programs, and from which students can download the resulting object files into the Dragon boards at their stations. This paper details a lab environment that uses a *single* personal computer running the multi-user LINUX operating system, serving as a host for *all* lab stations in the microcomputer design lab. This environment removes the clutter and expense involved with personal computers at every station, but requires some enhancements to the Dragon board to allow the lab stations to function as independent terminals on the LINUX system.

To work as independent stations on the LINUX system, each station must include an alphanumeric keyboard and an alphanumeric display to provide capability to interact with the host computer as a terminal for creating, editing, and assembling source code. The design of the lab station presented here includes an alphanumeric matrix keyboard, scanned using the n-key rollover algorithm through two of the eight-bit input/output ports on the Dragon board. The lab station design also includes an alphanumeric display, showing thirteen rows of thirty-two characters each on a standard, low-cost television monitor. The composite video for the television display is generated using a Serial Peripheral Interface (SPI) on the ‘HCS12 microcontroller, and requires only a few external passive components added to the Dragon board.

This paper presents the design of this innovative ‘HCS12 Dragon-based lab station, and discusses some applications in the lab environment.

Background

The Freescale ‘HCS12 sixteen-bit microcontroller is a descendent of the very popular Motorola MC68HC11 8-bit microcontroller. The instruction set of the ‘HCS12 is a full superset of the instruction set of the MC68HC11, so all programming experience that users have with the MC68HC11 is directly applicable for writing programs on the ‘HCS12. However, memory addresses and peripheral device addresses are different in the ‘HCS12 processor, and in some cases, the internal peripheral devices operate slightly differently from those in the predecessor processor, so some effort must be employed to adapt to the ‘HCS12. However, the ‘HCS12 offers significant enhancement of capabilities through an expanded and more powerful instruction set, new addressing modes for accessing memory operands, and improved and expanded input/output capabilities in the internal peripheral devices included on the microcontroller. This author has considerable experience using the older MC68HC11 microcontroller and designing educational lab stations around it.[1,6,7] He admittedly had some reservations about abandoning the tried and true processor to switch to the ‘HCS12 as the vehicle for an introductory microcontroller course. After only a little experience with the ‘HCS12, however, he now is a strong proponent of using the ‘HCS12 in educational environments, and has documented earlier work with Motorola’s predecessor to the Freescale ‘HCS12 processor, the MC68HC12.[2,3,4,5]

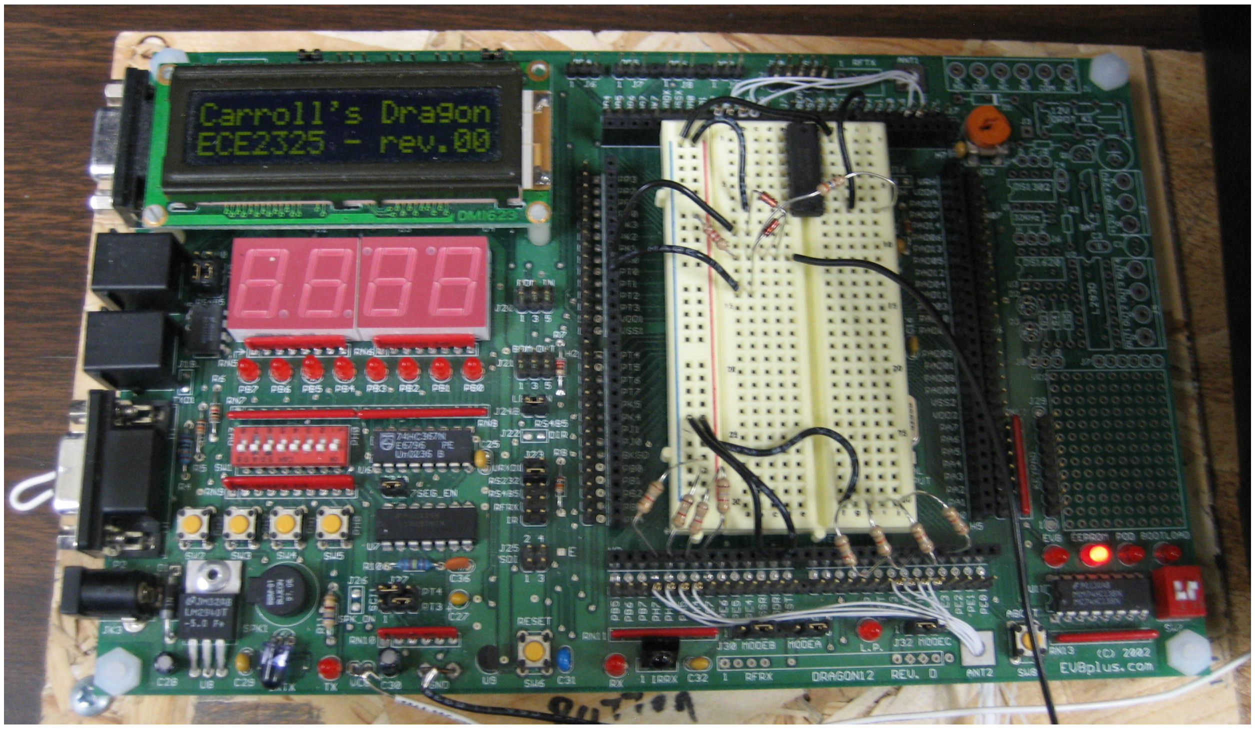


Figure 1. The Freescale Dragon development board as used in the lab station

The lab stations to be described here implement a microprocessor lab in which students learn to write programs in assembly language and to interface external devices through the input/output capabilities of the processor. Each station consists of a commercial Dragon development board from Freescale, shown in Figure 1, which has been enhanced through the addition of hardware and software to be described. Each station interfaces to a single lab host personal computer over a standard terminal line. The host computer runs the multi-user LINUX operating system, and provides file storage, editing, assembling, and downloading capabilities for the lab stations. Figure 2 shows the lab station embellished with keyboard and alphanumeric display to be discussed in this paper.

The goal of the work described here was to make the lab stations based on the Freescale Dragon boards free-standing, autonomous stations that do not need a dedicated personal computer for each station in the way that the Dragon boards are intended to be used. By adding an alphanumeric keyboard and display to the boards, and linking them over terminal lines to a multi-user single host computer, this goal was achieved.

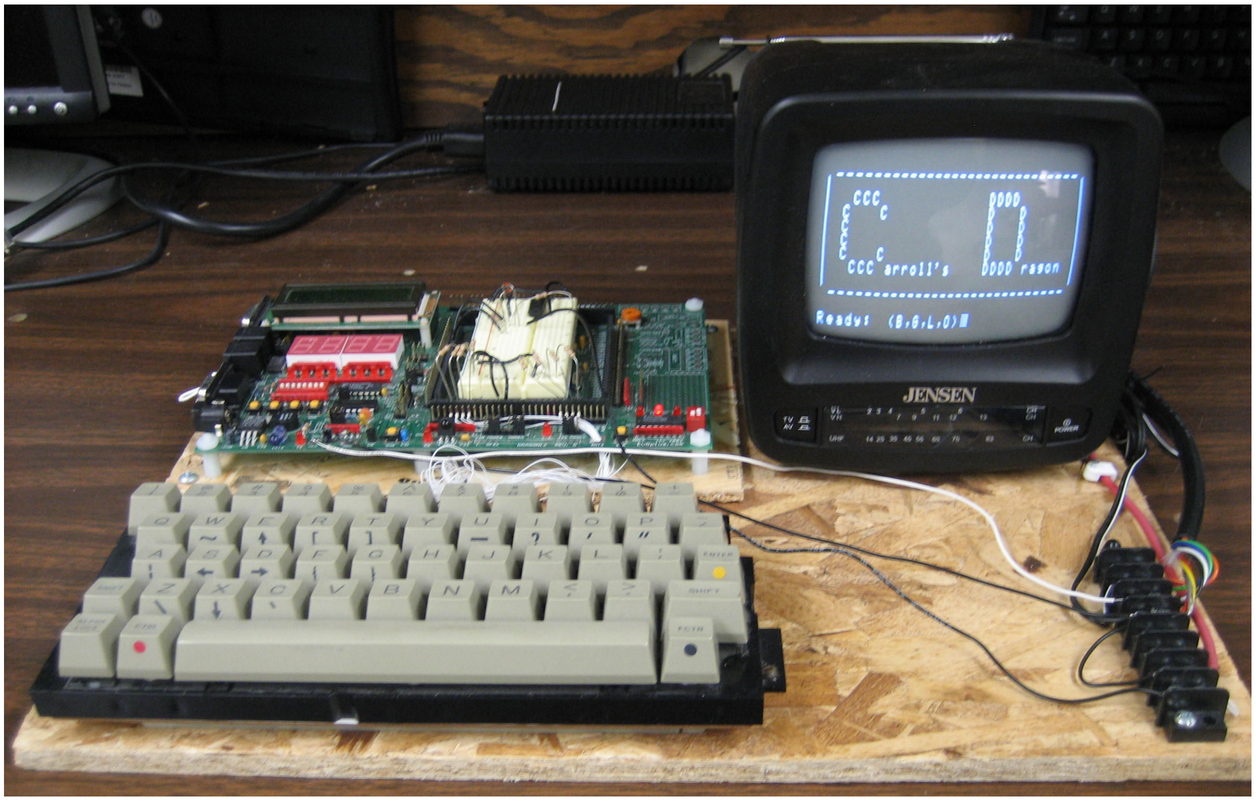


Figure 2: Lab station using the Freescale Dragon board, plus keyboard and display.

Dragon Board Hardware Enhancements

The ‘HCS12 microcontroller on the Dragon board provides a wide variety of input/output features that can be used in all sorts of lab experiments. However, in order to create a stand-alone lab station based on the Dragon board, alphanumeric input and output were the first necessity. Fortunately, adding those features required almost no additional hardware other than the keyboard and the display itself, due to features present in the ‘HCS12 internal peripherals.

The alphanumeric keyboard for the lab station, shown in Figure 2, consists of just switches in a matrix arrangement. Six row wires and eight column wires form the matrix, and a switch at each row/column intersection connects the row and column wires at that point when pushed. This “matrix keyboard” is the standard way to connect arrays of switches to microcontrollers. The column wires have pull-up resistors connected to them so that they default to logic 1 when nothing else pulls them down to ground. The eight column wires are read through one of the eight-bit input ports on the ‘HCS12. A second port, configured as an open-collector output port, drives the rows of the keyboard matrix. By driving a single logic 0 on one row of the matrix at a time and reading the column port, the keyboard matrix is “scanned” to detect key pushes. This process requires some software overhead to scan the rows of the keyboard at the proper rate, and to debounce the switch closures and openings, but that process requires no additional hardware, and is straightforward to implement in software. Scanning the keyboard in software provides a challenging lab exercise for students to help them understand the uses and capabilities of the standard 8-bit input/output ports on the ‘HCS12 microcontroller. This keyboard interface implements the required alphanumeric input capability for the lab station.

Providing the alphanumeric display to complete the needs of the stand-alone lab station required more creativity and innovation. To minimize costs, the display device itself is a cheap ($30) standard small television that includes a composite video input jack. The challenge is to generate the composite video signal using the resources of the 'HCS12 and minimal additional hardware.

The timing signals that form part of the composite video are easy to generate. The timer system in the 'HCS12, like that in the earlier MC68HC11 microcontroller, is very easy to use and very flexible. Two timing synchronization pulses, the "vertical sync" and the "horizontal sync," are generated on two of the eight timer outputs using the "output compare" feature of the timing system on the microcontroller. The vertical sync signal produces a low-going pulse at a rate of 60 Hz to establish the frame rate for the image generated on the television. The horizontal sync signal produces a low-going pulse at a rate of 15,360 Hz, 256 times the rate of the vertical sync. Each horizontal sync pulse causes one scanline to be produced on the display, and 256 scanlines (minus 20% or so of them that are wasted during vertical retrace time) together form the raster that produces the display on the television. The vertical and horizontal sync signals are combined through a Diode Logic AND gate to form the timing part of the composite video, as shown in Figure 3. Resistor values are chosen to provide appropriate contrast and sync levels.

The video portion of the composite video signal is the more challenging part. The video controls the electron beam on the display as it sweeps across the television screen on each scanline in the raster. In order to obtain useful resolution for producing images on the screen, the video must have a relatively high bandwidth. Somehow, bits must emerge from the 'HCS12 fast enough to provide the required bandwidth to turn on and turn off the electron beam on the television display fast enough to provide useful resolution in the image. Fortunately, the 'HCS12 provides an output device called the "Serial Peripheral Interface" or SPI that can spit bits out at rates up to twelve megabits per second. This is adequate for the alphanumeric display requirements.

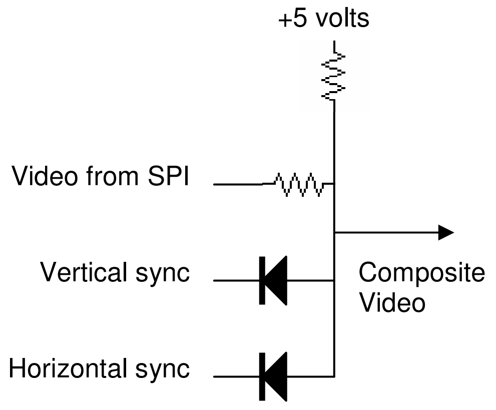
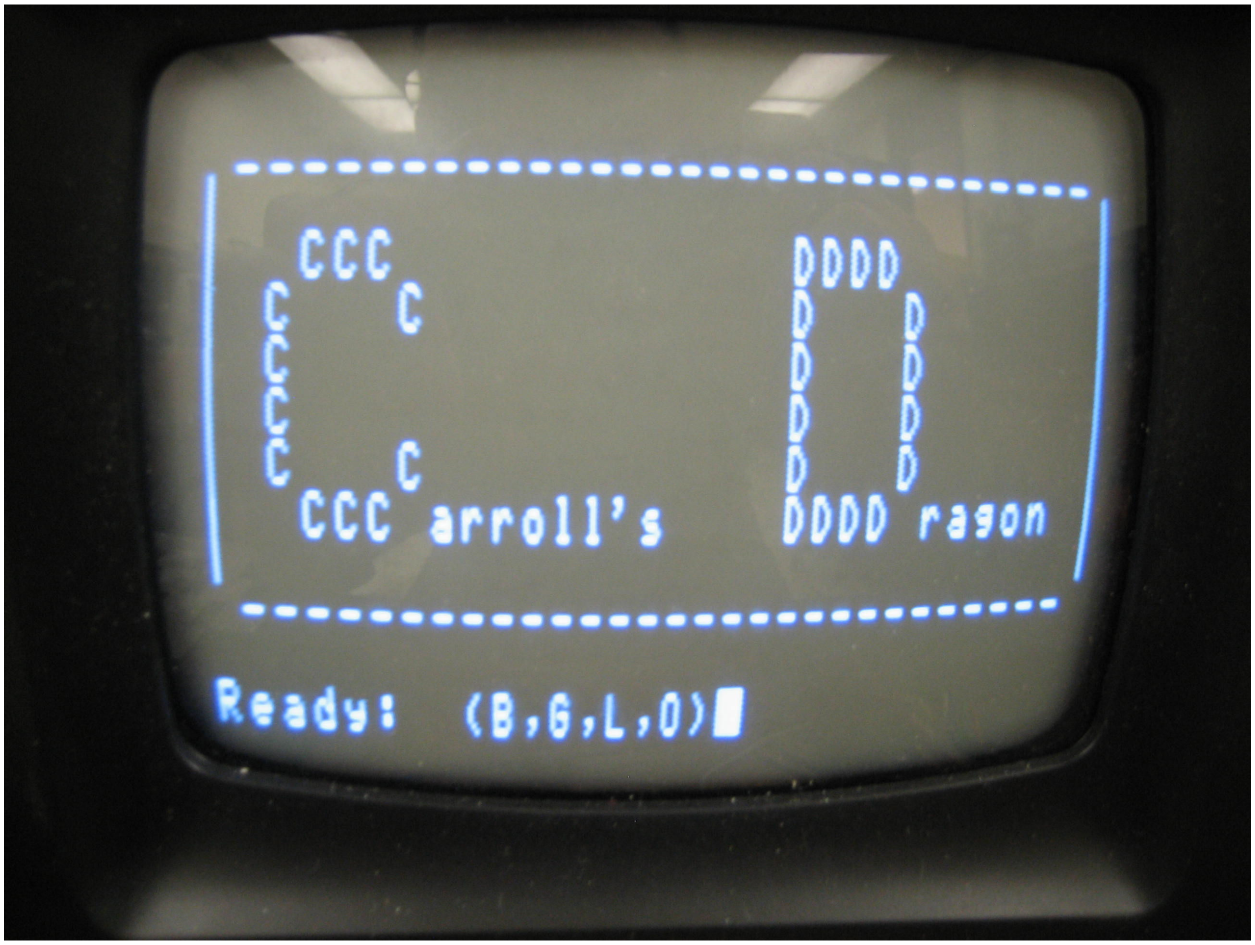


Figure 3. Composite video generation

The alphanumeric display draws characters using an eight-by-eight array of pixels for each character. With 256 scanlines (minus 20% to allow for vertical retrace time and screen overscan) that could conceivably support up to 26 rows of characters. Each scanline lasts for 1/15,360 of a second, or about 65 microseconds. Some time is needed for the horizontal retrace, so only about 50 microseconds are available per scanline to produce an image. With pixels emerging from the 'HCS12 at a rate of 12 million bits per second, conceivably each row of the display could contain up to 75 or so characters. Practically speaking, the bandwidth of the video circuits in the cheap television limits the rate at which the video can switch, and the small size of the television screen makes the small characters hard to read anyway. A design decision was made to implement a display with 13 rows of 32 characters each. Scanlines were used in pairs to produce the necessary 104 pixels of vertical resolution, and the SPI was programmed to emit bits at a rate of only 6 million bits per second, to match the need of 256 pixels per scanline resolution. The resulting display is shown in Figure 4, and provides a good compromise that acknowledges the low display quality and the small screen size while still providing adequate text on the screen for program editing and development. Figure 4 shows a display with white characters on a dark background, which required a logical inversion of the SPI output, but which produced a more easily photographed image. The actual lab station display uses black characters on a white background, and avoids the need to invert the SPI signal.

The video output produced from the SPI is mixed with the sync signals from the timing system to produce the overall composite video using a simple resistor divider network. (Figure 3.) The resulting composite video connects to the video input of the television monitor. That is the extent of the additional hardware added to the Dragon board to implement this lab station.



```
 ---------------------------
|                           |
|  CCC            DDDD      |
| C   C           D   D     |
| C               D   D     |
| C               D   D     |
| C               D   D     |
| C   C           D   D     |
|  CCC arroll's   DDDD ragon|
|                           |
 ---------------------------

Ready:  (B,G,L,O)█
```

Figure 4. Television display produced on the lab station by the composite video signal.

Dragon Board Software Enhancements

The Dragon development board comes with significant software to support program development and testing. That software is meant to interact with a personal computer dedicated to the lab station, so that each lab station requires its own PC. The goal of this work was to replace the need for a PC at each station with a single host computer running a multiuser operating system (LINUX) to support the various lab stations over terminal lines. That different environment, and the need to interact with the alphanumeric keyboard and display on the lab stations, required that the software that came with the Dragon board be completely replaced.

The new lab station software was written in ‘HCS12 assembly language and loaded into the 3K bytes of EEPROM included in the ‘HCS12 microcontroller. The Dragon board was then configured to jump to this EEPROM upon reset and execute the code there rather than execute the software supplied with the board. In this way, the original software was left unchanged, and can still be used by changing the operating mode of the Dragon board. This is useful for developing the lab station software itself, and occasionally for other maintenance purposes. Happily, the Dragon board’s processor supports two separate asynchronous serial ports, Serial Communications Interfaces (SCIs) 0 and 1. SCI0 is used to communicate with a local PC in case of a need to use the commercial software for station development. SCI1 is used to communicate with the host LINUX multiuser computer for normal lab operation.

Squeezing system software for the lab station into 3K bytes of EEPROM was a challenge, especially considering the fact that several large lookup tables needed to be included such as character generator tables to produce character images on the alphanumeric display, and keycode-to-ASCII tables to report alphanumeric keyboard entries in a standardized way. In fact, some planned features of the software had to be omitted in order to fit in the 3K bytes of space available, but the resulting software does allow student users to access and edit files on the host computer, assemble their programs, download the resulting object code from the host, and run their programs on the lab station, which is all that is necessary. Some bells and whistles were sacrificed in order to keep the software size down, but without sacrificing function.

The system software provides several useful subroutines that can be called from user programs. These subroutines simplify the job of collecting keystrokes from the keyboard, and putting characters out to the display. Several handy subroutines print bytes or words as strings of hexadecimal characters, or print prompts or other messages to the screen in easily managed ways. The subroutines are not introduced to students until they have practiced performing the various tasks themselves in lab exercises, so that nothing is hidden from students behind the veil of a “magic” subroutine. The provided routines simply give handy access to the keyboard and display input/output so that the students can focus more precisely on the job at hand.

For alphanumeric input, one system subroutine scans the matrix keyboard as described earlier to detect and report key presses. The algorithm used is the “n-key rollover” algorithm that reports key presses whenever a key is depressed, regardless of how many other keys may already be pressed. Details such as “phantom key” detection (when three keys in a right-angle relationship in the key matrix are pressed together) are discussed and demonstrated, so that students gain a complete understanding of the interface to the keyboard switches that is provided in the lab station. Several other input subroutines collect hexadecimal bytes or words entered as sequences of key presses, and these additional subroutines aid in developing larger programs.

The overall framework of the lab station system software consists of a command loop that collects single character commands from the user. There are four commands implemented. Typing “B” invokes the Byte command, that allows the user to examine and change bytes in memory. Typing “O” invokes the Online command so that the user can communicate over the terminal line to the multiuser LINUX system for file editing and assembling. Typing “L” invokes the command that downloads an object file from the host computer into the station memory. Finally, the “G” command allows the user to “Go to” a specified address to run the program there. Those four commands provide adequate control over the lab station to allow students to perform the actions they need to accomplish.

Lab Exercises

The ‘HCS12 processor provides a rich environment that can support a wide variety of possible lab exercises. Looking at the photograph of the Dragon board in Figure 1, one can notice various input and output devices that can be used as the basis for student programs.

Most obvious in Figure 1 is a four-digit, seven-segment display that can be used as an output device. This device is a multiplexed output device, meaning that only one digit is actually displayed at a time, and it is the responsibility of the user’s program to cycle the information through the four digits of the display fast enough so that they all appear to be on at once. This is an interesting lab exercise for students. More fundamentally, eight separate LEDs are available for displaying bytes of information, and four push button switches and eight DIP switches can be used to enter information into programs. A liquid crystal display (LCD) showing two lines of sixteen characters each is available, and interfacing to that requires that students manipulate data at the bit level and understand the timing involved. Not easily identified in Figure 1 are a speaker for producing sounds, and infrared transmit and receive devices that permit interaction with infrared remote control units from consumer appliances and other applications.

In addition to these hardware features included on the Dragon board to support interesting lab exercises, all the features of the ‘HCS12 microcontroller are available as well. These include the sophisticated timing features of the chip (output compare, input capture, real time interrupt, pulse accumulator, etc.), serial input/output (serial peripheral interfaces, serial communication interfaces, controller area network interfaces), signal generation and sampling (pulse width modulators, analog to digital input), and general parallel port input/output (for example, detecting key presses on the keyboard). This plethora of input/output features makes the Dragon board and its ‘HCS12 processor ideal for use in an introductory microcontroller laboratory.

This rich environment for lab exercises provided by the ‘HCS12 microcontroller and the Dragon board places it above the capabilities of other processors when choosing a vehicle for teaching introductory microcontroller/microprocessor topics. It clearly outpaces the older Motorola MC68HC11 capabilities, not only due to technology advances that allow higher clock rates but also due to the enhanced instruction set and improved internal peripheral operation. The Texas Instrument’s competitor, the TMS370 family, was a colleague of the MC68HC11 that never really caught on in the marketplace, and no general purpose microcontrollers have emerged from that ancestry. Intel’s 8051 microcontroller has been around since the early days of microcontrollers, and although popular, is nowhere close to the level of capability provided by the ‘HCS12 family from Freescale. The PIC microcontrollers from General Instruments, being Harvard architecture devices, are difficult to use in a lab setting, and do not have the wealth of input/output capabilities demonstrated by the ‘HCS12.

There are some drawbacks associated with using the 'HCS12 in an introductory lab setting. The 'HCS12 does not offer the opportunity to examine high performance structures such as cache memory or virtual memory management, but such topics are not normally included in an introductory course focused on assembly language programming. Most regretfully, it is not easy to experiment with the 'HCS12 microcontrollers themselves because of their packaging. The chip carrier flat packs or surface mount packages that are used for the 'HCS12 can't be simply inserted in breadboards, so users generally are forced to use commercial development boards.

Summary

The lab station described here using Freescale's 'HCS12 microcontroller and Dragon development board was successfully used in an introductory microcontroller course during Fall semester, 2007, and continues to be the core instructional facility in the microcomputer lab here. By replacing the software in the Dragon board with custom software, and by adding alphanumeric input and output via a keyboard and television display, the need for personal computers dedicated to each lab station was eliminated, and replaced with one common personal computer to serve all the stations in the lab. This resulted in a much less cluttered and more easily maintained lab, and demonstrated interesting input/output capabilities of the 'HCS12 processor that were used in the station design. Students seemed satisfied with the lab stations, and were eager to use them to explore the capabilities of the 'HCS12 processor.

References

1. (authors omitted for review), "Comparing the MC68HC11 and the TMS370 as Vehicles for an Introductory Microcontroller Laboratory," *1994 ASEE Annual Conference Proceedings*, Edmonton, Alberta, Canada (1994).

2. (authors omitted for review), "Design Workshop Experience Using the Motorola 68HC12 Microcontroller," *Proceedings of the 1998 North Midwest Section Meeting of ASEE*, Winona, MN (1998).

3. (authors omitted for review), "Fuzzy Logic on the MC68HC12 Microcontroller: A Student Design Workshop," *2000 ASEE Annual Conference Proceedings*, St. Louis, MO (2000).

4. (authors omitted for review), "Intelligent Systems on Motorola's Microcontroller: A Team Design Workshop," *Proceedings of the ICEE-2000*, Taipei, Taiwan (2000).

5. (authors omitted for review), "Fuzzy Logic on Motorola's Microcontroller," *3rd Working Conference on Engineering Education: Engineering Education for the 21st Century*, Sheffield Hallam University, England (2000).

6. (authors omitted for review), "New Life for the MC68HC11 Evaluation Board," *2002 ASEE Annual Conference Proceedings*, Montreal, Canada (2002).

7. (authors omitted for review), "Video Graphics Using the SPI on the MC68HC11 Microcontroller," *2004 ASEE Annual Conference Proceedings*, Salt Lake City, UT (2004).