

**Week 5: Research Report**  
Student: *Craig Harrison*  
Advisor: *Dr. Richard Eason*

The majority of this past week's work has been done at the Allen-Edmonds factory in Lewiston where one of the machines has been shipped. In the last report I state that we were able to turn the motors on but unable to move them. This week has been focused on discovering why we are unable to move the motors as well as implementing a fault-resistant persistent connection from the machine in Lewiston to the University of Maine campus.

Creating the persistent connection was an important goal as it would save three hours of traveling to and from Lewiston each day. To this end we first needed to get port 22 opened in the factory firewall, which we accomplished with a few calls to the IT department. Next we got the electrical and computer engineering department server (hammer) to accept ssh connections from the factory IP address. At this point we had discovered that we needed to open a reverse tunnel out from the machine in the factory to hammer, though which we could then ssh back in. However, opening the connection was not enough, as it would go down multiple times a day as well as whenever the computer was restarted and needed some technical knowledge to bring the tunnel back up. To this end we wrote a bash shell script that starts on boot and will continually work to bring up a new tunnel whenever the old connection ever goes down. Currently we have reached the point where we simply have to call down to the factory and ask someone to hit <Ctrl>-C once or twice a day when the machine goes offline. We have a few ideas as to how we can modify the shell script this week so as to remove this limitation.

Our other work this week has been on discovering why the software runs but is unable to physically move any motors. In order to gain a better understanding of what commands should be sent to the motor, we modified and recompiled the old DOS code to log all motor commands, replies, and digital I/O to a text file and ran it. We then modified our new Linux code to log the same functions and compared the output. It quickly became apparent that something was not functioning as we thought because only the first bytes of our commands were being sent correctly to the DCX card. It turns out that unlike the Borland compiler, which keeps arguments in order on the stack, GCC optimizes its passing of arguments in whatever way it thinks best. After rewriting our dcxcmd() function to take this into account we were pleased to discover that both the working DOS code and the new Linux code logged the same commands as being sent to the DCX card.

We are still unable to control the machine's motors with the Linux code, however, which implies that the commands we send may not be making the full trip to the DCX card. We tested this by sending "set gain" commands to the card and then asking the card what the current gain is. Not surprisingly the gain, and therefore any command, is not being received by the DCX card. It appears that only two of the four bytes that make up a command argument are getting to the motor controller card. This is an important discovery because, once fixed, all of the input and output from our new Linux software will be the same as the working DOS software.

**References:**

*Why FreeDOS*. Retrieved June 17, 2008 from <http://www.freedos.org>

*The Fedora Project*. Retrieved June 12, 2008 from <http://fedoraproject.org/>

*MultiFlex PCI 1000 Series Documentation*. Retrieved June 12, 2008 from <http://www.pmccorp.com/support/mfx1000.php>