

Week 2: Research Report
Student: *Craig Harrison*
Advisor: *Dr. Richard Eason*

The goal of this past week's development has been to connect the new Linux software to the old motor controller cards and control the shoe trimming machine from there. This has involved traveling to the Allen Edmonds factory in Lewiston where one of the machines has been shipped. We are also attempting to get port 22 opened in the company firewall so as we can work on the machine remotely over ssh with the aid of a web cam.

The first piece of work at the factory involved transferring the old hardware over to the newer computer and testing that everything still worked. This hardware was a DCX-PC 100 motor controller card with eight axis attached and a digital IO card for controlling the power to the motors and reading the control buttons. Once attached to the newer 300MHz computer we used bootable floppy disks to boot into DOS and test that the old software still worked, which it did. Our next test was to load Linux and attempt to talk to the cards from there. At first our home-written software worked, but we soon discovered that card would lock up and we would have to remove and reinstall the kernel module. The problem turned out to be that the kernel module was locking itself in order to be more robust, but the locking was not done entirely correctly and was misbehaving under certain circumstances. As we will not be using more than one card or running more than one process that accesses the card at the same time we decided to simply remove the locking, thereby preventing any of the problems related to it from occurring.

Next we tried to run our rewritten SST software under Linux with the cards attached for the first time. Naturally it failed, but we discovered an easily fixed problem with attempting to read from the IO card's ports before a call to `ioperm()`, which is what gives us permission to access the ports under Linux. We also discovered that the replacement functions for sending commands and reading replies from the motor controller card were locking between themselves, which created problems when we had an error and tried to stop the motors before checking for replies from the card. This was easily fixed by removing the `ioctl(SLOCK)` and `ioctl(ULOCK)` commands from the command and reply functions.

The card was still locking up, but we were able to discover that the problem had now moved from the driver to the hardware itself. When the card locked up now nothing but killing power to the computer would bring it back online. The problem was that we were sending incorrect commands to the card. After some searching through the code we figured out that the motor controller command and reply functions were making some assumptions about the byte lengths of "int"s and "long"s that were no longer true under Linux. To fix this we turned to the `sys/types.h` header file, which define fixed length types such as `int16_t` and `int32_t`. Rewriting portions of the code to cast everything to the correct size was a usable fix to this problem.

Currently our code is working fairly well up to the point where motors are supposed to move, at which point it times out while waiting for the stylus motor to go home. This is a logical thing for it to do, as the stylus is not moving. Our next goal is to figure out why we can turn the motors on but they will not move.

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>