**Building a VXI LLRF Front-end from Scratch**

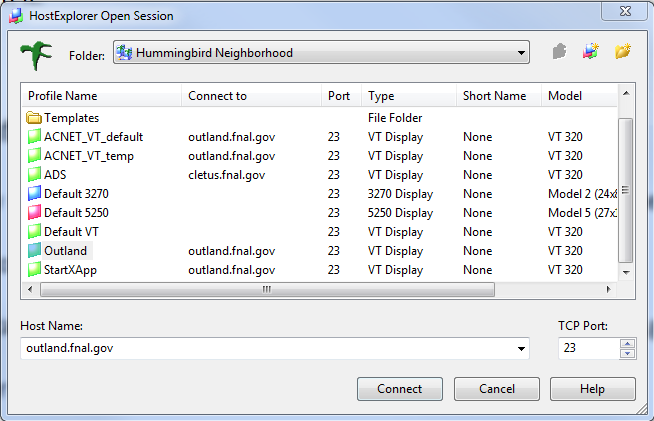
Ed Cullerton

3/5/2012

This procedure outlines how to create a new LLRF front-end from your local Windows machine. The procedure assumes you have a kerberized Windows machine with Hummingbird Connectivity installed on it.

**1. Setup Accounts**

First get an outland.fnal.gov account and a nova.fnal.gov account if you do not have these accounts already. Contact Tim Zingleman to get accounts on those machines. Log into outland using *Hummingbird Host Explorer*. You can copy one of the existing sessions (ACNET\_VT\_default) and rename it “Outland”. Right click on the session to edit the properties, and under Connection/Telnet, make sure the host name is outland.fnal.gov.



Type the following, after logged into outland, to log into nova:

*ssh –X nova.fnal.gov*

Now you should be logged into nova. The set of software tools called “RFIES software tools” are needed to build a LLRF front-end. Configure your nova account with the RFIES software tools [1][2] needed to build a frontend by typing: (include space after ‘source’)

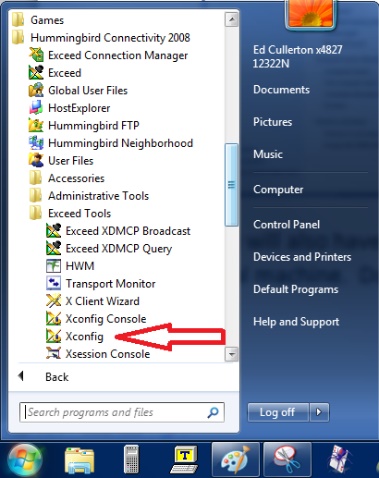
*source /export/home/rfies/esd/useresdconfig*

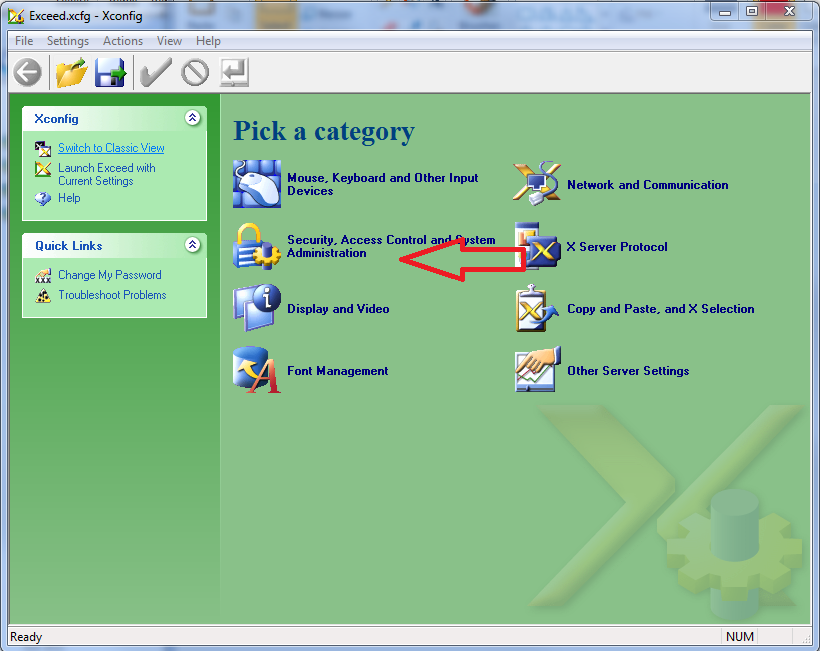
(Enter *rfies* for the LLRF group)

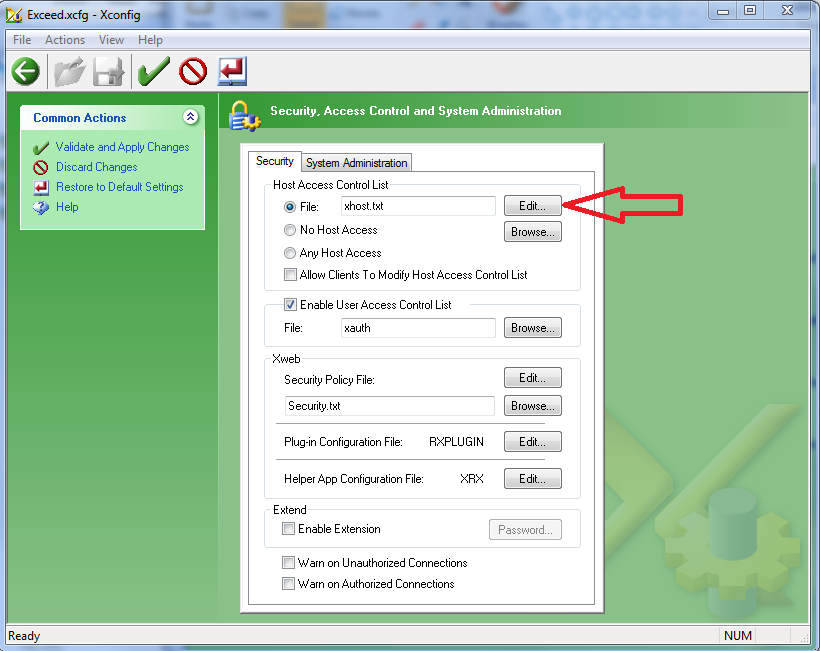
(Enter *y* for yes to use the example bash configuration)

Next, we need to configure your local Windows machine to use an X-Windows type text editor from your nova account, like “nedit”.

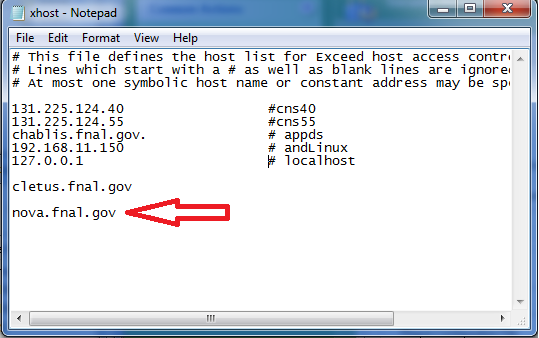
The first thing you will need to do is add nova.fnal.gov to the list of remote computers (xhost.txt) allowed to display X-Windows on your local machine. Do this by opening Hummingbird Xconfig:



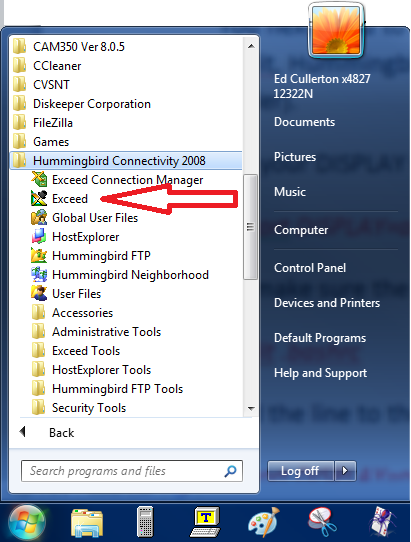




Add nova.fnal.gov to the list at the end of the file.



Save and close the text file and start Hummingbird Exceed on your local windows machine from the Hummingbird folder.

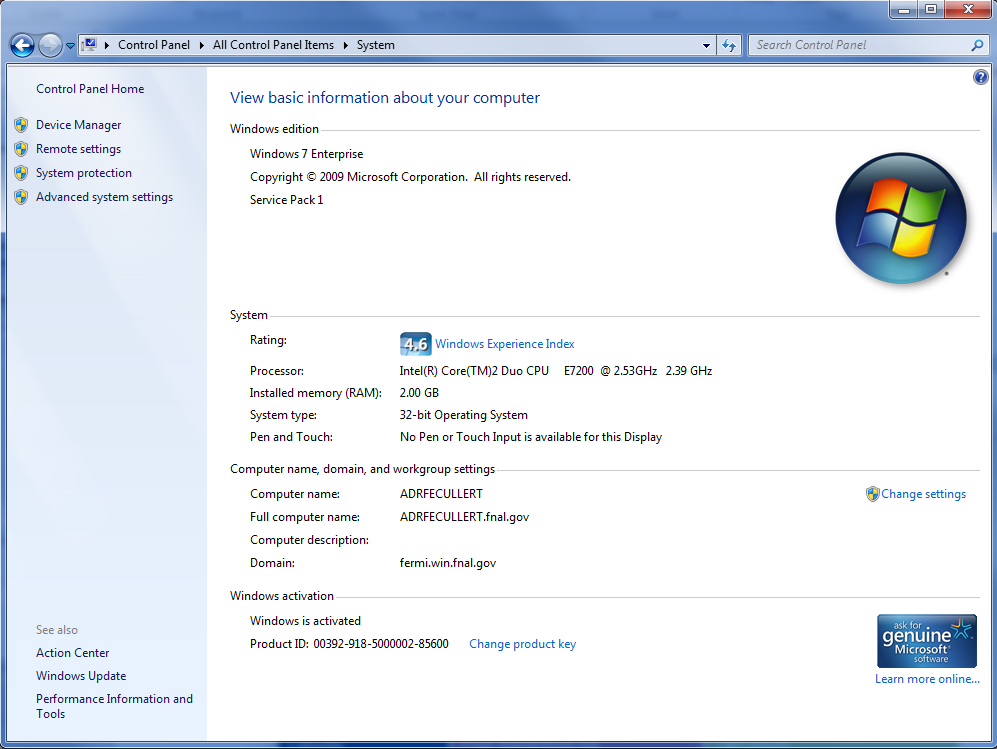


You will not see a window pop up, but Exceed will be running in the background.

Return to your nova terminal and set your X-Windows DISPLAY variable to your local machine:

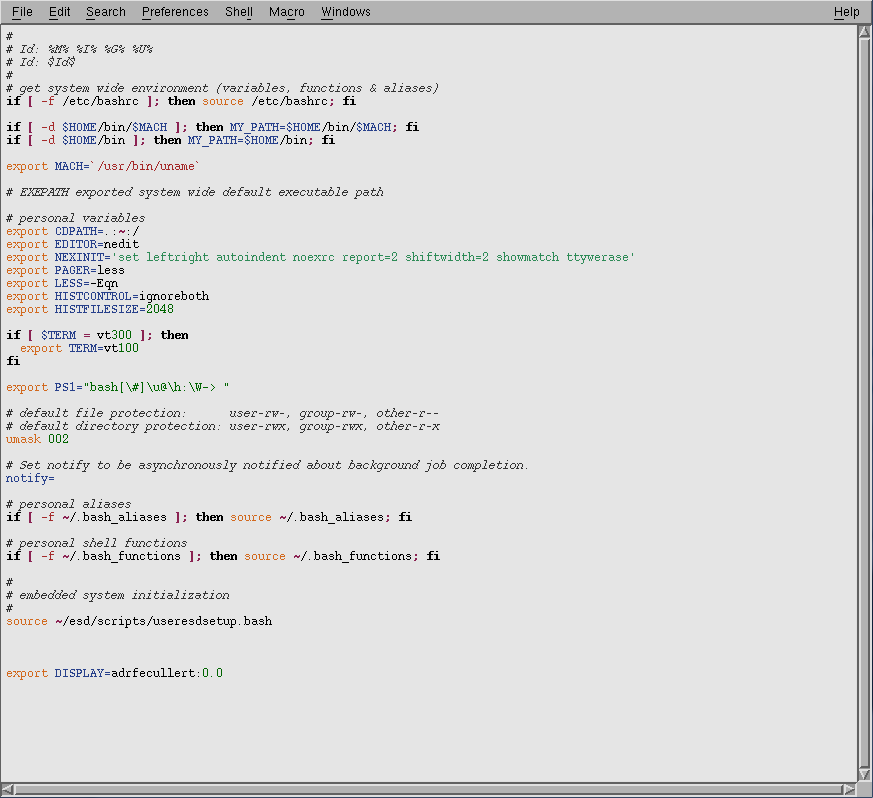
*export DISPLAY=computername:0.0*

This will make sure that X-Windows from your nova account will be displayed on your local Windows machine. If you do not know the name of your local Windows machine, go in the control panel, under System, and find out.



To make sure the DISPLAY variable is set at every login, edit the .bashrc file:

*nedit .bashrc*



Add the line to the bottom of the file:

*export DISPLAY=computername:0.0*

Also notice the line above that reads: source ~/esd/scripts/usersetup.bash. This runs a script at login to make your RFIESD software tools available at startup. Make sure that this line is not commented out. Save the file and close it. Your nova account is now ready for creating new projects.

**2. Setup a New Front-End Project**

From you nova terminal, start a new project by using the RFIES “setup” tool by typing the following command: (The project ‘edstest’ is used for example in this document. Please choose your own project name.)

*setup edstest*

If the project does not exist, it will create a new project for you. To make things easier later, make the project name six letters. You will be asked some questions if you are creating a new project. Since we are creating a new front-end project, enter yes to add front-end code templates to your new working directory. This will configure a new front end project for you. For more information, see section 4.2 of [2]. Any time in the future you want to work on this new project, type:

*setup edstest*

If the project has already been setup, the command will bring you to your working directory. To see the files in you working directory, type:

*ls*

You should see the front-end templates that were created for you (Makefile, Target, and UDF.dox). Now we will add your new project to the CVS repository [3]. The CVS repository exists in the file directory “$CVSROOT/fermi”. All the LLRF front-end projects are in the directory “$CVSROOT/fermi/rfies/fe”. The first thing you need to do is add an alias for your project to the CVSROOT/modules file. CVSROOT is a project in the repository. Check out a copy of the CVSROOT project from the CVS repository into your “sandbox”. Your sandbox is a local copy of the files and directories in a CVS repository. If this is your first time checking out the CVSROOT project into your sandbox, you can get the CVSROOT project into your sandbox by typing:

*cd $USER\_SANDBOX\_DIR*

*cvs checkout CVSROOT*

*cd CVSROOT*

If you have already checked out CVSROOT previously into your sandbox, you can just type:

*setup CVSROOT*

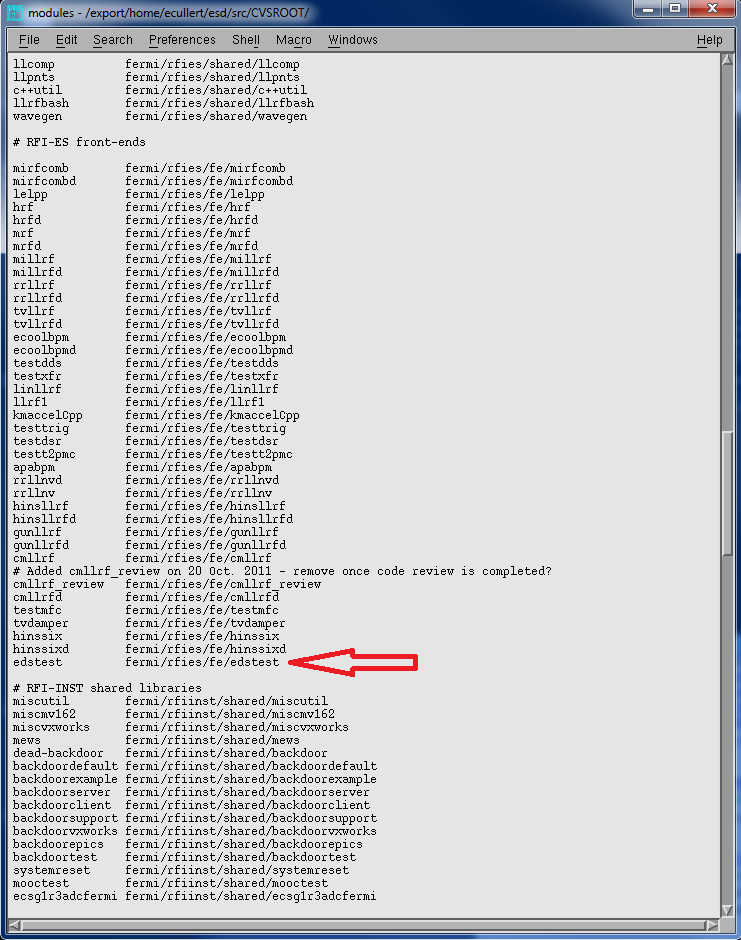
*update*

The ‘update’ command updates all your files in your sandbox to the version of the file that is in the repository. Now open a text editor and edit the ‘modules’ text file to add your new project to the list.

*nedit modules*

Add the alias specification for your project by typing the following line into RFI-ES section:

*edstest ferm/rfies/fe/edstest*



Save the file, close the file. To put the change in the ‘modules’ file back in the repository, you have to ‘commit’ the change back to the repository by typing:

*cvs commit*

A text file will pop up for you to add a comment at the top of the file to explain what you did. For this CVS commit, enter something like:

*Added CVS alias for new project edstst*

Now you have added your project to the list of CVS project, and now you are to ready to add you files to the repository.

You can check all the projects that you have setup by looking in the directory (cd $USER\_SANDBOX\_DIR, or cd /export/home/ecullert/esd/src). Type *ls* and you should see all the projects that have been setup. You should see at least your new project, and the CVSROOT project. As a sanity check, you can check your environment variable “$USER\_SANBOX\_DIR” by typing:

echo $USER\_SANDBOX\_DIR

You should see the command return the directory location of your sandbox: (/export/home/ecullert/esd/src)

The next step is to make sure you are in your new projects working directory, or sandbox, by typing:

*setup edstst*

Make sure all the files in your working directory/sandbox have owner and group write permissions by typing:

*chmod 664*

Now add the files in your working directory/sandbox to the CVS repository by typing:

*cvs import fermi/rfies/fe/edstst*

Another window will pop up and you can add a comment about adding your new project to CVS for the first time.

Take a look at the website to see that your new project files are in the repository:

<http://www-bd.fnal.gov/cgi-acc/cvsweb.cgi/fermi/rfies/fe/edstest/?cvsroot=Front-End>

To test the repository, make a small change to the file Makefile in your working directory, or sandbox. Use nedit to edit the file.

*nedit Makefile*

Find the line,

*#SCRIPTS = $(wildcard \*startup)*

and remove the # sign. This will allow you to run the RFIES command ‘make installscript’. The command ‘make installscript’ will be used later. Save the file and close the file.

Commit your sandbox to the CVS repository to save the change to Makefile in the repository.

*cvs commit*

You will get another pop up window, and add a comment like,

*Made a change to Makefile to allow “make installscript”*

Take another look at the website:

<http://www-bd.fnal.gov/cgi-acc/cvsweb.cgi/fermi/rfies/fe/edstest/?cvsroot=Front-End>

See that the files that are in the repository, and verify that a new version of the Makefile that you just committed is in the repository. There are many more important details about CVS that you will need to learn, but for now, lets move on to the hardware.

**3. Turning on the Hardware**

Now we need an actual front end. Get a VXI crate and a MVME 5500 controller with a Kinetic Systems v15x VME to VXI chassis and plug it into slot 0. Have the controller registered with networking, and give the controller the same name as your project if possible, to make things simple. (ie, edstst.fnal.gov) Plug in an ethernet cable and connect it to the network.

Next, connect a cable between the serial port of your local windows machine and the COM1 port on the slot 0 controller. Open a terminal on your local machine. Tera Term Pro is a terminal program used by the LLRF group, and it can be downloaded for free off the web. When Tera Term prompts you for a connection type, connect through serial port COM3. The COM port may be different depending on your serial port hardware.

Now get the latest version of VxWorks boot image loaded into the controller FLASH. The controls department has a web page with instructions on how to install a VxWorks boot image on the controller:

<http://www-bd.fnal.gov/issues/wiki/BootRoms>

The guide will ask you to power up your VXI crate. Some older VXI crates take a few power cycles to turn the controller on correctly. You can tell if the controller is starting correctly by looking at the green CPU light by the COM 1 port and making sure it stays lit for a few tens of seconds after turning on the VXI crate.

The latest VxWorks version that is supported by the Fermilab controls department is version 6.4, so click on the link to MVME-5500 Series VxWorks 6.4 and follow the instructions. Be aware that you will have to change your baud rate from 9600 to 38400 in your Tera Terminal program after loading the latest 6.4 kernels. When you get to the part where you set up the boot parameters, you can define some of the parameters to ensure proper boot. You will need to define the boot device, processor number, host name, file name, inet on ethernet, host inet, gateway inet, and user. Don’t worry about assigning the target name or startup script for now, you can leave that part of the boot parameters blank. The boot parameters should read as such:

boot device: gei0 (dc0 for MVME2434, gei0 for MVME5500)

processor number: 0

host name: fecode-bd

file name: /vxworks\_boot/kernel/mv5500/v6\_4/vxWorks\_6\_4\_5500\_a

inet on Ethernet is: 131.225.zzz.xxx:ffffff00 (get this info from your network assignment)

host inet: 131.225.121.145 (fecode-bd address)

gateway inet: 131.225.zzz.200 (zzz should be the same as zzz in the inet on Ethernet)

When you reach the ‘Final Steps’ portion of the guide, you may need to contact Denise Finstrom or Tim Zingleman to gain access to the fecode-bd machine to add your front-end to the list of those authorized to load boot files. In the next section, we will create a startup script and tell the boot parameters which directory to load the startupscript from.

**4. Startup script**

The startup script is a collection of commands, or functions, for the VxWorks realtime operating system. The commands are used to configure the operating system so that it is able to load any code, or modules, related to the functions of your LLRF system. To fully understand each command or function, you will need the Wind River VxWorks manuals handy. The Fermilab controls department has a collection of manuals at the website:

<http://www-ad.fnal.gov/controls/micro_p/manuals/vxworks6.html>

There is also a nice cheat sheet I found on the web with a list of useful commands:

<http://touro.ligo-la.caltech.edu/~cparames/CDS/vxWorks_commands.html>

When you turn on your new front end, the slot 0 controller will go to a remote machine to load all its code. Fecode-bd.fnal.gov is the machine where the slot 0 controller will go to load all your code. The controller will be told where to load code by its *boot parameters,* which was partially setup during the flash of the boot image. At boot, the front end will go to particular fecode-bd directories to find a kernel for your controller and a startup script to define what other code will be loaded onto the controller. The file location of the kernel was entered into the boot parameters earlier.

We first need to create a directory for your project on fecode-bd [4], which is the directory where the controller will load code from. Make sure you are logged into nova, and type:

*mkdir /fecode-bd/vxworks\_boot/fe/edstst*

Give proper read/write permissions to that folder:

*chmod 775 /fecode-bd/vxworks\_boot/fe/edstst*

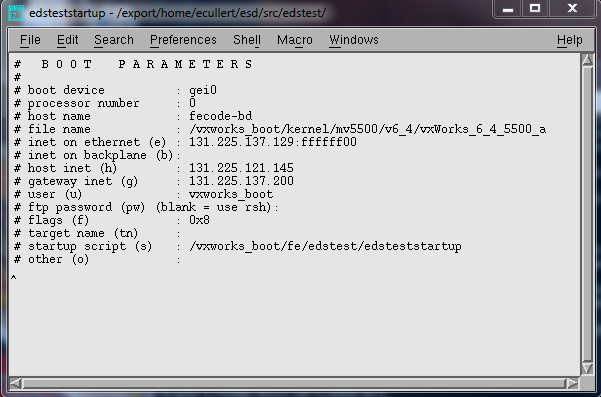
Once you are done with that, we need to create a startup script to put in that fecode-bd directory. There is a defined procedure for putting code on fecode-bd that we will discuss later.

First, create a startup script in your working directory using nedit. Make sure that you are in your project sandbox by typing:

*setup edstst*

*nedit edstststartup*

A simplified startup script shown below contains only comments about the boot parameters. These comments should reflect the actual boot parameters of the slot 0 controller. One should be care to make sure that these comments are accurate, otherwise confusion may set in.



Once you have saved your startup script in your sandbox, add the file to your cvs repository:

*cvs add edstststartup*

Add an appropriate comment to your CVS commit.

Now use the RFI-ES “make” tool to install your startupscript on fecode-bd:

*make installscript*

You can check to see if you startup script was installed correctly by:

*cd /fecode-bd/vxworks\_boot/fe/edstst*

*ls*

You should see a copy of your startup script in that directory.

I would suggest taking a look at the RFI-ES directory diagram [5] and find the pointer where “make installstript” is used to place the startup script from your sandbox into the fecode-bd directory. We will be using the “make” tool to install code from our sandbox to fecode-bd in the proper manner. Do NOT move code into the fecode-bd directories by any other method. No links, no copying, or no moving. Just “make”. More rules for using make will be discussed later in the section on code.

Next we need to change the boot parameters of your slot zero controller in order to tell it where to load the startup script. Start by powering up your VXI crate. We know that the controller will boot the 6.4 kernel, because we defined this in the boot parameters when we flashed the kernel. Wait for the prompt in your Tera Term Terminal. When you get a prompt, type:

*bootChange*

This command allows you to change the boot parameters of the controller. If you don’t get a prompt, because of some error in the boot parameters, you can press any key immediately after starting the crate, and you will get a prompt:

[VxWorks boot]:

You can press “c” the change the boot parameters in the same way as “bootChange”.

The parameters should look like this:

boot device: gei0 (dc0 for MVME2434, gei0 for MVME5500)

processor number: 0

host name: fecode-bd

file name: /vxworks\_boot/fe/edstest/vxWorks-55

inet on Ethernet is: 131.225.zzz.xxx:ffffff00 (get this info from your network assignment)

host inet: 131.225.121.145 (fecode-bd address)

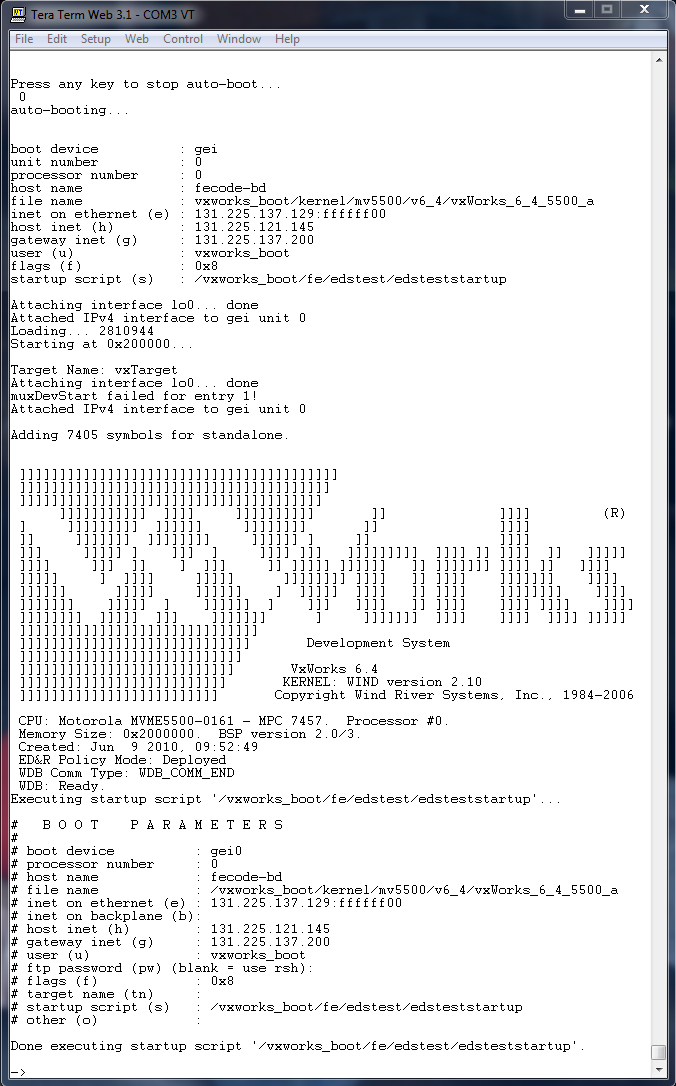
gateway inet: 131.225.zzz.200 (zzz should be the same as zzz in the inet on Ethernet)

target name (tn) :

startupscript: /vxworks\_boot/fe/edstest/edsteststartup

After completing your changes, verify your changes by typing *bootChange* again and scrolling through the parameters. For more information on boot parameters, see [6], p118. Or Paul’s document [7].

Once your boot parameters are properly defined, restart the VXI crate and watch the boot process in the serial port terminal. If all was done correctly, you should see your new startup script being loaded and your comment echoed in your terminal. Below is what a proper boot sequence should look like.



After a successful boot, we can add some new VxWorks functions to our startup script. Make sure you are logged into nova and are in your working directory, or sandbox.

*setup edstest*

*nedit edsteststartup*

The first VxWorks function we will call from our startup script is nfsMount(). This VxWorks function mounts a UNIX type NFS directory from another machine to your local VXI controller. Add the following line to your startup script to mount the new project directory that we created on fecode-bd, and call it “/home”.

*nfsMount( "fecode-bd", "vxworks\_boot/fe/edstest", "/home" )*

This will give us the ability to load any code that we created in our new project. Since we have not created any code, there is nothing to load right now.

The next directory we are going to mount contains many generic files, or libraries, used in multiple systems. The libraries contain support code for the Kinetic Systems VME to VXI chassis, Universal Clock Decoder (UCD), Simple Network Time Protocol (SNTP), MOOC and ACNET, MFC FPGA board, and various other support code.

*nfsMount( "fecode-bd", "vxworks\_boot/fe/rfies/lib/VW\_55/MVME5500", "/targetLib" )*

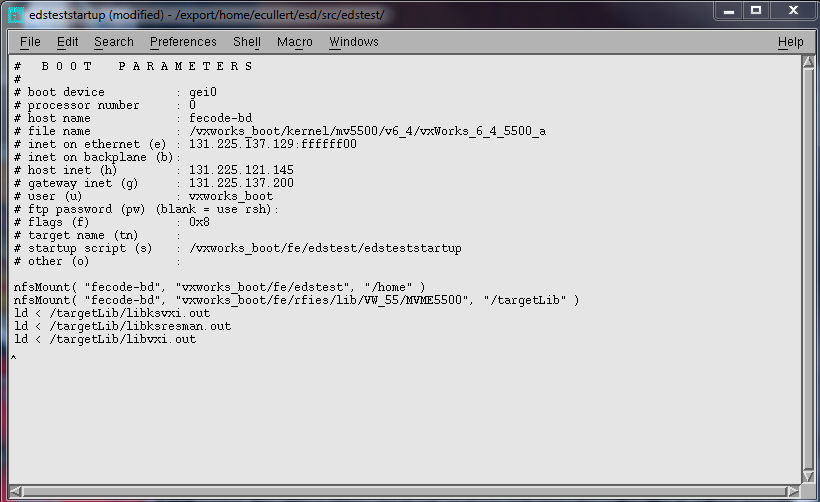
Now we want to load some modules from the new mounted drive “/targetLib”. The modules we want to load are a set code to make the Kinetic System VME to VXI chassis work correctly. We will use the VxWorks function “ld” to accomplish this.

*ld < /targetLib/libksvxi.out*

*ld < /targetLib/libksresman.out*

*ld < /targetLib/libvxi.out*

Your startup script should now look something like this:



After you have made the changes, save the file, close the file and do a “make installscript” to install the new startup script in the fecode-bd directory.

*make installscript*

Open your Tera Term Pro terminal and turn on your VXI crate. Once the controller has successfully booted, you can verify that your new drives have been mounted, and the modules have been loaded. From the Tera Term VxWorks terminal, type the VxWorks command “devs” to see a list of all the devices on the system. You should see your newly mount drives “/home” and “/targetLib” in the list.

Next type the VxWorks command “moduleShow” to see the modules that were loaded. You should see the three modules that were loaded from the fecode-bd directory ‘targetLib’ by the startup script.

Up to this point, we have learned the basics of getting a controller booted, mounting remote fecode-bd directories on the controller, and loading code modules on the controller. Before we go any further, we need to talk about rules for developing code, and this will be discussed in the next section.

**5. Code Development Procedures for the LLRF Group**

**References**

[1] “Configuring Accounts on Nova”, D. Voy.

[2] “Managing Embedded System Software with the RFIES Development Tools”, D. Voy.

[3] “Creating CVS Projects”, D. Voy.

[4] “Downloading Front-ends from fecode-bd”, D. Voy.

[5] “RFI-ES Embedded System Development File Organization”, D. Voy.

[6] “Vxworks\_kernel\_programmers guide\_6.7”,

<http://www-ad.fnal.gov/controls/micro_p/manuals/vxworks_kernel_programmers_guide_6.7.pdf>

[7] “LLRF Embedded System Development”, P. Joiremen.

[8] <http://www-kryo.desy.de/documents/vxWorks/V5.4/vxworks/netguide/c-tcpip3.html>

[9] <http://www-ad.fnal.gov/controls/micro_p/manuals/vxworks_kernel_api_reference_vol1_6.6.pdf>