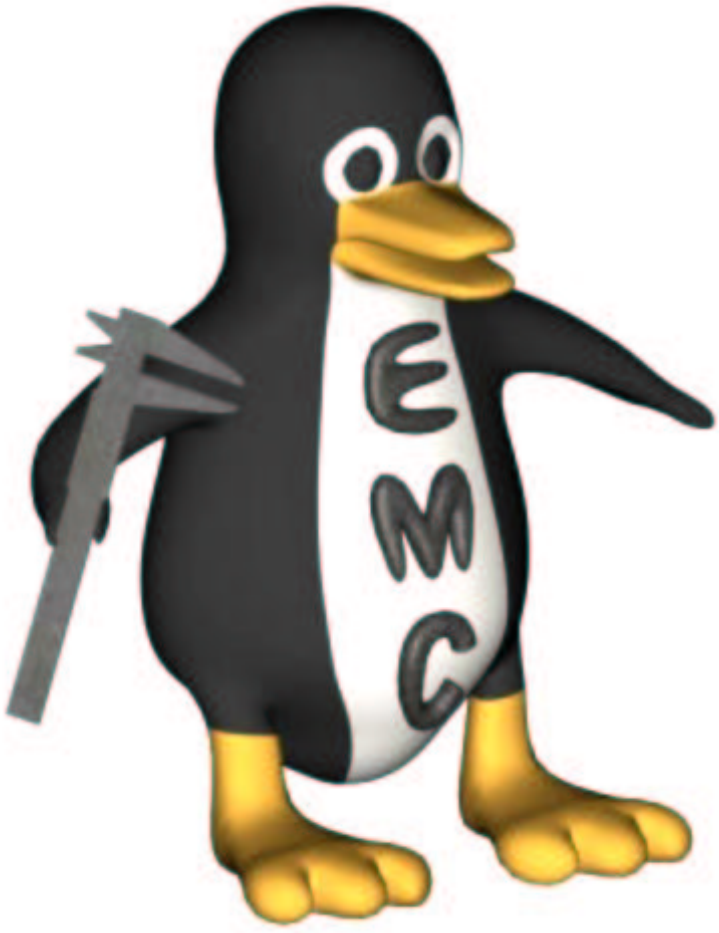
The Enhanced Machine Control Integrator Handbook



The EMC Team[1]

20th July 2003

[1]Ray Henry <rehenry@up.net> serves as the general editor. Authors include Dan Falk, Will Shackleford, Fred Proctor, Jon Elson, Henkka Palonen.

This handbook is a work in progress. If you are able to help with writing, editing, or graphic preparation please contact any member of the writing team or join and send an email to emc-users@lists.sourceforge.net.

Contents

Chapter 1

Introduction

This handbook describes compiling the EMC from source and set up for specific kinds of machines.

(flesh this out using current machine parameters)

Chapter 2

Installing EMC

Installation can be a daunting task to people new to Linux. The hardest part is getting the Real Time Linux patch up and running. After that, installing EMC is pretty easy. With that said, most people have found that the easiest way to install Real Time Linux and EMC is to go to Tim Goldstein's web page and use his install script for EMC on the linux 2.0.36 release. Or you can find or order a copy of one of the BDI install cdroms that are produced by Paul Corner. The 2.x disk installs Linux 2.2.18 and Real Time Linux 3.0 and comes with a recent EMC release. The TNG disks installs Linux 2.4.20 and RTAI Real-Time. This series is no longer being upgraded by Paul. The Live series will boot and run from CDROM and can be used to install a Morphix version of Linux 2.4.20 and RTAI Real-Time.

2.1 Patching the RedHat(tm) 8 distribution3

The following section was written by Kim Lux. It is a report of how to install the real time patches to the standard kernel supplied with the RedHat(tm) release 8.0. Kim's reports on compiling the EMC and the RCSLIB are included in the EMC build chapter.[1]

Here is my foolproof method of installing EMC (and RedHat 8 with a RT OS) on a clean hard drive. ăDon't be mislead by the title of this document: you aren't a fool if you've got problems with these instructions, they've just always worked for me. ă

I'm breaking the process down into several steps:

1. Build a kernel, for the fun of it.
2. Patch the kernel you just built with the rtai real time patch.
3. Build a kernel for real.
4. Build and install EMC
5. Enjoy !

[1] CAVEAT: At the time of writing this, I have not installed nor tested EMC on an RTAI kernel.

These instructions are volumous, but you might learn something along the way too. ăI suggest reading them from beginning to end before starting. The whole process is quite a bit simpler than it looks.

A few notes before we start:

1. If you don't understand the commands I use in this document, I suggest you try the following information sources:
   (a) >man <command>. ăYou'd be surprised how much you can learn reading man pages.
   (b) google the command on the internet: www.google.com. ăTry both the web and the groups.
2. I run the build process as root. ăSome commands don't need you to be root, but some do. ăI don't know which is which and I'm not patient so I run as root. ăI think that "make install" is one that needs root for sure. ă
3. There is no such thing as a dumb question, especially in the Linux community. ăIf you don't know, ask. ăI suggest googling before asking. Newsgroups are a wonderful thing. ăalt.linux.redhat and alt.redhat are two good ones.
4. I'd like to make this document as helpful as possible. ăIf you've got suggestions, I'm all ears.
5. You needn't build every EMC hard drive from scratch. ăI've got a "How to clone a Linux drive" document that outlines the process. ăCloning is much easier than starting from scratch every time. ăI'll gladly clone you a hard drive of our EMC setup for the price of a hard drive plus a nominal fee for someones time to run it. ă
6. The modern Linux OS is a great operating system in its own right. ăIt is very stable, very powerful, well supported and diverse. ăThose of you coming from a Windows background are probably wanting to do as little with Linux as possible to get EMC working. ăAlthough this document will probably help you achieve that, I urge you to give it further consideration. ăYes, there is quite a learning curve to Linux. ăHowever, after some reading and experience with it you'll want no other OS. Linux newbies are going to take a look at the process to build a kernel and shake their head. ăAll this to run EMC ? ăThe flip side to that statement is that it isn't possible to make Windows into a real time OS to properly run something like EMC. ăYes, Windows can be made to "work" as a CNC controller, but it isn't the same. ă ă ă
7. Linux is a user help user community. ăIf this document helps you install Linux/EMC, you should help someone else out in the future. ăI've personally benefited from numerous Linux people on numerous Linux issues and thus I share this. ăNow it is your turn.
8. The whole install/build process is slow. ăIt doesn't require much human time, but it takes a long time to run, especially on a slow computer. ăIt also requires some concentration on the part of the person doing it. ăI personally do this sort of thing at home in the evenings, away from the office so that I'm not interrupted by co workers, ringing phones, etc.

9. First impressions are frequently wrong ! ăBuilding and patching a kernel is just about the most complex process a typical Linux user will ever undertake. Everyday Linux use is a lot like using Windows. ăIf these are your first days with Linux, don't think they will all be like this. ăBuilding kernels is not an everyday (or even every month) activity in Linux. ă

10. What does a "make" command do ? ăMake commands manipulate human readible source code into machine executible files. ăA linux archive contains literally hundreds of files containing drivers, functions, ă

2.1.1 Step 0: Install Redhat 8

I'm building our EMC systems around RedHat 8 Linux because we use it on a bunch of other computers. ăI'm very familiar with RH8 and thus it makes a good starting point for me. ăFurthermore, RH8 is a fairly complete and stable OS unto itself. ă I'm going to have employees operating our EMC driven equipment and I want them to have an OS at that station that is fully equipped (ie it has email, a desktop GUI, etc) and a simple graphical editor. ăRH8 has all this and is easy to use as well.

It seems kind of weird to be running the same basic operating system on our CNC mill as our accounting server, but I guess that speaks to the power and diversity of Linux.

A few hints about installing RH8:

1. Select either the personal installation or the workstation installation, but select the OPTIONAL inclusion of the Kernel Development Support files. To do this you must enter the "Customize" section of the package installation routine. If you forget to do this during the install, you can run the "packages" utility and install this stuff after the fact.

2. Check the validity of the install CDROMs. ă One of the first options in the installation process is the ability to check the installation disks for errors. ăTWICE I've been hampered by bad CDROMs when installing Linux. ăFrom my experience it is well worth the time to check the CDROMS. ăThis is a lengthy process with a slow computer/slow CDROM drive, but it runs unattended so I do it before actually doing the install. ăPreparation always pays off.

3. RH 9 is not a good EMC platform, not at the time of this writing. ăRH 9 uses a different process threading system than RH8 and there isn't an available run time patch to work with it. ăI'm sure this will change in the future, but for now I recommend sticking with RH8.[2] ă

4. Most people, Linux newbies especially, aren't comfortable with using emacs as their editor. ăI highly recommend installing the graphical version of emacs, ie xemacs during installation. ăI think it is under "Editors" in the custom installation options. ăMost of us won't need vi. You can delete that if you like. ă(I think I just insulted 80% of the Linux community...) ă

[2]Note: Some people are claiming success running Kernel 2.4.20 under RH9 with the real time patch for it. ăI can't say if this works or not. ă

5. RedHat 8, workstation edition takes about 2 GB of disk space. ăThe personal edition takes sightly less, maybe 1.7 GB. ăGone are the days when you could fit RH with X Windows on a 1 GB drive. ăThat isn't necessarily a bad thing either.

6. RH8 has very good user documentation. ăThe last CDROM is dedicated to the documentation. ăI've printed all the manuals (our printer has a duplexor) and have them bound in a 3 ring binder for instant access. ăIf I were a newbie, I'd for sure print and read and make notes in the installation document. ăIt outlines things like hard drive partitioning, etc. ăLinux isn't Windows, it is different. ăSome of what you know about Windows applies, but some things are outright different. ăThere is no "C:" drive in Linux and that is a good thing.

7. I don't recommend dual booting. ăI'm sure it works, but it is probably more hassle than it is worth. ăIf you really need to run Windows, I suggest using removeable hard drives and leaving your Windows drive alone and installing Linux on its own hard drive. ăIf you need to send data between Linux and Windows, just boot Linux and mount your Windows drive into the file system. ăThe command to do that is this:

   >ămount -tvfat /dev/hd?1 /windows where ? = b,c or d, depending where you've got the Windows drive mounted. (See man mount for more info.)

   As far as I'm concerned, removable harddrives are highly underrated computing accessories. ăThey are right up there in utility with wireless network cards, LCD touchscreen monitors, software RAID and Linux itself.

8. If you are a newbie, don't be in a hurry. ăTake this process step by step and stop if you get stuck. ăThink about it, come back to it. ăAsk someone else for help. ăOne can waste a lot of time trying to solve a simple problem when trying to do things in a hurry. ăI recommend downloading and burning your RH8 disks one evening, installing Linux another evening, compiling your first kernel another evening, compiling the real time kernel another evening and then building and installing EMC on the last evening. ăNone of these tasks should take an evening but you'll be frustrated if you think you can just burn through the whole process in an afternoon. ăPace yourself and stop before you get tired or frustrated if you get stuck.

9. Keep notes. ăThis might not be your last linux installation/build ! I personally keep a "linux journal" with all the notes from things I do in Linux. ăA lot of things I remember off the top of my head, but not if I haven't done them for 3 months ! ăA quick flip through my linux journal refreshes my memory and I'm solving my problem without starting from scratch. ăFor instance, I haven't built a kernel in 4 months and I had to dig up my kernel building notes to build this one. ă

For further information on installing RH8, consult "The Official Redhat Linux x86 Installation Guide", which is on the last disk of the RH8 set. ă

2.1.2 Step 1: Build a kernel, for the fun of it.

Before we get all caught up in properly patching our kernel for use as a run time kernel for EMC, I suggest building a regular kernel (from scratch) to get a feel for the process. ă(Baby steps... walk before you run.) Building a kernel and then running it will give you the confidence and understanding to patch and build the real time kernel. ă

Here is a sure fire method for building new kernels in RH8:

Get the source code for the new kernel.

RH8 ships with kernel 2.4.18-14. ăThat number is a version/release number. ăDon't be scared by it, it is just a number. ăAs of this writing, the latest stable "production" kernel is 2.4.20. ăThere is also a real time patch available for this kernel, thus it is the one that I am going to build in these instructions. ă

Using your favorite web browser, go to www.kernel.org and follow the links to the download section. Find the full source code archive for 2.4.20, not the patches. ăDownload and save the kernel archive in your home directory ie /home/"you". The file name you download should be something like linux-2.4.20.tar.gz. It is about 32MB in size. The url for that is http://www.kernel.org/pub/linux/kernel/v2.4/

From here on in, I work in a command terminal. Fire up your terminal, which is usually located in the Start->System Tools menu. ăNote that the fonts in your terminal may be pretty ugly or garbled. ăSet the fonts to something you like by using Settings->fonts. ăYou have to save settings or the next time you start your terminal they will be garbled again. ă ă

Unzip the archive in /usr/src/

First we need to copy our Linux archive (ie Linux-2.4.20.tar.gz) into the /usr/src directory. ăYou can do this with the following Linux command, assuming you are in your home directory and you saved the archive there:

check if it is there: >ls Linux* Linux-2.4.20.tar.gz

copy it to /usr/src: >cp Linux-2.4.20.tar.gz /usr/src

Now change directories to /usr/src: >ăcd /usr/src

Now we can unpack the archive with the following command ă >ătar -zxvf Linux-2.4.19.tar.gz

I recommend making a new directory for each version of the kernel that you build,ie 2.4.28, 2.4.19, etc. ăUntaring the tar will do this automatically. ă

Check that Linux actually unpacked the archive:

>ăls Linux* You should see a directory called "Linux-2.4.20". ăNote that there might be directories for other Linux kernels. ăI've got a directory for linux-2.4.18 as well. ă

Change directories into the new directory: >ăcd linux-2.4.20

4) Do the build process. ăDon't skip any steps. Now that we've got the source code unpacked, we are ready to start building the kernel.

a) Issue this command. ă(You should be in /usr/src/linux-2.4.20. ăYou can use "pwd" to check where you are.) >ămake mrproper

A bunch of text is going to go scrolling by on your terminal. ăIt is just the make process doing its thing.

b) Now we want to make the "oldconfig" setup. ăIssue this command:

>ămake oldconfig

Everything is straightforward with this EXCEPT that every feature that has been added to the Linux kernel since the last version is going to come up for verification. ăAssuming that I am happy with the way the last kernel ran, I ALWAYS just press enter at each question, accepting the default.

c) Now we want to configure the kernel we are going to build. ăWe do this by using one of the two commands. ă

>ămake xconfig

or ă

>ămenuconfig

A couple of notes here.

First, I couldn't make xconfig on some of my RH8 installations, so I had to settle for menuconfig. ăI get make errors trying to make xconfig. ăI didn't bother looking into it.

Menuconfig is the graphical version, whereas xconfig is the XWindows version. ăBoth work equally well. ăSometimes my terminal fonts are yucky and xconfig works better. ăSometimes I can't make xconfig and I use menuconfig. ăLinux gives you both. ă

Secondly, when you run the config program you will be bombarded by choices of various features, hardware, etc. Rather than try to figure out what you want, I think it is imperative to import the config settings from a known working RH Linux config file as a starting point for the build. ă

To do this, at the bottom of ămenuconfig, one can elect to import the settings of a current config file as the starting point of the config. ăI highly recommend that one does this, just to get a feel for what RedHat included if nothing else. The file to use as the starting point is

/boot/config-2.4.18-14

or whatever your most current kernel is. Note that this file does not start with .config as menuconfig (and the Kernel-HOWTO doc) suggest. Once you've imported the file you can look around a bit, make any changes you want and then save and exit. ăOne really gets a feeling for the depth and breadth of Linux by looking at all the various hardware and feature options. ă

NOTE: THERE IS A BUG IN THE linux-2.4.20 source that might need to be addressed here. ăWith the source archive that I downloaded, I need to enable "Point to Point Protocol" and "PPP Deflate Compression" as "kernel" features in order to get around a zlib compression and decompression include file problem.

To enable these two items in the config process, look under the "Networking Protocol" section and find "Point to Point Protocol". ăIn my version of

RH8 config it was enabled as a Module feature ie with an "M". ăChange that to a "Y" for kernel feature. ăA few lines down, change the "PPP deflate compression" setting to a "Y" also.

NOTE: THERE IS ANOTHER BUG IN THE linux-2.4.20 source: the low level SCSI drivers for the NCR 53710 device won't compile under the version of gcc included with RH8. ăTo get around this problem, I delete the drivers for the 53710 devices by doing the following:

i) Open the SCSI Support option, then the Low Level Drivers option. ii) Find the NCR 537,8xx driver. ăDisable it, ie set it to "N". iii) Find the Simple 53710 driver, about 10 lines below. ăDisable it, ie set it to "N". iv) Be sure to press OK on the way out of this screen.

Another work around for this issue *might* have been to use the Alan Cox kernels, which are typically noted as suffix -ACX, where X is the number of the latest release. ă(ie linux-2.4.20-AC3, for example.) Alan Cox is a noted Linux code cleaner and I'm sure his source wouldn't have this problem, although I didn't try it. ăOne of the possible issues with using a "-ACx" kernel is that the real time libraries were probably built against a stock kernel and there might be conflicts applying them to an "-ACx" kernel. ă ă

BTW: the error caused by the lack of the zlib library files showed up during the "make bzImage" process. ăIt stopped with an error complaining the zlib library wasn't available. ăWith the PPP options enabled as Kernel, the proper include files are included and the problem is solved. ă

The 53710 SCSI driver error showed up during "make modules".

d) Now we build the dependency file. ăDo this by issuing the following command.

>ămake dep ă Once again you can sit back and watch text scroll by. ăThe only thing to watch is that each of these commands ends normally, ie they don't end with "Error..."

e) Next we issue the following command:

>ămake clean

Like the command suggests, this does cleanup

f) Next we make a boot image by issuing the following command:

>ămake bzImage

This is a rather lengthy build, even on my machine which is a 2GHz AMD with 512 MB of RAM.

The Kernel-HOWTO suggests that the resulting image needs to be manually copied to the /boot directory. This is incorrect for RH8 builds. If you perform all of the steps in here, ie: "make install" it ends up there on its own. ăNo manual copying is necessary.

g) Next we need to make the modules for the kernel to run. ăTo do this we issue the following command: ă

>ămake modules

This is a really long compile, even on a fast machine. ăHave entertainment ready. ăRemember that Linux is multi tasking OS and you can surf the web or write G code while waiting for this to compile.

h) Next, we install the modules. ăThe command to do this is:

>ămake modules_install

i) Next, we install the kernel itself. ăTo do this we issue the following command:

>ămake install

5) configure LILO or GRUB.

In previous Linux versions one had to manually configure the boot loader (ie Lilo or Grub) to allow the kernel to be booted. RH8 uses Grub and you don't have to configure anything. ăIf you open grub.conf in /boot/grub, you'll find that make adds the appropriate lines necessary to boot the new kernel. Here is how:

>ăcd /boot/grub >ăemacs grub.config

You'll see that there is an entry for linux-2.4.20 in the file. ăThe command to exit emacs is ctrl x ctrl s, I think. ăI always use xemacs, so I close with the mouse... ă

6) Reboot the computer, boot your new kernel

Close your terminal window, logout of your session and reboot your computer. During the boot process, Grub will display a number of kernels to boot. Select your new 2.4.20 kernel. ă

Congratulations, you've just built your own linux kernel ! ăIt wasn't really that hard, was it ?

Once the "stock" kernel building process works properly, you can make whatever modifications you want, knowing that if there is a problem, it lies in your kernel configuration, not the build process. ăNote that we didn't destroy the old kernel. ăIf there is a problem with our new kernel, we can reboot our computer and select the old kernel. ăNothing is lost and nothing is damaged. ă

Other notes:

1) make oldconfig is very important when you use a config file from a different kernel release. Otherwise, you can skip it.

2) one could skip a step or three in this process. ăHowever, I can never remember which ones to skip and thus do things in the same order each time.

2) If you are already using the same kernel version and want to save the build files, either edit the makefile to use a different EXTRAVERSION= variable, or be sure to back up your /lib/modules/<kernelversion> directory before doing make modules_install.

3) if you forget which command you just ran, you can use the up arrow key to recall it from the terminal history. ăUse the down arrow key to get back to an empty line. ă

Step 1a: OPTIONAL: Rename your kernel so you don't overwrite it.

The purpose of this whole process is to build a real time 2.4.20 kernel. ăHowever, sometimes it is nice to have a "regular" 2.4.20 kernel to run. ăThe way the linux kernel build process works, every time you build a 2.4.20 kernel it installs it as a 2.4.20 kernel, overwriting the previous 2.4.20 kernel.

In order to avoid this, we need to rename the kernel files in the /boot directory and change the settings in the grub.conf file to match. ă ă To change the the file names, I rename all the files ending in 2.4.20 to something different, like this:

>cd /boot

>ămv config-2.4.20 config-2.4.20-stock >ămv initrd-2.4.20.img initrd-2.4.20-stock.img >ămv System.map-2.4.20 System.map-2.4.20-stock >ămv vmlinuz-2.4.20 vmlinuz-2.4.20-stock ă

Now I make a corresponding change in grub.config:

>ăcd /boot/grub >ăemacs grub.conf (or xemacs grub.conf)

I scroll down to the lines for my new kernel:

ăăăăăăăăăkernel /vmlinuz-2.4.20 ro root=LABEL=/ hdd=ide-scsi ăăăăăăăăăinitrd /initrd-2.4.20.img ăăăăăăăăă And change them to match the file names I gave in /boot:

ăăăăăăăăăkernel /vmlinuz-2.4.20-stock ro root=LABEL=/ hdd=ide-scsi ăăăăăăăăăinitrd /initrd-2.4.20-stock.img

We are now ready to build our next 2.4.20 kernel without destroying the one we just made.

Here is an example of my /boot directory. ăI've made a number of different kernels.

<paste begins> $ls boot.b ă ă ă ă ă ă ă ă ălost+found ă ă ă ă ă ă ă ă System.map-2.4.20 chain.b ă ă ă ă ă ă ă ă message ă ă ă ă ă ă ă ă ă System.map-2.4.20.old config-2.4.18-14 ă ă ă ă ămessage.ja ă ă ă ă ă ă ă ă vmlinux-2.4.18-14 config-2.4.18-18.8.0 ă ă ămodule-info ă ă ă ă ă ă ă ă vmlinux-2.4.18-18.8.0 error ă ă ă ă ă ă ă ă ă ă module-info-2.4.18-14 ă ă ăvmlinuz grub ă ă ă ă ă ă ă ă ă ă ămodule-info-2.4.18-18.8.0 ăvmlinuz-2.4.18-14 initrd-2.4.18-14.img ă ă ăos2_d.b ă ă ă ă ă ă ă ă ă vmlinuz-2.4.18-18.8.0 initrd-2.4.18-18.8.0.img ăSystem.map ă ă ă ă ă ă ă ă vmlinuz-2.4.19 initrd-2.4.19.img ă ă ă ă System.map-2.4.18-14 ă ă ă vmlinuz-2.4.19.old initrd-2.4.19-stock.img ă System.map-2.4.18-18.8.0 ă vmlinuz-2.4.19-stock initrd-2.4.20.img ă ă ă ă System.map-2.4.19 ă ă ă ă ăvmlinuz-2.4.20 kernel.h ă ă ă ă ă ă ă ă ăSystem.map-2.4.19.old ă ă ăvmlinuz-2.4.20.old <paste ends>

and the corresponding grub.conf file:

<paste begins> # Note that you do not have to rerun grub after making changes to this file # NOTICE: ăYou have a /boot partition. ăThis means that # ă ă ă ă ăall kernel and initrd paths are relative to /boot/, eg. # ă ă ă ă ăroot (hd0,0) # ă ă ă ă ăkernel /vmlinuz-version ro root=/dev/hda3 # ă ă ă ă ăinitrd /initrd-version.img # boot=/dev/hda default=0 timeout=10 splashimage=(hd0,0)/grub/splash.xpm.gz title Red Hat Linux (2.4.20) ăăăăăăăăă-root (hd0,0) ăăăăăăăăăkernel /vmlinuz-2.4.20 ro root=LABEL=/ hdd=ide-scsi ăăăăăăăăăinitrd /initrd-2.4.20.img title Red Hat Linux (2.4.19) ăăăăăăăăă-root (hd0,0) ăăăăăăăăăkernel /vmlinuz-2.4.19 ro root=LABEL=/ hdd=ide-scsi ăăăăăăăăăinitrd /initrd-2.4.19.img title Red Hat Linux (2.4.19) (stock) ăăăăăăăăăroot (hd0,0) ăăăăăăăăăkernel /vmlinuz-2.4.19-stock ro root=LABEL=/ hdd=ide-scsi ăăăăăăăăăinitrd /initrd-2.4.19-stock.img title Red Hat Linux (2.4.18-18.8.0) ăăăăăăăăăroot (hd0,0) ăăăăăăăăăkernel /vmlinuz-2.4.18-18.8.0 ro root=LABEL=/ hdd=ide-scsi ăăăăăăăăăinitrd /initrd-2.4.18-18.8.0.img title Red Hat Linux (2.4.18-14) ăăăăăăăăăroot (hd0,0) ăăăăăăăăăkernel /vmlinuz-2.4.18-14 ro root=LABEL=/ hdd=ide-scsi ăăăăăăăăăinitrd /initrd-2.4.18-14.img <paste ends>

Notice how the file names in /boot have been modified so that I have 2 2.4.19 kernels and grub.conf has been modified to correspond to one of them. ăIf you do this before the second build, the boot files will not be destroyed. (I used a 2.4.19 kernel before the 2.4.20 kernels were available.)

2.1.3 Step 2: Patching the 2.4.20 kernel to include the real time functions.

The next step in our journey to a running EMC computer is to patch the kernel with the latest real time OS patches.

a) The first thing we need to do is to download the real time patches. Using your favorite browser, go to www.rtai.org and download the latest patch set for your kernel into your home directory. ăThe numbering scheme at rtai is that their "24" patches work with kernels of the 2.4 series, so I downloaded 24.1.11.

b) Unpack the rtai archive.

Check that you have it:

>ăls rtai* <should give you rtai-24.1.11.tgz)

Now move it to the /usr/src directory:

>ămv rtai* /usr/src

Now change directories to the /usr/src directory:

>cd /usr/src >pwd (should tell you you are in /usr/src)

Check that the rtai archive is there:

>ls rtai*

Now unpack it:

>tar -xvzf rtai-24.1.11.tgz.

This will create a directory labelled rtai-24.1.11. ăCheck that it is there:

>ls rtai*

c) Apply the patch to the source code.

First we need to move into the linux-2.4.20 directory:

>cd linux-2.4.20

Now we patch the code. ăIssue the following command from within the linux-2.4.20 directory:

patch =p1 < /usr/src/rtai-24.1.11/patches/patch-2.4.20-rthal5g

Watch the output and make sure there are no error messages present.

NOTE: There is a README.INSTALL file in the rtai directory (which we left back in /usr/src...) ăIt speaks of patching the kernel with both rthal and adeos. ăAs far as I know at this point adeos is not required to run EMC. ăI did patch the kernel with both initially but experienced build errors due to what looked like patch errors. ăI applied the pathes in the order listed in the INSTALL doc, ie rthal and then adeos. ăYour mileage may vary.

Your kernel source code is now patched with the necessary RTAI patch to run EMC. ăYou need to build it in order to run it. ă

d) Build yourself another kernel, this time it will be the real time kernel.

The patching process applied the real time code changes to the kernel source code. ăYou've already built a kernel before, so it should be a breeze this time. ăFollow all the steps you used before, starting at "make mr-proper". ăDon't miss any ! ăNote that when it comes time to configure the kernel you've now got 2 choices for a starting config file to import: 2.4.18-14 or the config file for the 2.4.20 you built last time. ăRemember to set the PPP settings to "kernel" if you use the 2.4.18-14 file. ăThey should already be in the 2.4.20 config file because you selected them when you built the last kernel. ă

e) Reboot your computer, selecting the new "real time" 2.4.20 kernel.

Note that Grub automatically boots the first kernel in the Grub file. You may want to edit grub.config to make the run time kernel the first kernel for the file. ăThis would be especially helpful if you've got employees running the computer or there is a power outage. ăAny accidental reboot without someone knowingly selecting the "real time" kernel will result in the default (ie kernel "0") being run and EMC isn't going to run right with a regular kernel. ăParts could be wasted and machine tools damaged until this is figured out. ăTo edit grub.config, do the following:

Change directories to /boot/grub:

>cd /boot/grub

Open grub.config in xemacs:

>xemacs grub.config

Make your changes and save. ăIe copy and paste the real time kernel to be the first kernel, ie "0". ăThe real time kernel will now automatically boot every time. ă ă

Conversely, one could edit the "default=0" line to default to the right kernel. ăThe first kernel is 0, the second is "1", etc. ă

2.1.4 Step 3: Build the Kernel With the Runtime Patch.

Building the kernel with the runtime patch is *exactly* the same as building the previous kernel, EXCEPT:

a) when config is run (ie xconfig or menuconfig) the following changes need to be made to the configuration:

i) In the "Processor Type and Features" options, you need to enable "Config_RTHAL", ie change it to "y".

ii) In the "Loadable Module Support" options, you need to disable "Set version information on all module symbols". ă Otherwise the build is exactly the same. ăFollow all the steps listed above and you should have no problems. Reboot and run your new kernel. ăNote that you might have to select your new kernel when grub asks for your selection during boot.

NOTE: the RTAI README.INSTALL doc mentions that the Advanced Power Management options should be disabled in the kernel. ăI elected not to do that. ăObviously having a PC go into sleep mode or having the hard drive wind down while executing a real time application is not a good thing. ăI thus take this one step further and disable all APM functions in the BIOS of the PC, rather than mess around with kernel options.[3]

Stay tuned !

2.2 Linux Install

One of the old work horse distributions for EMC is the Red Hat 5.2 distribution. While you may have some trouble finding it, this distribution was well worth the search if what you want to do is make chips with your converted machine tool. Support for this distribution will be dropped from the EMC system before to long.

[3] Kim Lux <lux@diesel-research.com>

Dave Anderson had a number of copies of a public domain version at NAMES in 2000. You might ask if he has them still. He also had a collection of EMC info on a second CDROM. You could ask if he has either of these available by posting to CAD_CAM_EDM_DRO@yahoo.com.

There are a couple of ZipEmc distributions that have been made. These use a windows file system so that you do not have to re format an existing ms windows machine. If you are interested in one of these post a request to emc@nist.gov and someone should answer with a current source for these CD's.

The newer releases of Linux produce mixed results. A large part of this seems to stem from the specific compiler and glibc that is shipped with the distribution. There are several pages in the handbook that were written by Ray Henry that detail the installation of real time and EMC while using a Mandrake 7.0 distribution. You should look for these under rtlinux and EMC.

2.2.1 Real-Time Linux Install

EMC requires that the Linux kernel be altered so that it has a Real Time patch running within it. This is what makes the machine control run reliably and consistently.

The Real Time patches, prepatched kernels, and modules are available from rtlinux.org `http://www.rtlinux.org`.

Installing Real Time Linux (RTLinux) seems to be the toughest part to doing an EMC installation. With that said, here are some pages and links designed to help with this.

Tim Goldstein has the essential process for installing RTLinux on RedHat 5.2 `http://206.19.206.56/installemc.htm` here. If you are looking for the most reliable and easiest to use EMC installation, this is the place to go. This system has been running machines that make production parts for more than a year now.

Ray Henry has written a detailed page on installing RTLinux-2.0 and RTLinux-2.2a on the 2.2.2 distribution. Some have found this helpful for other distributions as well.

Andrew followed the Mandrake install and took notes on his install of rtlinux-2.2 and EMC with a Red Hat 6.1 distribution.

A very recent innovation by the rtlinux organization is the building of RPM packages for some releases of the realtime kernel. These packages install just like any other RPM. They do not provide the source code needed if you will be compiling your own EMC or doing other real time programming that requires compiling.

Good luck!

2.2.2 rtl2.0 and rtl2.2 install

Realtime linux install using Mandrake 7.0 as a basis.

2.2.2.1 Introduction

I have updated this install page to include rtlinux-2.0, rtlinux-2.2, and rtlinux-2.2a files. Where there are differences they will be noted by a head-

ing that will look something like Note - 2.0.

This guide is based on an installed Linux Mandrake 7.0 that I purchased at Borders. It is a Macmillan published version and has a 3 cdrom set and uses the Linux 2.2.14 kernel with minor modifications by Mandrake. When I installed the Linux to a new hard drive, I let mandrake set up the disk partitions but renamed the smaller one / and the bigger one /usr. I also asked that the install include the development libraries.

Mandrake comes with several pre-compiled kernels and the modules to support them. A tar of the linux kernel is not ordinarily installed on the hard drive.

It is my understanding from several installers that this page will also help with an install using the Red Hat 6.2 distribution. There will be different windows on the screen and some differences in appearance.

Throughout this page I will use dark green letters to indicate the lines of text that I entered into the console konsole. I will use dark red to indicate the response of the computer to my command. Medium red lines are the focus on the discussion.ă For the sake of brevity, I have edited many of the responses of the computer when it was asked to compile large numbers of files. Some of these lengthy responses are included as files andă are linked to this document so that you can compare your results.

2.2.2.2 Download and tar

Download a clean prepatched kernel from rtlinux.org `http://www.rtlinux.org` rtlinux-2.0 is intended to be used with Linux 2.2.13 rtlinux-2.2 is intended to be used with Linux 2.2.14 rtlinux -2.2a is also intended to be used with Linux 2.2.14 Sometimes my browser wants to show a file rather than save it so I place the cursor over the file that I want to download and click the right mouse button rather than the left. This brings up a window that lets me name the file to download.

I placed this file in the home directory of root.ăWhen the download was complete I looked at the directory.

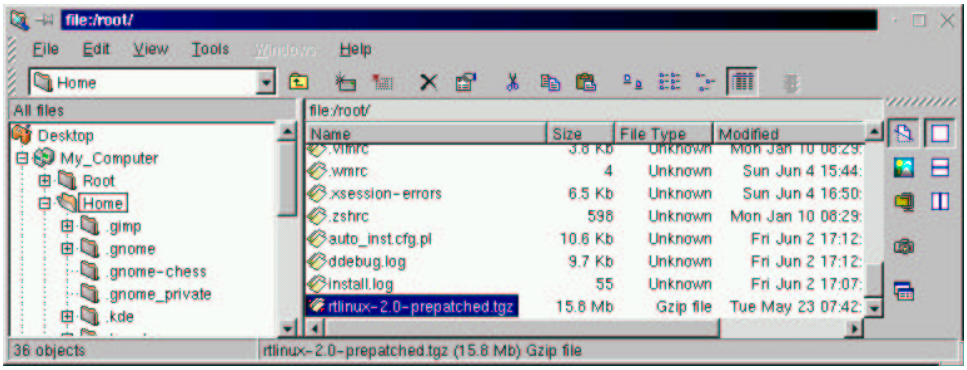
```
ls -l /root/rt*
-rw-r--r-- 1 root  root  16617862 May 16 14:59 rtlinux-2.2a-prepatched.tgz
```

That looks OK so I want to untar it into /usr/src

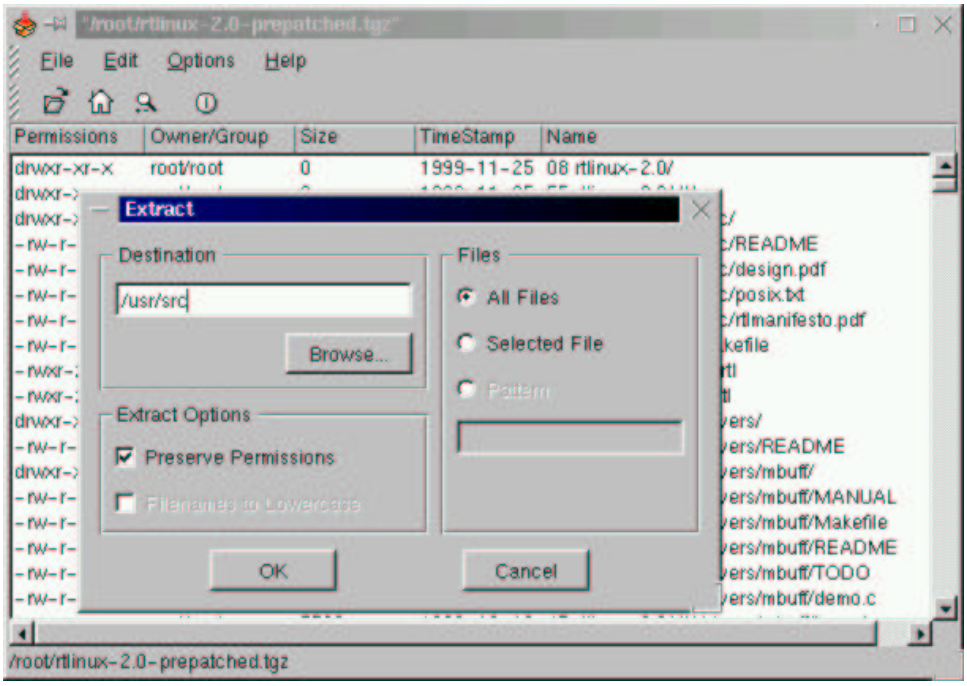
```
tar xzvf /root/rtlinux-2.2a-prepatched.tar.gz /usr/src
```

Note 2.2/2.2a - Both use the same rtlinux-2.2 directory when taken out of the tar ball.

A whole bunch of stuff goes across the screen indicating all of the files that tar is placing in /usr/src. This creates a new directory /usr/src/rtlinux-2.2 and sets up the new kernel structure below it. [install/images/expl01.gif]



If you are uncomfortable with using command line stuff like tar, you can use the kde file system explorer and archiver. You will need to start x-windows and use the kde windows manager for your system to look anything like the pictures included above. **(Note shows 2.0)** [install/images/archive.gif]



Now if you double left click on the little file symbol to the left of the rtlinux...tgz file name, explorer will open that file with the kde archiver. This window allows you to see all of the files that are in the tgz file.

You can select the extract to item from the file menu or click on the little magnifying glass icon and archiver brings up the extract window. Enter the destination for the extraction /usr/src and click OK and in a minute or so the job will be done.

2.2.2.3 The linux link

Some of the compiler operations that you are about to invoke with make, require that the linux kernel stuff be found in the /usr/src/linux directory. Your new Mandrake system may or may not have such a directory. You can find out using the explorer shown above or you can use the console. If you use the explorer, you may have some difficulty removing a link by deleting it. I found some kind of conflict that developed so I'll only show the console procedures below.

The konsole window allows you to repeat commands by pressing the up arrow until you get to the command that you wish to enter again. This feature makes itä so that you do not have to type all commands. You can just find the one you want and repeat it by pressing enter or return. (note that in the directory command ls -l that l is lowercase L)

```
[root@localhost /root]# cd /usr/src
[root@localhost src]# ls -l
total 12
drwxr-xr-x   7 root     root         4096 May 16 06:47 RPM/
lrwxrwxrwx   1 root     root           12 Jun 4 17:16 linux -> linux-2.2.14/
drwxr-xr-x  20 root     root         4096 May 16 10:42 linux-2.2.14/
drwxr-xr-x   4 root     root         4096 Nov 25 1999 rtlinux-2.2/
```

I see from the fifth line that - linux ->linux-2.2.14. So the linux file in this directory is a link. Since it is only a link I can delete it and then check to make sure it is gone.

```
[root@localhost src]# rm linux
rm:  remove ‘linux’?  y
[root@localhost src]# ls -l
total 12
drwxr-xr-x   7 root    root        4096 May 16 06:47 RPM/
drwxr-xr-x  20 root    root        4096 May 16 10:42 linux-2.2.14/
drwxr-xr-x   4 root    root        4096 Nov 25  1999 rtlinux-2.2/
```

Now if it is a real directory with a bunch of sub directories, I'd need to move it to some other location. I'd do that by first seeing that it was a directory rather than a link to a directory.

```
[root@localhost src]# ls -l
total 16
drwxr-xr-x   7 root    root        4096 May 16 06:47 RPM/
drwxr-xr-x   3 root    root        4096 Jun  4 17:36 linux/
drwxr-xr-x  20 root    root        4096 May 16 10:42 linux-2.2.14/
drwxr-xr-x   4 root    root        4096 Nov 25  1999 rtlinux-2.2/
```

I see from the light red line that linux is a common directory – its line begins with d rather than l and it just has a / after its name. I need to move this linux directory out of the way before I compile a new kernel.ăI'll called it linuxold. The mv command with the linux name and linuxold as the new name, doesăthe deed for me.

```
[root@localhost src]# mv linux linuxold
[root@localhost src]# ls -l
total 16
drwxr-xr-x    7 root     root        4096 May 16 06:47 RPM/
drwxr-xr-x   20 root     root        4096 May 16 10:42 linux-2.2.14/
drwxr-xr-x    3 root     root        4096 Jun  4 17:36 linuxold/
drwxr-xr-x    4 root     root        4096 Nov 25  1999 rtlinux-2.2/
```

Now you need to make a new link in place of the old /usr/src/linux

```
[root@localhost src]# ln -s rtlinux-2.2/linux linux
[root@localhost src]# ls -l
total 16
drwxr-xr-x   7 root     root        4096 May 16 06:47 RPM/
lrwxrwxrwx   1 root     root          17 Jun  4 17:49 linux -> rtlinux-2.2/linux/
drwxr-xr-x  20 root     root        4096 May 16 10:42 linux-2.2.14/
drwxr-xr-x   3 root     root        4096 Jun  4 17:36 linuxold/
drwxr-xr-x   4 root     root        4096 Nov 25  1999 rtlinux-2.2/
```

Make certain that the link goes from /usr/src/linux to the realtime/linux. Not the other way round. With this done we are ready to get down to work.

2.2.2.4 Configuring the kernel

This is a part of the process that mystifies me still. When I first installed the real time patch to Red Hat 5.2 and used Tim's install script, www.ktmarketing.com `http://www.ktmarketing.com` I never really appreciated all of the work that went into setting up that config file. I just accepted his file and went on.

You can do essentially the same thing with the new kernel (not with his RH5.2 file) but you will wind up without a lot of the things that you may want and some things that you will need.ă All of the config choices will present you with a list of default options that you can accept, but that list is limited in what it's kernel will do for you.

The first thing that I need to do is look at what is running on my system right now.

```
[root@localhost src]# /sbin/lsmod
Module                  Size  Used by
nls_cp437               3580   1  (autoclean)
vfat                   11004   1  (autoclean)
fat                    32640   1  (autoclean) [vfat]
```

```
nls_iso8859-1         2052   2  (autoclean)
sound                64184   0  (autoclean) (unused)
soundcore             3524   3  (autoclean) [sound]
soundlow               300   0  (autoclean) [sound]
lockd                33256   1  (autoclean)
sunrpc               56612   1  (autoclean) [lockd]
slhc                  4392   0
supermount           14880   2  (autoclean)
nfs                  31832   1  (autoclean)
```

Now, I warn you that the list of modules running on your system may be very different from those running on mine. But at least I have a start at knowing what needs to be included when I run make oldconfig, make config, make menuconfig, or make xconfig. You will want to be certain that each of the modules returned by the /sbin/lsmod command is included in your .config file.

It also took me a while to discover that the exact name of the module is included in the help files that are available when you run make menuconfig or make xconfig. There is a lot of confusion that can develop between the all caps variable names and the actual module name.

I have included a few lines of a .config file below so that you can see how one is setup.

This is cryptic stuff. Each parameter is given a name by someone far away. Those names are all uppercase and where multiple words are used the space between is underlined. Parameters that are not used are commented # out. Parameters that are used have a y or an m after the equals sign. A y means that the parameter will be included directly in the kernel when it is compiled. An m means that the parameter will be compiled as a module and will be installed in the kernel whenever it is needed and removed when it is not needed.

```
#
# Automatically generated make config:  don't edit
#
# Code maturity level options
#
CONFIG_EXPERIMENTAL=y
# Processor type and features
#
# CONFIG_M386 is not set
# CONFIG_M486 is not set
CONFIG_M586=y
# CONFIG_M586TSC is not set
# CONFIG_M686 is not set
CONFIG_X86_WP_WORKS_OK=y
CONFIG_X86_INVLPG=y
CONFIG_X86_BSWAP=y
CONFIG_X86_POPAD_OK=y
CONFIG_1GB=y
# CONFIG_2GB is not set
# CONFIG_MATH_EMULATION is not set
CONFIG_MTRR=y
# CONFIG_SMP is not set
```

"So what?" You say. Well the heart of this config thing is that each of the modules that was listed when I entered /sbin/lsmod needs to be included when you go through your config setup. But the modules listed are not the only way that the kernel is configured. There are also a bunch of parameters that are compiled right into the kernel itself.

So being a "quick and dirty" kind of installer, I could just copy the .config file from the old kernel structure to the new. But that will leave out essential information that is a part of the real time kernel. And as the first line in the file above says "# Automatically generated make config: don't edit." It is my understanding that you will need to go through the entire process of making a new .config in order to be certain that you have all of

the essential stuff. (If you know better, let me know and I'll revise this.)

So let's compare the four methods we can use to manually set up a config file and then pick one or more and do it. These will be brief descriptions with visuals. I'm going to make sure that I am in the correct directory before I enter the config command.

```
[root@localhost linux]# pwd
/usr/src/rtlinux-2.0/linux
```

Yes, I am in the correct location for the configuration process. So I need to select the method that I will use. The possibilities again are:

- make oldconfig
- make config
- make menuconfig
- make xconfig

This is what each looks like.

- **make oldconfig**

```
[root@localhost linux]# make config
rm -f include/asm
( cd include ; ln -sf asm-i386 asm)
/bin/sh scripts/Configure arch/i386/config.in
#
# Using defaults found in arch/i386/defconfig
#
*
* Code maturity level options
*
Prompt for development and/or incomplete code/drivers (CONFIG_EXPERIMENTAL) [N/y/?]
```

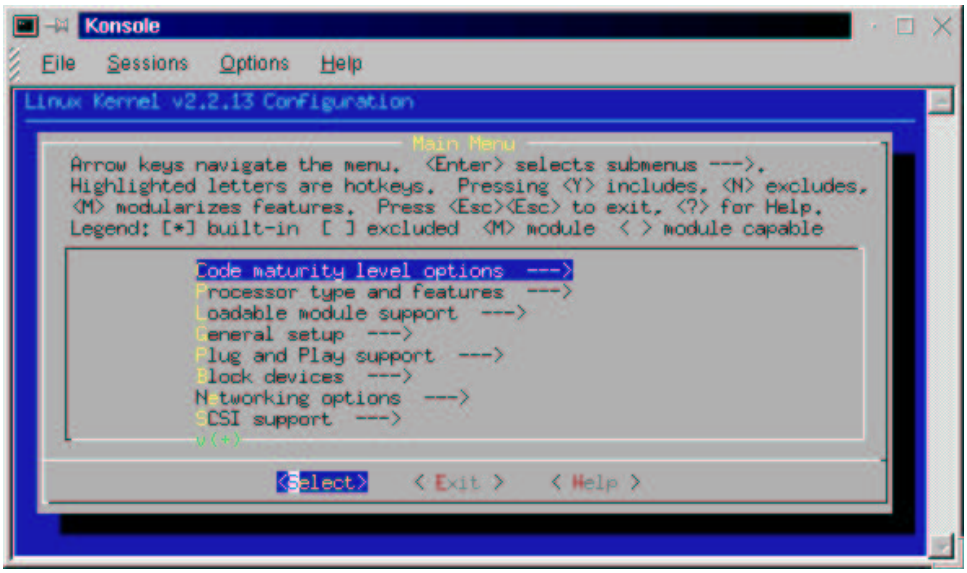
This is quick and is done directly in the console window. There is no help and no description beyond the parameter name. It is also linear – it runs from the first entry in "`Using defaults found in arch/i386/defconfig`" to the last one. Then it writes the file and suggests your next step.

- make menuconfig

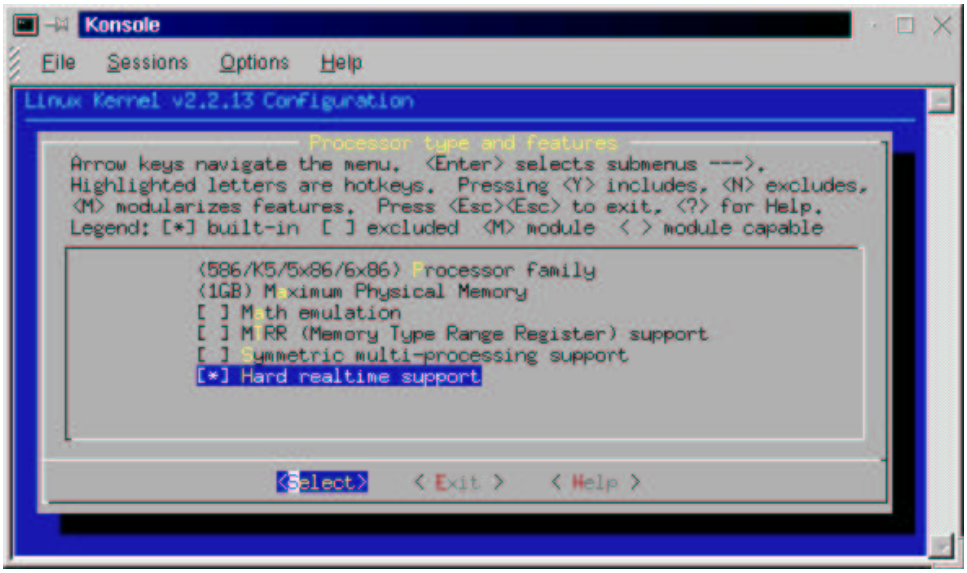
The second way of making a config file is menu oriented. To start this method you can just enter make menuconfig at the prompt.

```
[root@localhost linux]# make menuconfig
rm -f include/asm
( cd include ; ln -sf asm-i386 asm)
make -C scripts/lxdialog all
make[1]:  Entering directory '/usr/src/rtlinux-2.2/linux/scripts/lxdialog'
..
```

After a bit of checking and compiling, the following window shows up. [install/images/konsol02.gif]



This seems to be a little more friendly. It also allows you to skip around among the many sections of the config file and repeat your work in each section. It may take a bit of reading to figure out how to move around within the menu system and what to do next. The following screen shows the processor type and features menu. [install/images/konsol03.gif]



You can navigate around the menu choices with the up and down arrow keys. You can also navigate across the select, exit, and help items with the left and right arrow keys. Whenever a parameter is highlighted, you can select it with y, deselect it with n, or make it a module with m. The help files are very helpful here.

Chapter 3

Getting and setting up EMC software

3.1 Introduction

The third hurdle that you face when you begin to set up the EMC is getting and installing the EMC software itself. Much of the EMC and RCSLIB have been placed on SourceForge.net in a concurrent versioning repository and are available for download from that site. Along with that change there has been a change in the nature of the distributions.

Installation can be a daunting task to people new to Linux. The hardest part is getting the Real Time Linux patch up and running. After that, installing EMC is pretty easy. With that said, most people have found that the easiest way to install Real Time Linux and EMC is to go to Tim Goldstein's web page and use his install script for EMC on the linux 2.0.36 release. The information on this is under the download information link above.

Ray Henry has written a page on installation of EMC on his computer running Mandrake 7.0 with a prepatched rtlinux-2.2 kernel. Ray doesn't have an install script but the process is documented fairly well and he has some text files that you can view or download.

Brian Pitt has written a guide to installing Linux, Real-Time Linux, and EMC on a windows formatted hard drive using a Slackware installation. This looks like a reasonably easy way to try out the system. Look for installation CDROM's soon.

3.2 EMC Download Page

Introduction

You will find the most recent releases of the EMC on sourceforge.net in the EMC download area. The current releases of the EMC include a set of four files made by executing the packup file in the base directory. On my machines this is /usr/local/nist*. This command creates a master tar file that includes emc - the date of the packup - .tgz The current size of this file is around 3.5 to 4 meg.

There are three files that contain the same release but are broken into floppy sized chunks. These are indicated by the same release date but use .aaa .aab .aac. You can download these directly to an msdos floppy and put them together on your linux machine using Ray Henry's description below.

The last file to be included in the build is the .txt file which includes information about the release.

3.3 NIST Directory Description

This directory contains a series of EMC releases in various formats. The releases are dated according to the convention:

emc-DD-Mon-YYYY

where

DD is the two-digit day number, Mon is the three-letter month abbreviation, and YYYY is the four-digit year.

So, emc-03-Mar-1999 signifies the March 3, 1999 release. This date will be used in the examples below. There will be at least two sets of releases, possibly more. The most recent is the one you should download, unless you know otherwise.

The suffixes indicate the type of file:

.txt The release notes for the associated release. Read this first to decide if you want this release.

.tgz The full EMC release in compressed tar "tape archive" format.

Unpack and install in the /usr/local/nist directory via:

```
cp emc-03-Mar-1999.tgz /usr/local/nist
cd /usr/local/nist
tar xzvf emc-03-Mar-1999.tgz
```

.aa* The full .tgz release is several megabytes, currently about 4Mb. This is too big to fit on a floppy. If your Linux machine is not on the network, you can copy each of these files onto a floppy (they just fit), copy them onto your Linux box, and concatenate them together to form the full release, like this:

```
cat emc-03-Mar-1999.aa* > emc-03-Mar-1999.tgz
```

The .aaa file is the first part, the .aab is next, etc. The * wildcard will match alphabetically so this short form will expand the file names in the right order.

3.4 Links

Here are a few links to help you get the latest EMC release. The first two are located on a server at NIST itself:

For the Linux 2.0.36 kernel:
ftp://ftp.isd.mel.nist.gov/pub/emc/emcsoft/linux_2_0_36

For the Linux 2.2.13 kernel:
ftp://ftp.isd.mel.nist.gov/pub/emc/emcsoft/linux_2_2_13

This next link is located in the linuxcnc.org dropbox. I set this up in case the NIST server is down:
http://www.linuxcnc.org/dropbox/emc-31-Jan-2000.tgz

Here is a text file containing release notes on the January version version: http://www.linuxcnc.org/dropbox/emc-31-Jan-2000.txt

3.5 Ray's three disk download procedure.

Once I have the three EMC files on disk, the next thing I'd do is make a directory called /usr/local/nist on the Linux machine.

```
mkdir /usr/local/nist
```

Now move into that directory.

```
cd /usr/local/nist
```

The prompt should show nist on the right end. You can stay in this directory and copy files to it. Insert the first floppy into your machine.

```
mount -t msdos /dev/fd0 /mnt/floppy
```

After a second of looking at your floppy it should return your screen to your root prompt. This mount command will allow you to work with floppy files in your /mnt/floppy directory. If you have already defined /dev/fd0 as having the msdos file type you may be able to skip the -t msdos part of the above command.

```
cp /mnt/floppy/* /usr/local/nist
```

After a bit of floppy disk grinding you should get the prompt again. Check to see that the file was copied.

```
ls
```

You should see a list of the files in the nist directory. At this point there will be just one emc-**-***-1999.aa* if you copied it.

```
umount /dev/fd0
```

After a second you should get the prompt again. If you get a "can't umount message" it is probably because umount will not work when you are in the /mnt/floppy directory.

When umount is successful you'll need to repeat the mount, copy, ls, and umount commands for each of the disks. You don't need to type in each command every time, at the prompt, he up arrow will scroll you back to the previous commands. Just up arrow to the one you want and [enter].

Now that you've got a directory listing all of the emc floppy files in it, all you have to do is put them together. You can do this with the cat command. It will be easiest if your prompt and the files are in your nist directory.

cd /usr/local/nist
Type in:
cat emc > emc-dd-Mmm-yyyy.tgz*

(you should replace the dd-=Mmm_yyyy with the proper release numbers.) After a bit of work you should see the prompt again.

ls -l

Those are lower case L's not one's. The big tar file should be listed as a 3+ meg file with the .tgz extension.

3.6 INSTALLING THE NIST EMC SOFTWARE

The NIST EMC software can be installed on Linux PCs, Sun Solaris boxes, or most any Unix machine. Currently the distribution is set up for Linux.

0. For LINUX installations only: Download and install the New Mexico Tech RT-Linux, real-time patches and modules. If you have a system using a 2.0.x kernel get the kernel sources for 2.0.36 and RT-Linux patches and modules version 09J. If you have a 2.2.x kernel use RT-Linux 2.0 and kernel 2.2.13. The kernel sources are available from http://www.kernel.org. The New Mexico Tech RT-Linux patches and modules are available from http://www.rtlinux.org.

0.a: Follow the instructions from www.rtlinux.org for installing the RT-Linux, however I have the following additional recommendations. - For RedHat Linux 5.2 users, check out Tim Goldstein's installation instructions at
http://206.19.206.56/installemc.htm - Get the archive with the pre-patched kernel sources rather than patching your own. - Before untarring the archive move any existing directory or link /usr/src/linux to a different name. - Untar the archive in the directory /usr/src. - After untarring the archive create a symbolic link so that /usr/src/linux points to the linux directory that was included in the archive. For example cd /usr/src ln -s rtlinux-2.0/linux linux

- When you configure the kernel, under Loadable Module Support say Yes to set version info and to enable the Kernel module loader, and under Processor type say no to Symmetric Multi-processing support unless you actually have a multi-processor system. - Test the examples in the rtl directory, before trying to install EMC. Make sure the min and max values out of the measurements example are in the thousands and not in the millions.

1. Download the appropriate .tgz file from either the subdirectory for your platform. Read the README file within those subdirectories for the details. The linux_2_0_36 directory contains the code/binaries tested under linux kernel version 2.0.36 with RedHat 5.2 and the New Mexico Tech RT-Linux patch 09J. Th linux_2_2_13 directory contains the code/binaries tested under linux kernel version 2.2.13 with RedHat 6.1 and the New Mexico Tech RT-Linux patch 2.0. All the distributions contain the NIST Real-time Control System (RCS) header files and linkable library, EMC source code, and other miscellaneous files. The RCS source code is not included in this archive, since it would make it larger, but is available at http://www.isd.mel.nist.gov/projects/rcs_lib.

2. Log in to your Linux PC. Become root.

3. Copy the EMC distribution archive from wherever you downloaded it to into /usr/local, cd there, and unpack the archive. This looks like:

```
root> cp emc-<date>.tgz /usr/local root> cd /usr/local root> tar xzvf emc-<date>.tgz
```

4. You should see the following files in /usr/local:

readme – more instructions
packup – a script for making the .gz files below
putback – a script for checking the code back in to NIST
install – a script for installing and compiling this software
compile – a script for just compiling this software
clean – a script for removing EMC/RCS software (not alltar.gz)
rcslib.inc – a list of files used by packup to build the RCS archive
rcslib.exc – ones not to include
emc.inc – a list of files used by packup to build the EMC archive
emc.exc – ones not to include
floppy.inc – a list of these files, for building the floppy

skel.tgz – the gzip archive for the directory skeleton
rcslib.tgz – the gzip RCS archive
emc.tgz – the gzip EMC archive
linux.tgz – the full archive

5. Run the 'install' script:

```
root> ./install
```

The install script creates the directory skeleton for emc/plat; unzips the RCS code; and unzips the EMC code and compiles it.

6. For linux_2_0_36 only: You will need to setup shared memory. The linux 2.2.13 distribution of EMC uses mbuff which is a more flexible and automatic method than the linux 2.0.36 described below.

Edit the file lilo.conf to add the line indicated. You should set the mem value to 1 Mb less than the total amount of memory you have in your system.

```
image=/boot/vmlinuz label=linux root=/dev/hda3 append="mem=31m"
```
<– add this read-only

For a 64 MB machine, the "31" would be "63", etc.

Run lilo, as root:

```
root> lilo
```

and reboot.

7. Run the Controller in simulation.

To run the EMC code on a real machine, you need to set up all the hardware, create the wrappers or use ones provided, configure system parameters via the INI files, and code up the discrete I/O controller or use one provided. However, you can just run a simulation "out of the box."

This is done by running the bash script, emc.run, from the top-level emc directory.

The script will run several programs in the background, and then start a graphical user interface. There are several different interfaces that can be configured, but the current default tkemc is shown in tkemc.gif in this directory.

Before you can do anything you will need to come out of E-STOP (press the <F1> key) and turn the machine on (press the <F2> key).

You can then jog the axis using the arrow keys and page up/page down. Press F4 to go to Auto mode where you can run rs274ngc G-code part programs. The example programs are in the programs subdirectory. cds.ngc is a standard circle-diamond-square program.

You may want to run tkbackplot or emcplot3d (not yet available) to better visualize what is going on.

8. Trouble Shooting.

9. Editing and Recompiling

You may need to recompile the EMC code for a number of reasons. You may have modified the source code, or you may have downloaded just a few new files. To recompile, do the following:

```
root> cd /usr/local
root> ./compile
```

If you want to compile code in any of the subdirectories individually, cd to the EMC source directory and run individual makes:

```
root> cd /usr/local/emc/src
root> make PLAT=linux clean depend install
```

Chapter 4

Hardware – Parallel Port

Interfacing EMC to the real world

4.1 Parallel Port Connection

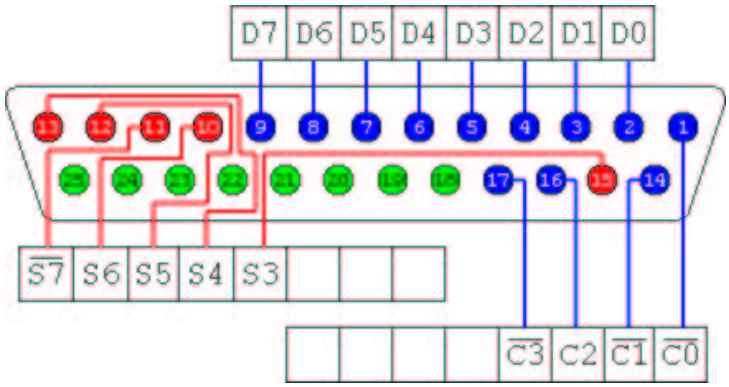


Diagram of the 25 pin connector used on most computers for connecting to the printer. Refer to the following subsection for pin assignments when using EMC.

4.1.1 Setting up Stepper Motors

Currently the EMC supports stepper motors with a 2-bit step-and-direction interface, with bits mapped to the parallel port. Each parallel port has 12 bits of output and 5 bits of input. The outputs are used to drive the step and direction of each motor. 12 bits of output mean that up to 6 stepper motors can be controlled. The inputs can be used to detect limit or home switch trips. 5 bits of input mean that only one axes can get full positive, negative, and home switch inputs. The EMC mapping compromises for 3 axes of stepper motor control, with all positive limit switches being mapped to one input, all negative limit switches being mapped to another input, and all home switches being mapped to a third input. Other permutations are possible, of course, and can be changed in the software. You could also add 2 additional parallel ports (LPT2, LPT3), and get 36 bits of output and 15 bits of input. Some parallel ports also let you take 4 outputs

and use them as inputs, for 8 outputs and 9 inputs for each parallel port. This would let you get 3 axes of control and full switch input per parallel port. See Using the PC parallel port for digital I/O for more information on the parallel port.

The pin outs for the EMC stepper motor interface using the first parallel port
Only Step/Dir function is available with steppermod, whilst all options can be used with freqmod and smdromod.

The bracketed numbers for the Two Phase function refer to the connections on a Bridgeport BOSS driver.

IO Pin	Function Step/Dir	Function Quadrature	Function Two phase	Function Four phase
D0, pin 2	X direction	X CW	X Phase 0 (P19)	X Phase 0
D1, pin 3	X clock	X CCW	X Phase 1 (P13)	X Phase 1
D2, pin 4	Y direction	Y CW	Y Phase 0 (P19)	X Phase 2
D3, pin 5	Y clock	Y CCW	Y Phase 1 (P13)	X Phase 3
D4, pin 6	Z direction	Z CW	Z Phase 0 (P19)	Y Phase 0
D5, pin 7	Z clock	Z CCW	Z Phase 1 (P13)	Y Phase 1
D6, pin 8	A direction	A CW	A Phase 0 (P19)	Y Phase 2
D7, pin 9	A clock	A CCW	A Phase 1 (P13)	Y Phase 3
C0, pin 1	B direction	B CW	B Phase 0 (P19)	Z Phase 0
C1, pin 14	B clock	B CCW	B Phase 1 (P13)	Z Phase 1
C2, pin 16	C direction	C CW	C Phase 0 (P19)	Z Phase 2
C3, pin 17	C clock	C CCW	C Phase 1 (P13)	Z Phase 3
S3, pin 15	X/Y/Z/ lim +	X/Y/Z/ lim +	X/Y/Z/ lim +	X/Y/Z/ lim +
S4, pin 13	X/Y/Z/ lim -	X/Y/Z/ lim -	X/Y/Z/ lim -	X/Y/Z/ lim -
S5, pin 12	X/Y/Z/ home	X/Y/Z/ home	X/Y/Z/ home	X/Y/Z/ home
S6, pin 11	Probe	Probe	Probe	Probe
S7, pin 10	spare	spare	spare	spare

Pin out for the second parallel port - Used with bridgeportio.
IO assignments can be changed within the ini file.

IO Pin	Function
D0, pin 2	Spindle reverse
D1, pin 3	Spindle Forward
D2, pin 4	spare
D3, pin 5	Spindle on
D4, pin 6	spare
D5, pin 7	spare
D6, pin 8	Mist Coolant
D7, pin 9	Flood coolant
C0, pin 1	Speed decrease
C1, pin 14	Speed increase
C2, pin 16	Estop output
C3, pin 17	Spindle brake
S3, pin 15	spare
S4, pin 13	Estop input
S5, pin 12	Lube input
S6, pin 11	spare
S7, pin 10	Spare

Stepper motor control is implemented using a second real-time task that

runs at 100 microseconds. This task writes the parallel port output with bits set or cleared based on whether the a pulse should be raised or lowered. This gives an effective period of 200 microseconds for a full up-and-down pulse, or a 5 kilohertz frequency.

Chapter 5

PICO Parallel Port Motion Control

For machines that use servo motors, one option is to use the servo motor interface card from Servo To Go Inc http://www.picosystems.com. This company makes a set of cards that have been integrated into the EMC and have been used in several installations. Here we attempt to document matters relating to this set of cards. The organization is FAQ-like at this point, addressing specific questions or aspects of using the PPAC.

Chapter 6

Servo To Go

For machines that use servo motors, one option is to use the servo motor interface card from Servo To Go Inc `http://servotogo.com`. It has been used in several installations, and the driver code is included with the EMC distribution. Here we attempt to document matters relating to this card. The organization is FAQ-like at this point, addressing specific questions or aspects of using the Servo To Go card.

./src/emcmot/stg.c ./src/emcmot/stg.h	Model 1 produces stg.o (4 axes) with -DSTG_8_AXES produces stg8.o (8 axes)
./src/emcmot/stg2.c ./src/emcmot/stg2.h	For Model 2 4 axis default (use -DSTG_8_AXES for 8 axes, 4 or 8 are currently your only choices).
./src/emcmot/stg_v2_axis8.c	same as above but does not have FIND_STG_BASE_ADDRESS variable and associated code. May be obsolete since there are some typos in comments that are corrected in stg2.c
./src/emcmot/stgdiag.c	diagnostics program a stand-alone console (command line) program to test card functions note that a similar program is available on the Servo To Go web site.
./src/emcmot/extstgmot.c	dispatcher/wrapper implements functions described in extintf.h by calling functions in stg.c or stg2.c
./plat/nonrealtime/bin/stg2diag	executable from stgdiag.c Model 2 Use from command line.
./plat/nonrealtime/bin/stg2mot	non-real-time from stg2.o Model 2 4 axes.
./plat/nonrealtime/bin/stgdiag	executable from stgdiag.c Model 1 Use from command line.
./plat/nonrealtime/bin/stgmot	non-realtime from stg.o Model 1 4 axes.
./plat/nonrealtime/lib/stg.o	from stg.c intermediate file
./plat/nonrealtime/lib/stg2.o	from stg2.c intermediate file
./plat/nonrealtime/lib/stg8.o	from stg.c with STG_8_AXES flag Model 1.
./plat/nonrealtime/lib/stgdiag.o	file used to make the executable in the bin directory.
./plat/realtime/lib/stg.o	from stg.c intermediate file
./plat/realtime/lib/stg2.o	from stg2.c intermediate file
./plat/realtime/lib/stg2mod.o	from stg2.o Model 2 4 axes
./plat/realtime/lib/stg8.o	from stg.c with -DSTG_8_AXES Model 1 8 axes
./plat/realtime/lib/stg8mod.o	installable kernel module from stg8.o Model 1 8 axis
./plat/realtime/lib/ stg_v2_8axis_mod.o	from stg2.o (not stg_v2_8axis.o) same as stg2mod.o (no, I don't know why)
./plat/realtime/lib/stgmod.o	from stg.o Model 1 4 axis

6.1 Connecting to the card

Here is the connections to connect EMC to a machine using the Servo To Go card. This is quoted from stg2.c.

DIGITAL INPUTS		
Index	Function	Connector/Pin
00	X home switch	P1/47
01	X +limit switch	P1/45
02	X -limit switch	P1/43
03	X amp fault	P1/41
04	Y home switch	P1/39
05	Y +limit switch	P1/37
06	Y -limit switch	P1/35
07	Y amp fault	P1/33
08	Z home switch	P1/31
09	Z +limit switch	P1/29
10	Z -limit switch	P1/27
11	Z amp fault	P1/25
12	A home switch	P1/23
13	A +limit switch	P1/21
14	A -limit switch	P1/19
15	A amp fault	P1/17
16	B home switch	P2/31
17	B +limit switch	P2/29
18	B -limit switch	P2/27
19	B amp fault	P2/25
20	C home switch	P2/23
21	C +limit switch	P2/21
22	C -limit switch	P2/19
23	C amp fault	P2/17

DIGITAL OUTPUTS		
Index	Function	Connector/Pin
00	X amp enable	P1/15
01	Y amp enable	P1/13
02	Z amp enable	P1/11
03	A amp enable	P1/9
04	B amp enable	P1/7
05	C amp enable	P1/5
06	(none)	P1/3
07	(none)	P1/1

ANALOG INPUTS		
Index	Function	Connector/Pin
00	(none)	P2/2
01	(none)	P2/3
02	(none)	P2/5
03	(none)	P2/7
04	(none)	P2/9
05	(none)	P2/11
06	(none)	P2/13
07	(none)	P2/15

ANALOG OUTPUTS		
Index	Function	Connector/Pin
00	X amp ref	P3/2
01	Y amp ref	P3/8
02	Z amp ref	P3/5
03	A amp ref	P3/11
04	B amp ref	P4/2
05	C amp ref	P4/8
06	(none)	P4/5
07	(none)	P4/11

ENCODER INPUTS		
Axis	Function	Connector/Pin
X	A+	P3/14
X	A-	P3/15
X	B+	P3/17
X	B-	P3/18
X	I+	P3/20
X	I-	P3/21
Y	A+	P3/23
Y	A-	P3/24
Y	B+	P3/26
Y	B-	P3/27
Y	I+	P3/29
Y	I-	P3/30
Z	A+	P3/32
Z	A-	P3/33
Z	B+	P3/35
Z	B-	P3/36
Z	I+	P3/38
Z	I-	P3/39
A	A+	P3/41
A	A-	P3/42
A	B+	P3/44
A	B-	P3/45
A	I+	P3/47
A	I-	P3/48
B	A+	P4/14
B	A-	P4/15
B	B+	P4/17
B	B-	P4/18
B	I+	P4/20
B	I-	P4/21
C	A+	P4/23
C	A-	P4/24
C	B+	P4/26
C	B-	P4/27
C	I+	P4/29
C	I-	P4/30
(none)	A+	P4/32
(none)	A-	P4/33
(none)	B+	P4/35
(none)	B-	P4/36
(none)	I+	P4/38
(none)	I-	P4/39
(none)	A+	P4/41
(none)	A-	P4/42
(none)	B+	P4/44
(none)	B-	P4/45
(none)	I+	P4/47
(none)	I-	P4/48

6.2 Options in extstgmot.c

Your system may require changes to extstgmot.c. After making any changes you will need to recompile. Here are some of the options from the source code.

```
/* Uncomment any of the following declarations to inhibit
the use of home switches or index pulses when homing, or to
ignore limit switches if you don't have them. */

/* Uncomment NO_HOME_SWITCH if you want homing sequence
to
ignore the home switch, and only use the index pulse for hom-
ing.
This has the effect of making the home switches always appear
tripped. */
/* #define NO_HOME_SWITCH */

/* Uncomment NO_INDEX_PULSE if you want homing sequence
to
ignore the index pulse, and only use the home switch for hom-
ing.
This has the effect of making the index pulse always appear
present. */
/* #define NO_INDEX_PULSE */

/* Uncomment NO_LIMIT_SWITCH if you don't have limit switches.
This has the effect of making the positive and negative travel
limit switches never appear tripped. */
/* #define NO_LIMIT_SWITCH */
```

6.3 Where's the driver?

Or sometimes the question is "is there a Servo To Go driver for Linux in there? If so where is it?". EMC's modular design is quite elegant, although it makes this particular question hard to answer. Are stg.c (for Model 1) and stg2.c (for Model 2) the drivers? No, these are modules which customize the driver for the Servo To Go card when they are linked in. They implement functions that the driver uses.

Can I point to the drivers? Yes, there's stgmod.o and stg2mod.o (and some variants) for RT-Linux. They are installable kernel modules. Communication with the driver is typically by a shared memory area (the other option is to use a fifo). The driver commands are listed in emcmot.h.

6.4 What is the difference between Model 1 and 2?

The Model 2 has:

- index pulse circuit redesigned.
- external latch input

- more flexible watchdog timer
- more options for using spare timer

Reading encoders and writing to DACs happen to be unchanged between the Model 1 and 2. So if that is all you are doing, then it will work even if you use the wrong driver. However if you are using other functions on the board (digital I/O or ADCs) then you will need the correct driver.

6.5 stgdiag

Included in the EMC distribution is a diagnostics program to help verify the Servo To Go hardware. The source code for the program is typically located in:
/usr/local/emc/src/emcmot/stgdiag.c

There are two versions of the executable (depending on whether it is linked with stg.c or stg2.c). stgdiag is for the Model 1 card and stg2diag is for the Model 2. The executables are usually located in:
/usr/local/emc/plat/nonrealtime/bin/

To run the program, either navigate to the above directory, or have it on your path. The syntax is (using the Model 2 version as the example):

```
stg2diag [ dio | dac | limit | home | index <axis> | latch <axis>][ base_address
]
```

first set of options:

```
none        position for x, y, and z continuously updated, control-C to quit
dio         digital I/O, displays state of ports A, B and D
dac         digital to analog converter, prompts for axis, then prompts for voltage
limit       displays limit switches continuously, new line at each change, control-C to
stop
home        displays home switches continuously, new line at each change, control-C to
stop
index <axis> continuously display state of index pulse for given axis, control-C to stop
latch <axis> continuously display state of index pulse latch, control-C to stop
```

Default base address is 0x200 unless the base address is specified in the final argument.

lstg

There is another program available to help debug and verify the Servo To Go hardware functionality. It is available from the Servo To Go website in the Download Area. Like stgdiag, it is a console application. However lstg is more general, and has slightly more functions. Here is an example of its use:

```
[root@mclinux don]# lstg
Servo To Go Demo
```

Enter the letter corresponding to the desired command. You don't need to follow it with a carriage return. If the letter isn't a valid command, the list of commands will be repeated. To exit from a demo, type any character.
Base Address = 200
Board Model = Model 2
ServoToGo: [carriage return]
Commands:
o - Digital output demo, toggle bit
i - Digital input demo
D - DAC demo for chan 3, ramp
d - DAC demo, set voltage

r - Some random motion on chan 0
a - ADC demo
A - ADC autozero ON
n - ADC noise demo
N - ADC autozero OFF
e - Encoder demo
E - Zero Encoders
x - Index pulse demo
X - Index latch demo
l - External Latch Test
y - IRR test, continually reset index latch
1..8 - Set default Axis
q - Quit

Chapter 7

Configuring EMC

7.1 The INI file reference

Most of the information contained in this section is accurate and will remain so for long periods of time. There are occasional additions to this fill that creep into the versions in sourceforge that do on immediately get reflected in this document. There are even fewer variables that get changed so that they do not work as promised here.

Machine Configuration file emc.ini

The machine configuration file emc.ini follows the Microsoft INI file format, in which values are associated with keywords on single lines, perhaps in sections denoted with square brackets, e.g.,

[SECTION]
; COMMENT
SYMBOL = VALUE

Everything from the first non-whitespace character after the = up to the end of the line is passed as the value, so you can embed spaces in string symbols if you want to.

You can edit the values for each keyword in any text editor. The changes won't take effect until the next time the controller is run. The file should be called emc.ini, but the name can be overridden using a command line argument to the controllers. See the section "Starting Up" for information on how to do this. The following sections detail each section of the configuration file, using sample values for the configuration lines.

[EMC] section. The [EMC] section contains general parameters for the whole controller. These are:

VERSION = $Revision: 1.2 $

The version number for the INI file. This is automatically updated when using the Revision Control System, which looks for the $Revision: 1.2 $ string and appends the revision number. If you want to edit this manually just change the number and leave the other tags alone.

MACHINE = My Controller

This is the name of the controller, which is printed out when it runs. You can put whatever you want here.

NML_FILE = emc.nml

The name of the NML file to use.

DEBUG = 0x00000003

Sets the verbosity of the debugging messages printed out on the text console. Valid options are :

0x00000000 Do not print any debugging messages.

0x00000001 Print invalid messages

0x00000002 Print configuration settings

0x00000004 Print defaults

0x00000008 Print version

0x00000010 Print task messages

0x00000020 Print IO points

0x00000040 Print NML messages

0x00000080 Print time taken for motion to complete

0x00000100 Print interpreter debugging

0x00000200 Print RCS debugging

0x00000400 Print raw trajectory data

0x00000800 Print interpreter list

0x7FFFFFFF Print all debugging messages

The numbers can be combined (logical AND) so that selected information can be printed. i.e. 0x00000003 would print invalid messages AND configuration settings.

RS274NGC_STARTUP_CODE = G21 G90

A string of NC codes that the interpreter is initalised with.

[DISPLAY] section. The [DISPLAY] section contains parameters for the Graphical User Interface.

PLAT = nonrealtime

Only one choice, nonrealtime.

DISLAY = tkemc

The name of the user interface to use. Valid options are :

emcpanel

keystick

tkemc

tkemcts

xemc

yemc

CYCLE_TIME = 0.200

The period, in seconds, at which the display will be updated.

HELP_FILE = doc/help.txt

Location and name of a plain text help file to be used when the GUI HELP button is pressed.

POSITION_OFFSET = RELATIVE

Initial display setting for position, RELATIVE or MACHINE.

POSITION_FEEDBACK = ACTUAL

Initial display setting for position, COMMANDED or ACTUAL.

MAX_FEED_OVERRIDE = 1.2

Highest value that will be allowed for feedrate override. 1.0 = 100%

PROGRAM_PREFIX = programs/

The prefix to be added to any NC program prior to loading.

INTRO_GRAPHIC = emc.gif

File name of the image to be displayed whilst EMC is loaded. This can be commented out if not required.

INTRO_TIME = 5

Time, in seconds, that the intro image is displayed.

BALLOON_HELP = 1

Enable popup balloon help with tk based GUIs.

LINEAR_UNITS = AUTO

Units used for display of linear axis. AUTO, INCH, MM, or CM

ANGULAR_UNITS = AUTO

Units used for display of rotary axis. AUTO, DEG, RAD, or GRAD.

[TASK] section. The [TASK] section contains general parameters for EMCTASK, which includes primarily the NC language interpreter and the sequencing logic for sending commands to EMCMOT and EMCIO.

PLAT = nonrealtime

Only one choice, nonrealtime.

TASK = minimilltask

Name of the task controller program, bridgeporttask or minimilltask. This will need to correspond with IO controller program.

CYCLE_TIME = 0.010

The period, in seconds, at which EMCTASK will run. You can make this as small as you want to increase the throughput. Making it 0.0 or a negative number will tell EMCTASK not to sleep at all. Ultimately the system loading will limit the effective throughput.

[RS274NGC] section. Part program interpreter section.

PARAMETER_FILE = emc.var

Name of the file containing variables used for coordinate offsets.

[EMCMOT] section. The [EMCMOT] section contains critical parameters for the motion control module.

PLAT = realtime

Only one choice for most, realtime.

EMCMOT = steppermod.o

The motion control module to be used. Choices are :

steppermod.o The original stepper motor module using the parallel port. Not available with rtai systems.

freqmod.o An improved stepper motor module requiring PERIOD to be set.

smdromod.o freqmod with encoder feedback. Requires DRO_BASE_ADDRESS to be set.

stgmod.o Servo motor control using the Servo To Go card.

stg8mod.o Eight axis version of stgmod

stg2mod.o Servo motor control using the Servo To Go II card.

newstgmod.o Updated version of stgmod for both versions of the Servo To Go card.

vtiisamod.o Servo motor control using the ISA Vigilant Products card.

vtipcimod.o PCI version of vtiisamod. Only available for 2.4.xx based systems.

ppmcmod.o Jon Elson's parallel port motion control system for servo motors.

ppmcmod8.o Eight axis version of ppmcmod

univstepmod.o Stepper motor control via Jon Elson's parallel port system.

simmod.o Simulated motion control module.

minitetra.o Hexapod kinimatics using freqmod.

SHMEM_KEY = 100

Key used for shared memory.

SHMEM_BASE_ADDRESS = 0x3F00000

Base address of physical shared memory. For rtlinux_09J, this must point to memory reserved during boot up via "mem = 31M" in /etc/lilo.conf. This is not required for all later realtime linux systems as a different mechanism is used to allocate shared memory.

IO_BASE_ADDRESS = 0x378

Base address for the motion control card or parallel port.

PARPORT_BASE_ADDRESS = 0x378

Obsolete since July 2002 in the CVS sources, and BDI-2.2.18. Use IO_BASE_ADDRESS instead.

STG_BASE_ADDRESS = 0x200

Obsolete since July 2002 in the CVS sources, and BDI-2.2.18. Use IO_BASE_ADDRESS instead.

DRO_BASE_ADDRESS = 0x200

Base address for the Mauch/Kaluga DRO card used by smdromod for encoder feedback.

COMM_TIMEOUT= 1.0

Timeout for comms to emcmot in seconds.

COMM_WAIT = 0.010

Interval between tries of comms to emcmot in seconds.

PERIOD = 0.000020

Base period for output pulses with freqmod in seconds. The shorter the interval, the higher the maximum feedrate becomes, but at the expense of system throughput. Too smaller a period will cause the whole system to lock up.

Do NOT use this parameter with any module other than freqmod or smdromod. The system WILL lock up.

MOTION_IO_ADDRESS = 0x3BC

Port used for IO synchonized with motion start/end.

[TRAJ] section. The [TRAJ] section contains general parameters for the trajectory planning module in EMCMOT.

AXES = 3

The number of controlled axes in the system.

COORDINATES = X Y Z

The names of the axes being controlled. X, Y, Z, A, B, and C are all valid. It is also possible to have X Y Y Z and control ganged slides.

HOME = 0 0 0

Coordinates of the homed position of each axis.

LINEAR_UNITS = 0.03937007874016

The number of linear units per millimeter. For systems executing in native English (inch) units, this value is as shown above. For systems executing in native millimeter units, this value is 1. This does not affect the ability to program in English or metric units in NC code. It is used to determine how to interpret the numbers reported in the controller status by external programs.

ANGULAR_UNITS = 1.0

The number of angular units per degree. For systems executing in native degree units, this value is as shown above. For systems executing in radians, this value is 0.01745329252167, or PI/180.

CYCLE_TIME = 0.010

The period in seconds at which trajectory calculations are performed. This is a multiple of the period at which servo calculations are performed, as set in the [AXIS_#] CYCLE_TIME entry. Trajectory calculations are called at multiples of the servo period to plan linear or circular motion in Cartesian space. These values are interpolated at the servo period and run through the inverse kinematics.

DEFAULT_VELOCITY = 1.0

The initial velocity used for axis or coordinated axis motion, in user units per second.

DEFAULT_ACCELERATION = 100.0

The initial acceleration used for axis or coordinated axis motion, in user units per second per second.

MAX_VELOCITY = 5.0

The maximum velocity for any axis or coordinated move, in user units per second.

MAX_ACCELERATION = 100.0

The maximum acceleration for any axis or coordinated axis move, in user units per second per second.

PROBE_INDEX = 0

Input index of the probe used in CMM applications. See the [EMCIO] section for details of _INDEX.

PROBE_POLARITY = 1

Polarity of the probe when triggered. See the [EMCIO] section for details of INDEX.

[AXIS_#] Sections. The [AXIS_0], [AXIS_1], etc. sections contains general parameters for the individual axis control modules in EMCMOT. The axis section names begin numbering at 0, and run through the number of axes specified in the [TRAJ] AXES entry minus 1.

TYPE = LINEAR

The type of axes, either LINEAR or ANGULAR. Values for the position of LINEAR axes are in the units (per millimeter) specified in the [AXIS_#] UNITS entry. Values for the position of ANGULAR axes are in the units (per degree) specified in the same entry.

UNITS = 0.03937007874016

Units per millimeter for a LINEAR axis, as defined in the [AXIS_#] TYPE section, or units per degree for an ANGULAR axis as defined in the same section. The following parameters P, I, D, FF0, FF1, FF2 are used by the servo compensation algorithm to optimize performance while tracking trajectory set-points. See Tuning Servos for information on setting up a servomotor system.

P = 50

The proportional gain for the axis servo. This value multiplies the error between commanded and actual position in user units, resulting in a contribution to the computed voltage for the motor amplifier. The units on the P gain are volts per user unit.

I = 0

The integral gain for the axis servo. The value multiplies the cumulative error between commanded and actual position in user units, resulting in a contribution to the computed voltage for the motor amplifier. The units on the I gain are volts per user unit-seconds.

D = 0

The derivative gain for the axis servo. The value multiplies the difference between the current and previous errors, resulting in a contribution to the computed voltage for the motor amplifier. The units on the D gain are volts per user unit per second.

FF0 = 0

The 0-th order feedforward gain. This number is multiplied by the commanded position, resulting in a contribution to the computed voltage for the motor amplifier. The units on the FF0 gain are volts per user unit.

FF1 = 0

The 1st order feedforward gain. This number is multiplied by the change in commanded position per second, resulting in a contribution to the computed voltage for the motor amplifier. The units on the FF1 gain are volts per user unit per second.

FF2 = 0

The 2nd order feedforward gain. This number is multiplied by the change in commanded position per second per second, resulting in a contribution to the computed voltage for the motor amplifier. The units on the FF2 gain are volts per user unit per second per second.

CYCLE_TIME = 0.001

This is the period in seconds at which servo calculations will run. The values can be different between different axes, and the lowest will be used for all. This ensures that the calculations will occur at least as fast as they are specified here. The value should be an integer submultiple of the trajectory cycle time specified in the [TRAJ] CYCLE_TIME entry, so that an integer number of interpolations will occur. If this is not the case the times will be forced so that the interpolation interval is the next highest integer.

INPUT_SCALE = 40000 0

These two values are the scale and offset factors for the axis input from the raw feedback device, e.g., an incremental encoder. The second value (offset) is subtracted from raw input (e.g., encoder counts), and divided by the first value (scale factor), before being used as feedback. The units on the scale value are in raw units (e.g., counts) per user units (e.g., inch). The units on the offset value are in raw units (e.g., counts).

Specifically, when reading inputs, the EMC first reads the raw sensor values. The units on these values are the sensor units, typically A/D counts, or encoder ticks. These units, and the location of their 0 value, will not in general correspond to the quasi-SI units used in the EMC. Hence a scaling is done immediately upon sampling:

input = (raw - offset) / scale

The value for scale can be obtained analytically by doing a unit analysis, i.e., units are [sensor units]/[desired input SI units]. For example, on a 2000 counts per rev encoder, and 10 revs/inch gearing, and desired units of mm, we have

[scale units] = 2000 [counts/rev] * 10 [rev/inch] * 1/25.4 [inch/mm] = 787.4 counts/mm

and, as a result,

input [mm] = (encoder [counts] - offset [counts]) / 787.4 [counts/mm]

Note that the units of the offset are in sensor units, e.g., counts, and they are pre-subtracted from the sensor readings. The value for this offset is obtained by finding the value of counts for which you want your user units to read 0.0. This is normally accomplished automatically during a homing procedure.

OUTPUT_SCALE = 1 0

These two values are the scale and offset factors for the axis output to the motor amplifiers. The second value (offset) is subtracted from the computed output (in volts), and divided by the first value (scale factor), before being written to the D/A converters. The units on the scale value are in true volts per DAC output volts. The units on the offset value are in volts. These can be used to linearize a DAC.

Specifically, when writing outputs, the EMC first converts the desired output in quasi-SI units to raw actuator values, e.g., volts for an amplifier DAC. This scaling looks like:

raw = (output - offset) / scale

The value for scale can be obtained analytically by doing a unit analysis, i.e., units are [output SI units]/[actuator units]. For example, on a machine with a velocity mode amplifier such that 1 volt results in 250 mm/sec velocity, we have:

[scale units] = 250 [mm/sec] / 1 [volts] = 250 mm/sec/volt

and, as a result,

amplifier [volts] = (output [mm/sec] - offset [mm/sec]) / 250 [mm/sec/volt]

Note that the units of the offset are in user units, e.g., mm/sec, and they are pre-subtracted from the sensor readings. The value for this offset is obtained by finding the value of your output which yields 0.0 for the actuator output. If the DAC is linearized, this offset is normally 0.0.

The scale and offset can be used to linearize the DACs as well, resulting in values that reflect the combined effects of amplifier gain, DAC nonlinearity, DAC units, etc. To do this, follow this procedure:

a. Build a calibration table for the output, driving the DACs with a desired voltage and measuring the result, e.g.,

RAW MEAS
— —-
-10 -9.93
-9 V -8.83
0 -0.03
1 0.96
9 9.87
10 10.87

b. Do a least-squares linear fit to get coefficients a, b such that

meas = a * raw + b

c. Note that we want raw output such that our measured result is identical to the commanded output. This means

cmd = a * raw + b

raw = (cmd - b) / a

As a result, the a and b coefficients from the linear fit can be used as the scale and offset for the controller directly.

MIN_LIMIT = -1000

The minimum limit (soft limit) for axis motion, in user units. When this limit is exceeded, the controller aborts axis motion.

MAX_LIMIT = 1000

The maximum limit (soft limit) for axis motion, in user units. When this limit is exceeded, the controller aborts axis motion.

MIN_OUTPUT = -10

The minimum value for the output of the PID compensation that is written to the motor amplifier, in volts. The computed output value is clamped to this limit. The limit is applied before scaling to raw output units.

MAX_OUTPUT = 10

The maximum value for the output of the PID compensation that is written to the motor amplifier, in volts. The computed output value is clamped to this limit. The limit is applied before scaling to raw output units.

FERROR = 1.0 MIN_FERROR = 0.010

FERROR is the maximum allowable following error, in user units. If the difference between commanded and sensed position exceeds this amount, the controller disables servo calculations, sets all the outputs to 0.0, and disables the amplifiers. If MIN_FERROR is present in the .ini file, velocity-proportional following errors are used. Here, the maximum allowable following error is proportional to the speed, with FERROR applying to the rapid rate set by [TRAJ] MAX_VELOCITY, and proportionally smaller following errors for slower speeds. The maximum allowable following error will always be greater than MIN_FERROR. This prevents small following errors for stationary axes from inadvertently aborting motion. Small following errors will always be present due to vibration, etc. The following polarity values determine how inputs are interpreted and how outputs are applied. They can usually be set via trial-and-error since there are only two possibilities. The EMCMOT utility program USRMOT can be used to set these interactively and verify their results so that the proper values can be put in the INI file with a minimum of trouble.

ENABLE_POLARITY = 0

The polarity for enabling the amplifiers. Set this to 0 or 1 for the proper polarity. This value can be determined by following all the electronics

back from the amplifier, through any driver circuitry, etc. or it can be set through a simple trial-and-error. Normally, for amplifiers which are enabled active-low (0 volts enables), this is a 0.

MIN_LIMIT_SWITCH_POLARITY = 1

The polarity for detecting minimum-travel hardware limit switch trips. Set this depending on how your switches are wired up to the digital inputs on the I/O board.

MAX_LIMIT_SWITCH_POLARITY = 1

The polarity for detecting maximum-travel hardware limit switch trips. Set this depending on how your switches are wired up to the digital inputs on the I/O board.

HOME_SWITCH_POLARITY = 1

The polarity for detecting homing switch trips. Set this depending on how your switches are wired up to the digital inputs on the I/O board.

HOMING_POLARITY = 1

The direction in which homing moves are initiated. 0 means in the negative direction, 1 means in the positive direction.

FAULT_POLARITY = 1

The polarity for detecting amplifier faults. Set this to 0 or 1 depending upon how the amplifier sets the logic level for its fault condition.

The following entries are used to set the parameters for the DC servomotor simulations. These are only used when running the EMC in simulation.

TORQUE_UNITS = OZ_IN

The units used to interpret subsequent values for ROTOR_INERTIA and DAMPING_FRICTION_COEFFICIENT. This can be OZ_IN for ounce-inches, LB_FT for pound-feet, or N_M for newton-meters.

ARMATURE_RESISTANCE = 1.10

The resistance, in ohms, of the motor.

ARMATURE_INDUCTANCE = 0.0120

The inductance, in henries, of the motor.

BACK_EMF_CONSTANT = 0.0254

The back EMF constant, or torque constant, in volts per radian per second.

ROTOR_INERTIA = 0.0104

The rotor inertia, in torque units * seconds2.

DAMPING_FRICTION_COEFFICIENT = 0.083

The damping coefficient, in torque units per radian per second.

SHAFT_OFFSET = 0

The angular offset, in radians, between the the motor initial position and the encoder initial position. Normally this is 0, but can be made any arbitrary value if the simulated motor shaft position is interpreted as the actual axis position and should be something other than 0 when the encoder reports 0.

REVS_PER_UNIT = 10

The amount of motor shaft revolutions per user unit of position. For example, for a 1/10 inch lead screw, where 10 rotations equals 1 inch, this would be 10. The following entry is used to set the parameters for the

amplifier simulations. These are only used when running the EMC in simulation.

AMPLIFIER_GAIN = 1

The gain of the amplifier, which multiplies the input voltage to generate an output voltage which drives the motor.

MAX_OUTPUT_CURRENT = 10

The maximum output current of the amplifier, in amps.

LOAD_RESISTANCE = 1.10

The resistance, in ohms, of the load on the amplifier. This is normally the same as the motor armature resistance, but it may not be, for example, if there is additional resistive load between the amplifier and the motor itself. The following entry is used to set the parameters for the encoder simulations. These are only used when running the EMC in simulation.

COUNTS_PER_REV = 4096

The number of encoder counts per motor shaft revolution. If there is gearing between the encoder shaft and the motor shaft, this value should include this.

[EMCIO] section. The [EMCIO] section contains control values and setup parameters for the digital and analog I/O points in EMCIO.

The following entries set general parameters for the I/O controller.

PLAT = nonrealtime

Only one choice, nonrealtime.

EMCIO = minimillio

Name of the IO controller program. Valid options are :

bridgeportio Used with bridgeporttask in [TASK] section.

minimillio Used with minimilltask in [TASK] section.

ppmcio Used with bridgeporttask in [TASK] section.

simio Can be used with either minimilltask or bridgeporttask.

CYCLE_TIME = 0.100

The period, in seconds, at which EMCIO will run. You can make this as small as you want to increase the throughput. Making it 0.0 or a negative number will tell EMCIO not to sleep at all. Ultimately the system loading will limit the effective throughput.

TOOL_TABLE = tool.tbl

The file which contains tool information. The format of the file is

```
POC    FMS    LEN    DIAM    COMMENT
1      1      0      0
2      2      0      0
```

where the first line is a comment (in this case the name of the columns), and the subsequent lines contain the pocket number in which the tool is located, the tool ID of the tool itself, the length, the diameter, and an optional comment. The length and diameter are in user units.

The following entries set parameters for spindle control.

IO_BASE_ADDRESS = 0x278

Base address for the port used for IO.

PARPORT_BASE_ADDRESS = 0x278

Obsolete since July 2002 in the CVS sources, and BDI-2.2.18. Use IO_BASE_ADDRESS instead.

SPINDLE_OFF_WAIT = 1.0

How long, in seconds, to wait after the spindle has been turned off before applying the brake.

SPINDLE_ON_WAIT = 1.5

How long, in seconds, to wait after the spindle brake has been released before turning the spindle on.

The following entries set the bit indices for the digital I/O so that the controller knows the mapping to I/O point wiring. The indices start at 0 for the least significant bit in the digital I/O map. See Setting up the External Interfaces for information on interfacing I/O boards to the software.

ESTOP_SENSE_INDEX = 1

The location of the input bit which is used to detect whether the system is in ESTOP.

LUBE_SENSE_INDEX = 2

The location of the input bit which is used to detect whether the lubrication level is OK or low.

SPINDLE_FORWARD_INDEX = 1

The location of the output bit which is used to drive the spindle forward. Only applicable to manual spindles.

SPINDLE_REVERSE_INDEX = 0

The location of the output bit which is used to drive the spindle in reverse. Only applicable to manual spindles.

MIST_COOLANT_INDEX = 6

The location of the output bit which is used to turn mist coolant on or off.

FLOOD_COOLANT_INDEX = 7

The location of the output bit which is used to turn flood coolant on or off.

SPINDLE_DECREASE_INDEX = 8

The location of the output bit which is used to decrease the spindle speed. Only applicable to manual spindles.

SPINDLE_INCREASE_INDEX = 9

The location of the output bit which is used to increase the spindle speed. Only applicable to manual spindles.

ESTOP_WRITE_INDEX = 10

The location of the output bit which is used to cause an ESTOP.

SPINDLE_BRAKE_INDEX = 11

The location of the output bit which is used to engage or release the spindle brake.

The following entries set the polarities for the digital I/O points. These can be set by trial-and-error, or by noting the levels and any inverting done by the electronics between the sensors and actuators and the electronics.

ESTOP_SENSE_POLARITY = 1

The polarity of the sensed estop input bit.

LUBE_SENSE_POLARITY = 1

The polarity of the sensed lube level bit.

SPINDLE_FORWARD_POLARITY = 0

The polarity of the sensed spindle forward bit.

SPINDLE_REVERSE_POLARITY = 0

The polarity of the sensed spindle reverse bit.

MIST_COOLANT_POLARITY = 0

The polarity of the sensed mist coolant bit.

FLOOD_COOLANT_POLARITY = 0

The polarity of the sensed flood coolant bit.

SPINDLE_DECREASE_POLARITY = 1

The polarity of the sensed spindle decrease bit.

SPINDLE_INCREASE_POLARITY = 1

The polarity of the sensed spindle increase bit.

ESTOP_WRITE_POLARITY = 1

The polarity of the sensed estop activation bit.

SPINDLE_BRAKE_POLARITY = 0

The polarity of the sensed spindle brake bit.

[EMCSERVER] section

EMCSERVER = emcsvr

Name of NML server, e.g., emcsvr; if not found then none will run.

[EMCSTRIP] section. Section for emc stripchart parameters.

EMCSTRIP = emcstripchart

Name of strip chart display program e.g. emcstripchart, if not found then none will run.

OPTIONS = -f emcstrip.conf.ferror

Options for emcstripchart ussually -f something.conf. This file says which variables to plot, colors etc. -u changes the update rate.

7.2 The VAR file reference

Current releases of the RS274NGC interpreter use only a few of the 5000+ variables available in this file. These are used for offset values in the various coordinate systems, the g92 offset system, and a couple of other functions. Early versions of the VAR file could include comments but comments will be stripped from any var files used with the latest interpreters.

7.2.1 Format

Format is:

<variable> <value>

<variable> runs from 1 to 5399 inclusive, <value> is a number.

7.2.2 Units

Units for <value> are INCHES, the default units for RS-274-NGC.

7.2.3 Lines

Lines not of that form, or blank lines, are considered comments. Don't enter a line beginning with two numbers unless you mean it.
You can put whatever you want in here as user variables, but some are reserved. See the RS-274-NGC manual for the reserved variables.
This file is loaded into the interpreter at the beginning, and dumped from the controller at the end.
Only variables you put explicitly in here will be dumped out from the interpreter.

7.2.4 Reserved variable names

7.2.4.1 G28 Home Variables

Reserves parameters **5161-5166** for home position in machine coordinates for this command.

7.2.4.2 G30 Home Variables

Reserves parameters **5181-5186** for home position in machine coordinates for this command.

7.2.4.3 5220

This variable tells the interpreter which of the coordinate systems is the default. It's value must be a whole number between 1 and 9 (will be made into a decimal with zeros when EMC writes values) any other value will cause a fault when the RUN script tries to install the task program.

7.2.4.4 Coordinate System Reserved Variables

Each coordinate system reserves a set of variables that are applied to the available axes. For the coordinate systems these begin with 5221 for g54 and jump to 5241 for g55

G92 5211 to 5219

G54 5221 to 5240

G55 5241 to 5260

G56 5261 to 5280

G57 5281 to 5300

G58 5301 to 5320

G59 5321 to 5340

G59.1 5341 to 5360

G59.2 5361 to 5380

G59.3 5381 to 5400

7.3 The NML file reference

7.3.1 Some Ini File Problems That I've Encountered

My system

Tigerdirect cheapo tower and board - S3 chipset.
Cyrix pr300 - 233 actual
64M ram memory
Mandrake 7.0
Rtlinux-2.0 prepatched (how is a subject that I am writing for handbook)

emc-05-May-2000.tgz download and install (a problem with emcsh compile)Email me for my work arounds on these.

Throughout this page I will use dark green letters to indicate the lines of text that I entered into the console (konsole). I will use dark red to indicate the response of the computer to my command. Medium red lines are the focus on the discussion.ă For the sake of brevity, I have edited many of the responses of the computer when it was asked to compile large numbers of files. Some of these lengthy responses are included as files andă are linked to this document so that you can compare your results.

So far, I have loaded Mandrake 7.0 on my clean hard drive. I have built a real time kernel, and I have successfully installed EMC. Now I am ready to try running EMC. So I enter the following two lines.

```
[root@localhost /root]# cd /usr/local/nist/emc
[root@localhost emc]# ./emc.run
Current platform is linux_2_2_13
Current real-time platform is rtlinux_2_0
inivar = plat/linux_2_2_13/bin/inivar
INIFILE = emc.ini
starting emc...
starting steppermod.o...plat/rtlinux_2_0/lib/steppermod.o:  invalid
parameter PERIOD can't install it
```

That's just a little disappointing but I learned several things. The startup file found my realtime and nonrealtime directories and made links to them.ă It found and is using emc.ini. It tried to start emc using steppermod.o. But it had a problem with a parameter (variable) named PERIOD so it quit.

Looking down in the emc.ini file, I find:

```
; Base task period, in seconds
PERIOD = 0.000016
```

From experience I know that steppermod will not run with period defined in the ini file. I have two choices, change the steppermod.o to freqmod.o or remove the period variable. For this try I'm going to keep steppermod.o because It won't make my computer work quite as hard. So I comment out period in emc.ini and save the file. Comment is a ; at the start of a line. (I have since found a bug in the compile of steppermod.o from the June 2000 release that does some goofy stuff to the ini file when you shut down EMC)

```
; Base task period, in seconds
; PERIOD = 0.000016
```

I return to the console and press the up arrow and get:

```
[root@localhost emc]# ./emc.run
```

I press enter and this is the reply.

```
Current platform is linux_2_2_13
Current real-time platform is
rtlinux_2_0
inivar = plat/linux_2_2_13/bin/ini

INIFILE = emc.ini
starting emc...
starting steppermod.o...done
starting bridgeportio...done
starting bridgeporttask...done
running xemc...
```

Now I see xemc with huge red letters. When I try to bring it out of estop it won't. I press file and slide down to quit and release the mouse button. A dialog pops up quit? Yes! And xemc and EMC go away.

Won't come out of estop. 'eh. Well what are the possibilities here. (look in emc.ini) The major sections are labeled:

1. EMC
2. DISPLAY
3. TASK
4. RS274NGC
5. EMCMOT
6. TRAJ
7. AXIS_0
8. AXIS_1
9. AXIS_2
10. EMCIO
11. EMCSERVER
12. EMCSTRIP

Now the huge red letters are a clue. They say that the axis limit switches are tripped. But I don't have any axis limits hooked up! Well what's happening here is that EMC is looking at the pins that the ini says are the limits and those pins are defined in the ini as being tripped.

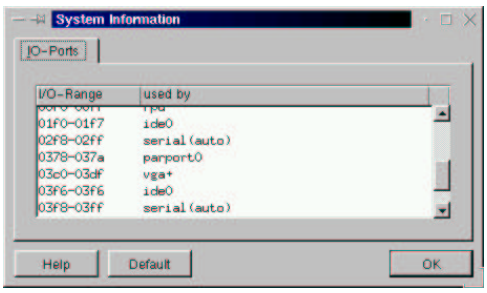
Let's start by looking in the EMCMOT section for answers. (We did that for the period problem already.)

```
[EMCMOT]
; Platform for motion
PLAT = realtime
; PLAT = nonrealtime Yes we want realtime motion so that's good.
; Name of motion control program
EMCMOT = steppermod.o
; EMCMOT = freqmod.o
; EMCMOT = stgmod.o
; EMCMOT = stg8mod.o
; EMCMOT = emcmotsim
```

We already know that it's trying to run steppermod.o. Steppermod and freqmod both use the parallel port(s) for their signals so let's look at what we have defined for the parallel port here with motion.

```
; Address for parallel port used for steppers
IO_BASE_ADDRESS = 0x378
```

Now I want to know if that is the address that my machine's parallel port is registered to so I look in the Kmenu under settings / information / I/O port information / in the popup menus for Mandrake 7. Yours may be different! [part2/images/sysinfo.gif]



Setting up Stepper Motors (from the nist page)

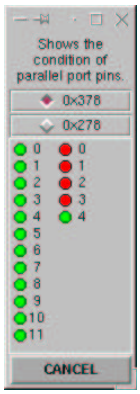
Currently the EMC supports stepper motors with a 2-bit step-and-direction interface, with bits mapped to the parallel port. Each parallel port has 12 bits of output and 5 bits of input. The outputs are used to drive the step and direction of each motor. 12 bits of output mean that up to 6 stepper motors can be controlled. The inputs can be used to detect limit or home switch trips. 5 bits of input mean that only one axes can get full positive, negative,and home switch inputs. The EMC mapping compromises for 3 axes of stepper motor control, with all positive limit switches being mapped to one input, all negative limit switches being mapped to another input, and all home switches being mapped to a third input. Other permutations are possible, of course, and can be changed in the software. You could also add 2 additional parallel ports (LPT2, LPT3), and get 36 bits of output and 15 bits of input. Some parallel ports also let you take 4 outputs and use them as inputs, for 8 outputs and 9 inputs for each parallel port. This would let you get 3 axes of control and full switch input per parallel port.

The pin-out for the EMC stepper motor interface is as follows:

```
Output          Parallel Port
------          -------------
X direction     D0, pin 2
X clock         D1, pin 3
Y direction     D2, pin 4
Y clock         D3, pin 5
Z direction     D4, pin 6
Z clock         D5, pin 7


Input           Parallel Port
-----           -------------
X/Y/Z lim +     S3, pin 15
X/Y/Z lim -     S4, pin 13
X/Y/Z home      S5, pin 12
```

Stepper motor control is implemented using a second real-time task that runs at 100 microseconds. This task writes the parallel port output with bits set or cleared based on whether the pulse should be raised or lowered. This gives an effective period of 200 microseconds for a full up-and-down pulse, or a frequency limited to about 5 kilohertz.
Well now I know that EMC is looking for all of the + limits on pin 15 - signal S3, the - limits on pin 13 - S4, and the home signals on pin 12 - S5. So what's happening at that parport address. Well there are several ways to find out. The one that I like personally is a little Tcl/Tk script that is included in EMC. It's called IO_Show.tcl. The direction signals are easy to see but the stepper clock pulses only show once in a while.



You can use this from the scripts menu in tkemc,ăwhile emc is running and see some of what's happening. You may also run it from a terminal and see what the pins are doing. In a new console window enter:

```
[root@localhost ray]# cd /usr/local/emc
[root@localhost emc]# scripts/IO_Show.tcl
```

The left column of pins are the D signals. The right column are the S signals. The right hand column begins with S3 so the first led shows that signal S3 is high.

Now I know that the EMC thinks that switches are closed to the limits and the home pins because the signals are high. Now I'm ready to look in the ini file again. This time I'm going to look in each of the axis sections because it is there that the state of the individual signals are set.

```
; First axis
[AXIS_0]
TYPE =                          LINEAR
UNITS =                         0.03937007874016
HOME =                          0.000
MAX_VELOCITY =                  1.2
P = 1000.000
I = 0.000
D = 0.000
FF0 = 0.000
FF1 = 0.000
FF2 = 0.000
BACKLASH = 0.000
BIAS = 0.000
MAX_ERROR = 0.000
DEADBAND = 0.000
CYCLE_TIME =                    0.001000
INPUT_SCALE =                   1000    0
OUTPUT_SCALE = 1.000 0.000
MIN_LIMIT =                     -10.0
MAX_LIMIT =                      10.0
MIN_OUTPUT =                    -10
MAX_OUTPUT =                     10
FERROR = 1.000
MIN_FERROR = 0.010
HOMING_VEL =                     0.1
HOME_OFFSET =                    0.0
```

```
ENABLE_POLARITY =               0
MIN_LIMIT_SWITCH_POLARITY =     1
MAX_LIMIT_SWITCH_POLARITY =     1
HOME_SWITCH_POLARITY =          1
HOMING_POLARITY =               1
JOGGING_POLARITY =              1
FAULT_POLARITY =                1
; Parameters for Inland Motor BMHS-0701 X 20
TORQUE_UNITS =                  OZ_IN
ARMATURE_RESISTANCE =           1.10
ARMATURE_INDUCTANCE =           0.0120
BACK_EMF_CONSTANT =             0.0254
ROTOR_INERTIA =                 0.0104
DAMPING_FRICTION_COEFFICIENT =  0.083
SHAFT_OFFSET =                  0
REVS_PER_UNIT =                 10
; Parameters for generic amplifier
AMPLIFIER_GAIN =                1
MAX_OUTPUT_CURRENT =            10
LOAD_RESISTANCE =               1
; parameters for generic encoder
COUNTS_PER_REV =                4096
```

I highlighted the three lines that really count for what I'm trying to do here. These are S4, S3, and S5 in roughly that order. Now we want the computer to think that the limit switches are open and the home switch is closed. That way it will home as soon as we press the home button. And it will never hit a hard limit. I should caution you that you don't want to run a real machine this way, at least not for long.

So I changed the settings of these to the lines shown below.

```
MIN_LIMIT_SWITCH_POLARITY =    0
MAX_LIMIT_SWITCH_POLARITY =    0
HOME_SWITCH_POLARITY =         1
```

Then I did it for the other two axis as well before saving the file and starting EMC again.

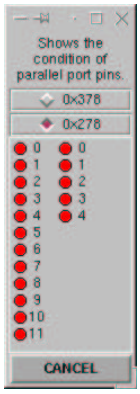
This time xemc came up with giant yellow letters. Trust me, this is an improvement. It means that each EMC axis has not been homed. But at least they are not sitting on limit switches like they were before. But EMC will still not come out of estop so we need to look further into the definitions within the ini file.

What else is there that would affect estop. Well there is a pin somewhere that looks at external estop because many machines use an external switch to turn off the power to motors when estop is required. Rather than the computer turning off power to drives, the computer is notified that power has been shut off to the drives. That way the computer can switch itself into estop state when that state exists for the machine tool. You may have noticed that this condition is not defined for the parallel port that NIST listed above. There is no estop in. So when we are using that parallel port definition, this should not be a problem. So what parallel port definition are we using with the stock emc.ini.

```
; section for main IO controller parameters ----------------------
------------
[EMCIO]
; Platform for IO controller
PLAT =                  nonrealtime
; Name of IO controller program, e.g., bridgeportio
```

```
EMCIO =                 bridgeportio
; EMCIO =               minimillio
; EMCIO =               simio
; cycle time, in seconds
CYCLE_TIME =    0.100
; tool table file
TOOL_TABLE =    emc.tbl
; address for parallel port used for auxiliary IO
IO_BASE_ADDRESS =    0x278
```

Aha! Auxiliary I/O uses a parallel port at 0x278. Now you may not know this, but the bridgeportio system uses the auxiliary I/O port for some of its functions. Stuff like mist or flood coolant, spindle speed and direction, and estop state. So let's use ioshow again to see what the state of these addresses are for my computer. (I don't have a parallel port installed at this address)



Everything at that address is set high. Now rather than figuring out the rest of what the I/O section of the ini file is trying to tell me, I'll just switch definitions of the I/O system. I can do this by changing one line. [part2/images/ioshow2.gif]

```
; Name of IO controller program, e.g., bridgeportio
EMCIO =                 bridgeportio
; EMCIO =               minimillio
; EMCIO =               simio
```

I comment out the bridgeportio line and uncomment the minimillio line. The result looks like this.

```
; Name of IO controller program, e.g., bridgeportio
; EMCIO =               bridgeportio
EMCIO =                 minimillio
; EMCIO =               simio
```

After I save my latest change, I'm ready to try running EMC again. And now I can get have green letters after I press <estop off> and <machine on>. Then with EMC still in manual mode, I clicked on the home button below the big numbers and x turned green, click on the y numbers or press y on the keyboard and clicked home again and y turned green. Repeated it for z and EMC is homed out and ready to work for me.

Your success may vary. It depends upon the version of emc.ini that is shipped with your release. It also depends upon the parallel port definitions that your machine is using and the way its parallel port hardware sets up the address values. You can click here to view my edited emc.ini.ă You are also welcome to save it and use it for your system.

Now at least you know how I did it.

Good luck

7.4 Tuning freqmod

When If you have stepper motors and a stepper motor controller that requires phase stepping, then freqmod.o is a solution. Phase stepping requires the computer to output the motor phase signals directly, the stepper driver is just an amplifier.

This procedure version 1.0 John Sheahan 11 September 2001, Developed with linux kernel2.2.18 / rtlinux-rtl3.0-6 / :emc-2.07. Setup of linux/emc is covered elsewhere and must be done first

Setup The assumption is that your stepper motors are running on the bench. Put a marker on the shaft so that rotation is clear. I used wood scraps. Connect the stepper controller to the chosen parallel port, wire up the steppers. Don't power up the steppers yet.

As I'm a shell dinosaur I'll describe things that way. Adjust suitably for GUI operation.

to test the base setup:

```
# get to the relevant directory
/usr/local/emc
# become root
su -
# run the simulation.  Note, the motors will not turn and are not needed,
yet.
./sim.run
# hit F1 to get from ESTOP state to ESTOP RESET
# hit F2 to get to ON state
# try the arrow keys and pageup pagedown to jog the motors
# hit F4 to get to AUTO state
# click file -> open
# click cds.ngc
# click open
# view ->  backplot
# click run
```

wait and watch the windows scroll.

This was intended to test the base install and introduce a little functionality. Next step is to make the software run with YOUR motors. Choose a name for your setup. I'll use table as I'm building a routing table. Substitute _yourname_ for table in the rest.

```
# copy the sim files as we now know they work :)
# copy the runfile.  If you do this with a gui, ensure the file is ex-
ecutable after
cp -p sim.run table.run
# copy the variables
cp sim.var table.var
# copy the tool file
cp sim.tbl table.tbl
# copy the nml file
cp sim.nml table.nml
```

Now we need to change the references in the new files. With your favorite editor, in table.run, change the line

```
INIFILE=sim.ini
```

to read

```
INIFILE=table.ini
```

The nml and tbl files should be ok for now. The real work comes in the ini file. tuning the .ini file with your favorite editor initial requirements You need to have this information to hand:

what units you want to operate the machine in, inches or mm

the leadscrew pitch [each axis]

the number of steps per revolution [each axis]

the maximum velocity each axis can run at.

this is getting harder, might be limited by the computer, the steppers, stepper driver, or fear.

the maximum acceleration for each axis

you might calculate this or just experiment Now edit table.ini Adjust the sections as described here. I'm mostly only going to discuss what needs to be changed.

```
[EMC]
MACHINE = Stepper Stand
```

set the name string to whatever you want.

```
NML_FILE = table.nml
DEBUG = 0x7FFFFFFF
```

bring it down to 0x3 or 0x0 as soon as things are making sense

```
RS274NGC_STARTUP_CODE=G20
```

set this to G20 to accept inches, or G21 for the rest of the world. This is the default unit for GCODE commands

```
[DISPLAY] is ok as is.
[TASK] is ok as is.
[RS274NGC]
PARAMETER_FILE = table.var
[EMCMOT]
PLAT = realtime
EMCMOT = freqmod.o
STEPPING_TYPE = 1
; 0= drn,clock ; 1 == phase step ; 2 == table driven, see emcmot.c
; only works with freqmod.o
PERIOD = 0.000050
```

this one is tricky, its the interrupt time for the timer ISR in seconds. If you make it too fast (small number), your computer will lock solid. 50us works for a 150MHz computer. If your cpu is faster, shrink this time accordingly. It should be as small as practical, and should divide cleanly into CYCLE_TIME. The smaller this is the faster and smoother your steppers can be controlled.

```
[TRAJ]
LINEAR_UNITS = 1.0
```

1.0 is good if you are going to use metric for the rest of the numbers in this section. If you prefer inches, set to 0.03937007874016

```
CYCLE_TIME = 0.010
```

controls the rate trajectory will be computed.

```
DEFAULT_VELOCITY = 0.03
MAX_VELOCITY = 2.60
```

this needs to be bigger than any of the axis limits. Don't make it too small, as the Z axis will behave very strangely!

```
DEFAULT_ACCELERATION = 1.5
MAX_ACCELERATION = 2.0
```

choose this to keep your motors tracking.

[AXIS_0] note, all the axis have the same format, described once.

```
UNITS = 1.0
```

as above [TRAJ]linear_units

```
MAX_VELOCITY = 1.2
```

an appropriate limit for this axis and your motors.

```
P = 10.000
```

gain constant. this may need tuning later, see various comments on PID tuning in other documents. 10 worked for me.

```
I = 0.0
D = 0.0
FF0 = 0.0
```

I got too much endpoint error unless this was 0.

```
FF1 = 0.0
FF2 = 0.0
BACKLASH = 0.000
```

tweak this later to match your measured mechanical performance

```
DEADBAND = 0.010
```

set this to just over half what your single step resolution is, ie a little more than 1/INPUT_SCALE

```
CYCLE_TIME = 0.0010000
INPUT_SCALE = 133.333333333 0
```

I have 200 step/rev stepper motors and a 1.5mm/turn leadscrew (cheap!) so I get 200/1.5 = 133.3 steps/mm. Offset is zero.

```
OUTPUT_SCALE = 133.333333333 0.000000000
```

set this the same as INPUT_SCALE

```
MIN_LIMIT = -100.0
```

this is the travel limit from home of your axis

```
MAX_LIMIT = 100.0
```

the travel limit in the positive direction

```
MIN_OUTPUT = -100
MAX_OUTPUT = 100
FERROR = 4.000
MIN_FERROR = 1.000
HOMING_VEL = 0.1
HOME_OFFSET = 0.0
ENABLE_POLARITY = 0
MIN_LIMIT_SWITCH_POLARITY = 0
```

these polarity settings set the way your limit switches are wired. If the system will not come out of RESET state, change to 1

```
MAX_LIMIT_SWITCH_POLARITY = 0
HOME_SWITCH_POLARITY = 0
HOMING_POLARITY = 1
JOGGING_POLARITY = 1
FAULT_POLARITY = 1
[EMCIO]
EMCIO = bridgeportio
```

this is probably silly, and will require a second parallel port in your computer. consider a different setting here. Advice please.

```
TOOL_TABLE = table.tbl
ESTOP_SENSE_POLARITY = 0
```

to let it start.

```
LUBE_SENSE_POLARITY = 0
[EMCSERVER]
; EMCSERVER = emcsvr
```

comment it out, not using it currently

now test it now all that typing is done, lets make the motors spin! Here is a suggested test procedure. Power up the stepper drivers (preferably with the steppers not mechanically connected to anything, as that may make expensive noises

execute *./table.run.*

hit F1. (with the cursor in the nice blue window) state should change from ESTOP to ESTOP_RESET. If not, you probably have ESTOP_SENSE_POLARITY backwards, swap 1 and 0.

alternatively you only have one parallel port and would be better off with EMCIO = minimillio

hit F2. State should now become ON. If not, one of your SWITCH_POLARITY settings is probably reversed. You can use scripts/IO_Show.tcl to check this, if the upper right hand column is red, you need a 0 in the POLARITY setting.

try jogging the axis by hitting the cursor arrow keys, and pageup / pagedown. The motors should move a tad. now try a bigger move at full speed.Lets use some gcode.

hit F5, state should change to MDI for manual entry.

type g0x0 <return>

this should move the X axis 1 mm (or 1 inch if that's the RS274NGC_STARTUP_CODE default chosen).

type g0x10y10z5 <return>

all three motors should move off to x=10 y=10 z=5 checks, please everyone add more.

if the motors rattle backwards and forwards a step, your deadzone is too small. click on the axis to fix, click settings->calibration, edit deadband, click ok.

if the final displayed value is too different to the final requested value, your deadband is too big. Or you may have too high FF coefficients.

if your motors emit funny noises when stopped or moving, your gain (p) is too high.

if your motor gets there very slowly, your P gain is too low.

if you get 'following' errors, your P gain is too low, or your output_scale is wrong. calibration check

type g0x0y0z0 <return>

this gets things back to zero. Arrange the motor position and shaft marker so you can see them.

now lets ask for a 1-motor-revolution move. I have 1.5mm pitch leadscrews, so:

type g01x1.5y1.5z1.5f10 <return>

I just asked to move to 1.5mm on all three axis, at 10mm/minute. Adjust for your leadscrew pitch.

the motors should turn exactly 1 revolution. If not, the following is possible:

you miscalculated or mistyped the distance.

the motor body moved

the shaft marker moved (it got me!)

the input_scale setting is wrong on that axis now switch to auto mode (F4 or the mouse) and file->open the skeleton file. click run and watch it all whir. then its probably time to connect up more mechanics and test some more.

John Sheahan

Chapter 8

PID Axis Tuning

Axis tuning is a critical part of most emc setups. Until recently we have been able to ignore tuning by using steppers and steppermod.o. You can still do this but the newer motion files, freqmod and smdromod allow stepper users much smoother and better control of axis motion. Smdromod.o even allows stepper users to feed back actual position using a home built feedback board. (no more lost steps) But the consequence of these newer motion systems is that users must now learn some of the basics of axis tuning.

The description of PID tuning that follows is not intended to be exhaustive or rigorous but should get the beginning emc user started with these newer motion files and at least able to keep their axis from tripping out on overtravel. This page assumes that the reader is acquainted with the emc.ini file where tuning values are stored for each axis.

There are a number of excellent internet resources that will extend this description. A few links are listed at the end of this page. There are several auto tuning and computational tuning procedures but these have not been tested with EMC.

P - proportional, I - integral, and D - derivative are three common mathematical techniques that are applied to the task of getting a working process to follow a setpoint. In the case of EMC the process we want to control is actual axis position and the setpoint is the commanded axis position. PID is NIST's chosen way to connect these two things.

A mechanical system, something like a pantograph, will serve to illustrate the control problem. With a pantograph you can trace the stylus around a pattern and the pen or pencil will produce the result. The stylus is the commanded position, the pen the actual. The pattern need not be the same size as the resulting drawing. That relationship depends upon the linkage.

With a pantograph, we can consider the question, "How good is the drawing?" The answer depends upon several factors, speed of movement, detail in the pattern, sharpness of the stylus and pencil, size of stylus and pencil, differences in size between the two, etc.

When we get to electrical or electronic systems like EMC, instead of a hard linkage between the pattern and the product, we have signals created by reading nc program code, reading jog commands, or reading position from a digital image or drawing. These signals are sent to an amplifier and its

output is sent to a motor. Most often the mechanical system driven by the motor has a feedback device that returns actual position to EMC.

EMC, or any computer controlled machine. should be thought of as a pantograph made with rubber bands in place of some of the rods and squishy bearings for some of the joints. PID tuning allows the integrator to control the stiffness of the rubber bands and the squashiness of the joints. With a mechanical pantograph the forces are all controlled by the fingers on the stylus. If the fingers are capable of 0.0001 offsets in motion then the results will be that fine. Starting, stopping, and changing the direction of the pen's inertia is also totally dependent upon the fingers of the operator. But with an electrical system, all of these things must be accounted for in the signals that produce the motion.

In any system that reproduces motion, one of the major "goodness" factors is what we call following error. Following error is a way of quantifying how close the actual position is to the commanded position while the tool follows the range of movements that can be commanded.

8.1 Ray's Experience

All three variables (6+ really) look at what is happening between commanded and actual position in the emc. My comments below may be a bit unsatisfying for those who are accustomed to very precise things but they come from a number of years of twiddling with servo controls.

P - process variable.

This is the gain control. It is a bit like the volume on a music system. If it is set too low you can't follow the lyrics. If it's set too high the windows rattle and the neighbors fuss.

Some authors refer to P as proportional band. Think of it as a pair of lines, one ahead of the commanded position and one behind it. Actual position should be somewhere between the lines. If the actual position is farther behind then that below line the controller will run the axis flat out to reach the slower line. Conversely, if the actual position is ahead of the above line the controller will do all it can to bring actual position down to that upper bound.

As long as the actual position is between the lines, the controller will ramp gain up and down so that actual approaches commanded.

The larger the P number the narrower the space between these lines. Set P too low and your axis works like a dedicated couch potato during super bowl or world soccer finals – even a fire may not rouse it. This condition is rather easy to spot because the axis is sluggish.

Set P too high and your axis will develop palsy. On most machines you can hear this condition by putting your ear to the motor. You may want to do this when others aren't watching – or use a stethoscope or a long socket extension – or perhaps the graph function.

Palsy will sound like a hum or grind when the motor is sitting still or moving very slowly. You can also feel palsy if you wrap your hand around the ball screw or grip the drive belt or pulley. (disclaimer – This demonstration is done by a professional with only three remaining fingers, don't try this at home)

Deadband may mask too much P when a motor is sitting still so you may want to move the axis very slowly and listen to the sounds around each step. If the axis keeps up with commanded position at high speed and during acceleration and there is not a lot of ringing, grinding, jumping at very low speed, then you are real close.

I - Integral variable.

———————-

Integral works a bit like a shock absorber. Any change in either actual or commanded position gets rounded off or averaged in so that acceleration/deceleration brought about by P is absorbed and released more slowly over time.

No integral and you get the full P effect of change in commanded position. Too much integral and the axis seems to wander off on it's own without much regard for P. A little integral may smooth out some of the frequency jumps when a stepper is running right near one of those troublesome rates.

D - Derivative variable

———————

Derivative works like passing gear for acceleration or a jake-brake for stopping. Whenever commanded position changes rapidly, d will really kick the amp/motor in the *** to follow the rate-of-change of the axis command rather than the difference between commanded and actual position.

Derivative works against inertia so if you've got lots of iron to start or stop dial some in. But derivative will increase palsy so you have to balance it against gain.

FF1-3 Feed Forward variables

———

I know very little about the effect of these variables.ă My first experience with them is while using EMC.ă I need to include an idea from Jon Elson and his work with servos and his Bridgeport.ă He has used a value up to 8.0 for FF1 with a somewhat reduced acceleration to very successfully minimize following error.ă The relevant portions of his ini file are:

```
DEFAULT_VELOCITY =      0.75
MAX_VELOCITY =          1.5
DEFAULT_ACCELERATION =  2.0
MAX_ACCELERATION =      2.0
MAX_VELOCITY =          1.2
P =                   100.000
I =                     0.000
D =                     0.000
FF0 =                   0.000
FF1 =                   7.500
FF2 =                   0.000
```

Your results will no doubt vary.

T - Test

————

The final proof of tuning is in the cutting. So after your best guess with all the watching, hearing, feeling done to each axis, get out a chunk of soft aluminum, a small end or ball mill, and begin to mill circles or arcs that pass 90 degrees between each pair of axis. I like outside circles because you can hold them up to the light and see how the finish looks near the quadrants.

Digital systems will give you some steps as one axis approaches zero and the other approaches the set feedrate so don't expect a perfect mirror arc finish. Backlash and backlash compensation also affect the appearance here. Servo drives will work better than steppers. (digital vs analog) But the smoother the saw teeth the better.

Be prepared to spend some metal on this! And make a list of your settings, changes, and a better/worse judgment about the result of each change. On occasion I've spent pages of paper and made piles of swarf to get a stubborn machine to where I wanted it.

Good luck and may the electromotive force be with you.

Ray <rehenry@up.net>

8.2 Tim's Experience

EMC now is handling steppers as if they were servos from what I understand (this is with freqmot, with or without DRO feedback) and using freqmot I can confirm that P,I, and D definitely have an effect on steppers. If I set P too low I get constant follow errors at any speed, If I set D to anything but 0 I start having a problem with the motors stalling on acceleration (probably because this setting is trying to make the motor get up to speed faster than the stepper can accelerate). I have played with various I settings between 1 and 1000 and it does seem to help me get even higher feed rates, but I have not really figured a way to see what works best on this one.

You can easily play with these settings by going to the settings menu and selecting calibration. The values will be for the axis that is currently selected and will take effect as soon as the calibration window is closed. Make sure you remember or write down the setting you settle on as the values you put in the calibration window are not saved back to your .ini file, you have to edit them in manually.

Tim
Date: Sat, 4 Mar 2000 14:28:49 -0700
From: "Tim Goldstein" <timg@ktmarketing.com>

8.3 Jon's Experience

I also found while working with the servo tuning (last summer)... FF1 (I think that is the non-zero FFn parameter in my .ini file) is VERY effective in improving following error, and is, in fact, BETTER than a PID could do, because, of course, it knows IN advance what is GOING to happen at a velocity change!ă I was able to get following error down to under .001" under practically all circumstances by setting FF1 to a small

value, somewhere between 4 and 7.5 or so. You can see my graph at http://www.artsci.wustl.edu/~jmelson/servo.html showing a jog move at 90 IPM. Oh, yes, one other thing, the acceleration parameters in the default EMC .ini file are ridiculous. I had been ignoring jerky motion when manually jogging when I started working with EMC, until I broke a brand new toothed belt on my Y axis. This led me to realize the system was demanding much too much acceleration, and I slowed it down quite a bit. No belts have broken since then.

Jon
Date: Wed, 14 Jun 2000 17:38:00 -0500
From: Jon Elson <jmelson@artsci.wustl.edu>

–2–

JohnDRoc@aol.com wrote:

> Yes, it's fast, it's definitely an industrial-strength milling center. It's
> not a humming sound it's more like a grinding or laboring sound. It runs in
> my mind that it ran more "freely" before, but it might just be a result of
> the compensation - maybe I didn't notice when it was slamming back and forth.

Well, I still think it is what I saw, but maybe not. Does the sound change at different jog speeds? What is your P parameter in the axis setup, in fact what are all your parameters in there? If the P gain is set too high, it could cause rough operation. Yes, it could be tuning of the servo amp, too. I found it was best to run with a conservative P, I and D set to zero, and a small value on FF1, about 5.0 This gave me really minute following error and quite solid position holding, without any instability.

>ă I think the next step is going to be working with the logging program, to
> determine the fine tuning of the amps.ă Then, I think I saw in a post from
> Fred, that there is supposed to be a program that helps come up with the PID
> values.

Unless they've done some serious repair in the PID department, the I and D are programmed wrong, and do not do anything useful. Unfortunately, the controls theory used in EMC gave a steady-state, or at least one quadrant, definition of PID algorithms. The problem is this is a motion control problem in all FOUR quadrants. And, Integral history from when you were in a different quadrant is not only irrelevant, but makes your solution MORE inaccurate, instead of reducing error. But, pedantically, because you are SUPPOSED to use the entire history of the system, EMC holds that you MUST use the entire history, even if it fails to perform the necessary goal.

The Derivative term probably works, in the general sense, but since this is a quantized system, and in slow motion there are so few encoder counts per servo cycle, the Derivative term from each encoder cycle has WAY too much quantization noise to perform well. It needs to be smoothed a bit. Fortunately, they put in the FF0, FF1 and FF2 terms, which are both mathematically pure, and of great use. FF1 is actually better than some combination of I and D, because it can respond BEFORE error develops, as in a rapid acceleration of the system from rest. The I term won't know what is going on for minutes, because it is taking the average of millions of tiny errors past, and due to the quantization, you can't set the D term very high, or the system gets unstable. As long as the servo amps are well

behaved, just using P and FF1 has worked VERY well for me.

So, I don't know what the auto-tune program will do with those parameters. If good, it will also find them to be unhelpful, and leave them close to zero.

Jon
Date: Wed, 12 Jul 2000 02:23:39 -0400 (EDT)
From: Jon Elson <jmelson@artsci.wustl.edu>

8.4 Fred's PID Report

I had a guest researcher here at NIST for a few months, looking into automatic system identification and PID tuning. I have a few MS Word pictures showing what he did that I WinZip'ed up and put on the FTP site in the emc/emcsoft directory, as "pidtuning.zip". They're 8.5x11 posters that show the theory and some figures for our Bridgeport machine.

It works for systems without a tachometer. We were trying to get the performance (following error) to be equal to a machine with tachometers, to reduce cost.

The idea is to hit the axis with a step voltage, and log the resulting position v. time. The curve rises to some steady state velocity. The steady state velocity is a function of applied voltage. The time to, say, 75% of steady state is the same. For several runs, you can get an average of steady-state velocity per applied voltage, and average rise time. These can be used to deduce PID gains.

The student, Kees ("Case") Stolk, from the University of Twente in the Netherlands, wrote a Tcl/Tk script that automates much of the process, including going into machine-off, opening the log, running the DAC out command, saving the log, storing multiple runs, and popping up PID gains. It's pretty slick. I'll put this up on the FTP site once I verify that it works with the new release.

I ran this on the Bridgeport and the resulting gains outperformed my manually tuned gains for current mode (no tach), and equaled the manually tuned gains (with a FF1 feedforward term) in velocity mode (with a tach).

Date: Mon, 6 Mar 2000 18:05:59 -0500 (EST)
From: Fred Proctor <proctor@cme.nist.gov>

8.5 Links

http://www.manufacturing.net/magazine/ce/archives/1998/ctl0301.98/03a305.htm

http://www.expertune.com/tutor.html
http://www.expertune.com/PIDspec.htm
http://www.manufacturing.net/magazine/ce/archives/1998/ctl0801.98/08abas.htm

http://members.tripod.com/aabi/index.htm
http://members.tripod.com/aabi/optimumpid/optimum_menu.htm
http://www.ctc-control.com/tutorials/pid.htm
http://www.newport.com/Motion_Control/Tutorial/Servo_Tuning_Principles
/Servo_Tuning_Principles/description.php
PID tuning: Lieslehto, J., Tampere University of Technology, Tampere, Finland, (used by Kees ("Case") Stolk)

This last one is a good web reference for PID tuning, with Java applets.

8.6 Print Resources

"Process Control Systems" by F. Greg Shinskey, available from the Foxboro Training Institute at 1-888-FOXBORO.

Controller Tuning and Control Loop Performance: A Primer, Second Edition, Subtitled PID Without the Math, by David W. St. Clair (Retired DuPont Engineer)

Process Instrumentation Applications Manual, by Bob Connel, McGraw-Hill Book Co., 1996

References used by Kees ("Case") Stolk while developing NIST tuning software:

Modeling of DC Motors, Kuo, Benjamin C., Automatic Control Systems, Prentice-Hall, Englewood Cliffs, NJ, 1981.

Nonlinear Least Squares Model Fitting, Press, William H. et al, Numerical Recipes in C: the Art of Scientific Computing, Cambridge University Press, Port Chester, NY, 1992.

PID controllers, Åstrom, K., and Haglund, T., PID Controllers: Theory, Design and Tuning, Instrument Society of America, Research Triangle Park, NC, 1995.

"Internal Model Control", Rivera, M., Internal Model Control for PID Controller Design, 1986.Credits

Chapter 9

INI File Reference

The EMC is configured with files that are read at startup and used to override the compiled defaults. No real controller will likely use the compiled defaults, so you will certainly need to edit at least some of these files to reflect the specifics of your machine.

There are four files: emc.ini, emc.nml, tool.tbl, and emc.var. The first, emc.ini, contains all the machine parameters such as servo gains, scale factors, cycle times, units, etc. and will certainly need to be edited. emc.nml contains communication settings for shared memory and network ports you may need to override on your system, although it is likely that you can leave these settings alone. tool.tbl contains the tool information such as which pocket contains which tool, and the length and diameter for each tool. rs274ngc.var contains variables specific to the RS-274-NGC dialect of NC code, notably for setting the persistent numeric variables for the nine work coordinate systems.

The specific formats of each of these files is detailed in the following sections.

Machine Configuration file emc.ini

The machine configuration file emc.ini follows the Microsoft INI file format, in which values are associated with keywords on single lines, perhaps in sections denoted with square brackets, e.g.,

```
[SECTION]
; COMMENT
SYMBOL = VALUE
```

Everything from the first non-whitespace character after the = up to the end of the line is passed as the value, so you can embed spaces in string symbols if you want to.

You can edit the values for each keyword in any text editor. The changes won't take effect until the next time the controller is run. The file should be called emc.ini, but the name can be overridden using a command line argument to the controllers. See the section "Starting Up" for information on how to do this. The following sections detail each section of the configuration file, using sample values for the configuration lines.

[EMC] section. The [EMC] section contains general parameters for the whole controller. These are:

VERSION = $Revision: 1.1 $

The version number for the INI file. This is automatically updated when using the Revision Control System, which looks for the $Revision: 1.1 $ string and appends the revision number. If you want to edit this manually just change the number and leave the other tags alone.

MACHINE = My Controller

This is the name of the controller, which is printed out when it runs. You can put whatever you want here.

NML_FILE = emc.nml

The name of the NML file to use.

DEBUG = 0x00000003

Sets the verbosity of the debugging messages printed out on the text console. Valid options are :

0x00000000 Do not print any debugging messages.

0x00000001 Print invalid messages

0x00000002 Print configuration settings

0x00000004 Print defaults

0x00000008 Print version

0x00000010 Print task messages

0x00000020 Print IO points

0x00000040 Print NML messages

0x00000080 Print time taken for motion to complete

0x00000100 Print interpreter debugging

0x00000200 Print RCS debugging

0x00000400 Print raw trajectory data

0x00000800 Print interpreter list

0x7FFFFFFF Print all debugging messages

The numbers can be combined (logical AND) so that selected information can be printed. i.e. 0x00000003 would print invalid messages AND configuration settings.

RS274NGC_STARTUP_CODE = G21 G90

A string of NC codes that the interpreter is initalised with.

[DISPLAY] section. The [DISPLAY] section contains parameters for the Graphical User Interface.

PLAT = nonrealtime

Only one choice, nonrealtime.

DISLAY = tkemc

The name of the user interface to use. Valid options are :

emcpanel

keystick

tkemc

tkemcts

xemc

yemc

CYCLE_TIME = 0.200

The period, in seconds, at which the display will be updated.

HELP_FILE = doc/help.txt

Location and name of a plain text help file to be used when the GUI HELP button is pressed.

POSITION_OFFSET = RELATIVE

Initial display setting for position, RELATIVE or MACHINE.

POSITION_FEEDBACK = ACTUAL

Initial display setting for position, COMMANDED or ACTUAL.

MAX_FEED_OVERRIDE = 1.2

Highest value that will be allowed for feedrate override. 1.0 = 100%

PROGRAM_PREFIX = programs/

The prefix to be added to any NC program prior to loading.

INTRO_GRAPHIC = emc.gif

File name of the image to be displayed whilst EMC is loaded. This can be commented out if not required.

INTRO_TIME = 5

Time, in seconds, that the intro image is displayed.

BALLOON_HELP = 1

Enable popup balloon help with tk based GUIs.

LINEAR_UNITS = AUTO

Units used for display of linear axis. AUTO, INCH, MM, or CM

ANGULAR_UNITS = AUTO

Units used for display of rotary axis. AUTO, DEG, RAD, or GRAD.

[TASK] section. The [TASK] section contains general parameters for EMCTASK, which includes primarily the NC language interpreter and the sequencing logic for sending commands to EMCMOT and EMCIO.

PLAT = nonrealtime

Only one choice, nonrealtime.

TASK = minimilltask

Name of the task controller program, bridgeporttask or minimilltask. This will need to correspond with IO controller program.

CYCLE_TIME = 0.010

The period, in seconds, at which EMCTASK will run. You can make this as small as you want to increase the throughput. Making it 0.0 or a negative number will tell EMCTASK not to sleep at all. Ultimately the system loading will limit the effective throughput.

[RS274NGC] section. Part program interpreter section.

PARAMETER_FILE = emc.var

Name of the file containing variables used for coordinate offsets.

[EMCMOT] section. The [EMCMOT] section contains critical parameters for the motion control module.

PLAT = realtime

Only one choice for most, realtime.

EMCMOT = steppermod.o

The motion control module to be used. Choices are :

steppermod.o The original stepper motor module using the parallel port. Not available with rtai systems.

freqmod.o An improved stepper motor module requiring PERIOD to be set.

smdromod.o freqmod with encoder feedback. Requires DRO_BASE_ADDRESS to be set.

stgmod.o Servo motor control using the Servo To Go card.

stg8mod.o Eight axis version of stgmod

stg2mod.o Servo motor control using the Servo To Go II card.

newstgmod.o Updated version of stgmod for both versions of the Servo To Go card.

vtiisamod.o Servo motor control using the ISA Vigilant Products card.

vtipcimod.o PCI version of vtiisamod. Only available for 2.4.xx based systems.

ppmcmod.o Jon Elson's parallel port motion control system for servo motors.

ppmcmod8.o Eight axis version of ppmcmod

univstepmod.o Stepper motor control via Jon Elson's parallel port system.

simmod.o Simulated motion control module.

minitetra.o Hexapod kinimatics using freqmod.

SHMEM_KEY = 100

Key used for shared memory.

SHMEM_BASE_ADDRESS = 0x3F00000

Base address of physical shared memory. For rtlinux_09J, this must point to memory reserved during boot up via "mem = 31M" in /etc/lilo.conf. This is not required for all later realtime linux systems as a different mechanism is used to allocate shared memory.

IO_BASE_ADDRESS = 0x378

Base address for the motion control card or parallel port.

PARPORT_BASE_ADDRESS = 0x378

Obsolete since July 2002 in the CVS sources, and BDI-2.2.18. Use IO_BASE_ADDRESS instead.

STG_BASE_ADDRESS = 0x200

Obsolete since July 2002 in the CVS sources, and BDI-2.2.18. Use IO_BASE_ADDRESS instead.

DRO_BASE_ADDRESS = 0x200

Base address for the Mauch/Kaluga DRO card used by smdromod for encoder feedback.

COMM_TIMEOUT= 1.0

Timeout for comms to emcmot in seconds.

COMM_WAIT = 0.010

Interval between tries of comms to emcmot in seconds.

PERIOD = 0.000020

Base period for output pulses with freqmod in seconds. The shorter the interval, the higher the maximum feedrate becomes, but at the expense of system throughput. Too smaller a period will cause the whole system to lock up.

Do NOT use this parameter with any module other than freqmod or smdromod. The system WILL lock up.

MOTION_IO_ADDRESS = 0x3BC

Port used for IO synchonized with motion start/end.

[TRAJ] section. The [TRAJ] section contains general parameters for the trajectory planning module in EMCMOT.

AXES = 3

The number of controlled axes in the system.

COORDINATES = X Y Z

The names of the axes being controlled. X, Y, Z, A, B, and C are all valid. It is also possible to have X Y Y Z and control ganged slides.

HOME = 0 0 0

Coordinates of the homed position of each axis.

LINEAR_UNITS = 0.03937007874016

The number of linear units per millimeter. For systems executing in native English (inch) units, this value is as shown above. For systems executing in native millimeter units, this value is 1. This does not affect the ability to program in English or metric units in NC code. It is used to determine how to interpret the numbers reported in the controller status by external programs.

ANGULAR_UNITS = 1.0

The number of angular units per degree. For systems executing in native degree units, this value is as shown above. For systems executing in radians, this value is 0.01745329252167, or PI/180.

CYCLE_TIME = 0.010

The period in seconds at which trajectory calculations are performed. This is a multiple of the period at which servo calculations are performed, as set in the [AXIS_#] CYCLE_TIME entry. Trajectory calculations are called at multiples of the servo period to plan linear or circular motion in Cartesian space. These values are interpolated at the servo period and run through the inverse kinematics.

DEFAULT_VELOCITY = 1.0

The initial velocity used for axis or coordinated axis motion, in user units per second.

DEFAULT_ACCELERATION = 100.0

The initial acceleration used for axis or coordinated axis motion, in user units per second per second.

MAX_VELOCITY = 5.0

The maximum velocity for any axis or coordinated move, in user units per second.

MAX_ACCELERATION = 100.0

The maximum acceleration for any axis or coordinated axis move, in user units per second per second.

PROBE_INDEX = 0

Input index of the probe used in CMM applications. See the [EMCIO] section for details of _INDEX.

PROBE_POLARITY = 1

Polarity of the probe when triggered. See the [EMCIO] section for details of INDEX.

[AXIS_#] Sections. The [AXIS_0], [AXIS_1], etc. sections contains general parameters for the individual axis control modules in EMCMOT. The axis section names begin numbering at 0, and run through the number of axes specified in the [TRAJ] AXES entry minus 1.

TYPE = LINEAR

The type of axes, either LINEAR or ANGULAR. Values for the position of LINEAR axes are in the units (per millimeter) specified in the [AXIS_#] UNITS entry. Values for the position of ANGULAR axes are in the units (per degree) specified in the same entry.

UNITS = 0.03937007874016

Units per millimeter for a LINEAR axis, as defined in the [AXIS_#] TYPE section, or units per degree for an ANGULAR axis as defined in the same section. The following parameters P, I, D, FF0, FF1, FF2 are used by the servo compensation algorithm to optimize performance while tracking trajectory set-points. See Tuning Servos for information on setting up a servomotor system.

P = 50

The proportional gain for the axis servo. This value multiplies the error between commanded and actual position in user units, resulting in a contribution to the computed voltage for the motor amplifier. The units on the P gain are volts per user unit.

I = 0

The integral gain for the axis servo. The value multiplies the cumulative error between commanded and actual position in user units, resulting in a contribution to the computed voltage for the motor amplifier. The units on the I gain are volts per user unit-seconds.

D = 0

The derivative gain for the axis servo. The value multiplies the difference between the current and previous errors, resulting in a contribution to the computed voltage for the motor amplifier. The units on the D gain are volts per user unit per second.

FF0 = 0

The 0-th order feedforward gain. This number is multiplied by the commanded position, resulting in a contribution to the computed voltage for the motor amplifier. The units on the FF0 gain are volts per user unit.

FF1 = 0

The 1st order feedforward gain. This number is multiplied by the change in commanded position per second, resulting in a contribution to the computed voltage for the motor amplifier. The units on the FF1 gain are volts per user unit per second.

FF2 = 0

The 2nd order feedforward gain. This number is multiplied by the change in commanded position per second per second, resulting in a contribution to the computed voltage for the motor amplifier. The units on the FF2 gain are volts per user unit per second per second.

CYCLE_TIME = 0.001

This is the period in seconds at which servo calculations will run. The values can be different between different axes, and the lowest will be used

for all. This ensures that the calculations will occur at least as fast as they are specified here. The value should be an integer submultiple of the trajectory cycle time specified in the [TRAJ] CYCLE_TIME entry, so that an integer number of interpolations will occur. If this is not the case the times will be forced so that the interpolation interval is the next highest integer.

INPUT_SCALE = 40000 0

These two values are the scale and offset factors for the axis input from the raw feedback device, e.g., an incremental encoder. The second value (offset) is subtracted from raw input (e.g., encoder counts), and divided by the first value (scale factor), before being used as feedback. The units on the scale value are in raw units (e.g., counts) per user units (e.g., inch). The units on the offset value are in raw units (e.g., counts).

Specifically, when reading inputs, the EMC first reads the raw sensor values. The units on these values are the sensor units, typically A/D counts, or encoder ticks. These units, and the location of their 0 value, will not in general correspond to the quasi-SI units used in the EMC. Hence a scaling is done immediately upon sampling:

input = (raw - offset) / scale

The value for scale can be obtained analytically by doing a unit analysis, i.e., units are [sensor units]/[desired input SI units]. For example, on a 2000 counts per rev encoder, and 10 revs/inch gearing, and desired units of mm, we have

[scale units] = 2000 [counts/rev] * 10 [rev/inch] * 1/25.4 [inch/mm] = 787.4 counts/mm

and, as a result,

input [mm] = (encoder [counts] - offset [counts]) / 787.4 [counts/mm]

Note that the units of the offset are in sensor units, e.g., counts, and they are pre-subtracted from the sensor readings. The value for this offset is obtained by finding the value of counts for which you want your user units to read 0.0. This is normally accomplished automatically during a homing procedure.

OUTPUT_SCALE = 1 0

These two values are the scale and offset factors for the axis output to the motor amplifiers. The second value (offset) is subtracted from the computed output (in volts), and divided by the first value (scale factor), before being written to the D/A converters. The units on the scale value are in true volts per DAC output volts. The units on the offset value are in volts. These can be used to linearize a DAC.

Specifically, when writing outputs, the EMC first converts the desired output in quasi-SI units to raw actuator values, e.g., volts for an amplifier DAC. This scaling looks like:

raw = (output - offset) / scale

The value for scale can be obtained analytically by doing a unit analysis, i.e., units are [output SI units]/[actuator units]. For example, on a machine with a velocity mode amplifier such that 1 volt results in 250 mm/sec velocity, we have:

[scale units] = 250 [mm/sec] / 1 [volts] = 250 mm/sec/volt

and, as a result,

amplifier [volts] = (output [mm/sec] - offset [mm/sec]) / 250 [mm/sec/volt]

Note that the units of the offset are in user units, e.g., mm/sec, and they are pre-subtracted from the sensor readings. The value for this offset is obtained by finding the value of your output which yields 0.0 for the actuator output. If the DAC is linearized, this offset is normally 0.0.

The scale and offset can be used to linearize the DACs as well, resulting in values that reflect the combined effects of amplifier gain, DAC non-linearity, DAC units, etc. To do this, follow this procedure:

a. Build a calibration table for the output, driving the DACs with a desired voltage and measuring the result, e.g.,

RAW	MEAS
-10	-9.93
-9 V	-8.83
0	-0.03
1	0.96
9	9.87
10	10.87

b. Do a least-squares linear fit to get coefficients a, b such that

meas = a * raw + b

c. Note that we want raw output such that our measured result is identical to the commanded output. This means

cmd = a * raw + b

raw = (cmd - b) / a

As a result, the a and b coefficients from the linear fit can be used as the scale and offset for the controller directly.

MIN_LIMIT = -1000

The minimum limit (soft limit) for axis motion, in user units. When this limit is exceeded, the controller aborts axis motion.

MAX_LIMIT = 1000

The maximum limit (soft limit) for axis motion, in user units. When this limit is exceeded, the controller aborts axis motion.

MIN_OUTPUT = -10

The minimum value for the output of the PID compensation that is written to the motor amplifier, in volts. The computed output value is clamped to this limit. The limit is applied before scaling to raw output units.

MAX_OUTPUT = 10

The maximum value for the output of the PID compensation that is written to the motor amplifier, in volts. The computed output value is clamped to this limit. The limit is applied before scaling to raw output units.

FERROR = 1.0 MIN_FERROR = 0.010

FERROR is the maximum allowable following error, in user units. If the difference between commanded and sensed position exceeds this amount, the controller disables servo calculations, sets all the outputs to 0.0, and disables the amplifiers. If MIN_FERROR is present in the .ini file, velocity-proportional following errors are used. Here, the maximum allowable following error is proportional to the speed, with FERROR applying to the rapid rate set by [TRAJ] MAX_VELOCITY, and proportionally smaller following errors for slower speeds. The maximum allowable following error will always be greater than MIN_FERROR. This prevents small following

errors for stationary axes from inadvertently aborting motion. Small following errors will always be present due to vibration, etc. The following polarity values determine how inputs are interpreted and how outputs are applied. They can usually be set via trial-and-error since there are only two possibilities. The EMCMOT utility program USRMOT can be used to set these interactively and verify their results so that the proper values can be put in the INI file with a minimum of trouble.

ENABLE_POLARITY = 0

The polarity for enabling the amplifiers. Set this to 0 or 1 for the proper polarity. This value can be determined by following all the electronics back from the amplifier, through any driver circuitry, etc. or it can be set through a simple trial-and-error. Normally, for amplifiers which are enabled active-low (0 volts enables), this is a 0.

MIN_LIMIT_SWITCH_POLARITY = 1

The polarity for detecting minimum-travel hardware limit switch trips. Set this depending on how your switches are wired up to the digital inputs on the I/O board.

MAX_LIMIT_SWITCH_POLARITY = 1

The polarity for detecting maximum-travel hardware limit switch trips. Set this depending on how your switches are wired up to the digital inputs on the I/O board.

HOME_SWITCH_POLARITY = 1

The polarity for detecting homing switch trips. Set this depending on how your switches are wired up to the digital inputs on the I/O board.

HOMING_POLARITY = 1

The direction in which homing moves are initiated. 0 means in the negative direction, 1 means in the positive direction.

FAULT_POLARITY = 1

The polarity for detecting amplifier faults. Set this to 0 or 1 depending upon how the amplifier sets the logic level for its fault condition.

The following entries are used to set the parameters for the DC servomotor simulations. These are only used when running the EMC in simulation.

TORQUE_UNITS = OZ_IN

The units used to interpret subsequent values for ROTOR_INERTIA and DAMPING_FRICTION_COEFFICIENT. This can be OZ_IN for ounce-inches, LB_FT for pound-feet, or N_M for newton-meters.

ARMATURE_RESISTANCE = 1.10

The resistance, in ohms, of the motor.

ARMATURE_INDUCTANCE = 0.0120

The inductance, in henries, of the motor.

BACK_EMF_CONSTANT = 0.0254

The back EMF constant, or torque constant, in volts per radian per second.

ROTOR_INERTIA = 0.0104

The rotor inertia, in torque units * seconds2.

DAMPING_FRICTION_COEFFICIENT = 0.083

The damping coefficient, in torque units per radian per second.

SHAFT_OFFSET = 0

The angular offset, in radians, between the the motor initial position and the encoder initial position. Normally this is 0, but can be made any arbitrary value if the simulated motor shaft position is interpreted as the actual axis position and should be something other than 0 when the encoder reports 0.

REVS_PER_UNIT = 10

The amount of motor shaft revolutions per user unit of position. For example, for a 1/10 inch lead screw, where 10 rotations equals 1 inch, this would be 10. The following entry is used to set the parameters for the amplifier simulations. These are only used when running the EMC in simulation.

AMPLIFIER_GAIN = 1

The gain of the amplifier, which multiplies the input voltage to generate an output voltage which drives the motor.

MAX_OUTPUT_CURRENT = 10

The maximum output current of the amplifier, in amps.

LOAD_RESISTANCE = 1.10

The resistance, in ohms, of the load on the amplifier. This is normally the same as the motor armature resistance, but it may not be, for example, if there is additional resistive load between the amplifier and the motor itself. The following entry is used to set the parameters for the encoder simulations. These are only used when running the EMC in simulation.

COUNTS_PER_REV = 4096

The number of encoder counts per motor shaft revolution. If there is gearing between the encoder shaft and the motor shaft, this value should include this.

[EMCIO] section. The [EMCIO] section contains control values and setup parameters for the digital and analog I/O points in EMCIO.

The following entries set general parameters for the I/O controller.

PLAT = nonrealtime

Only one choice, nonrealtime.

EMCIO = minimillio

Name of the IO controller program. Valid options are :

bridgeportio Used with bridgeporttask in [TASK] section.

minimillio Used with minimilltask in [TASK] section.

ppmcio Used with bridgeporttask in [TASK] section.

simio Can be used with either minimilltask or bridgeporttask.

CYCLE_TIME = 0.100

The period, in seconds, at which EMCIO will run. You can make this as small as you want to increase the throughput. Making it 0.0 or a negative number will tell EMCIO not to sleep at all. Ultimately the system loading will limit the effective throughput.

TOOL_TABLE = tool.tbl

The file which contains tool information. The format of the file is

POC	FMS	LEN	DIAM	COMMENT
1	1	0	0	
2	2	0	0	

where the first line is a comment (in this case the name of the columns), and the subsequent lines contain the pocket number in which the tool is located, the tool ID of the tool itself, the length, the diameter, and an optional comment. The length and diameter are in user units.

The following entries set parameters for spindle control.

IO_BASE_ADDRESS = 0x278

Base address for the port used for IO.

PARPORT_BASE_ADDRESS = 0x278

Obsolete since July 2002 in the CVS sources, and BDI-2.2.18. Use IO_BASE_ADDRESS instead.

SPINDLE_OFF_WAIT = 1.0

How long, in seconds, to wait after the spindle has been turned off before applying the brake.

SPINDLE_ON_WAIT = 1.5

How long, in seconds, to wait after the spindle brake has been released before turning the spindle on.

The following entries set the bit indices for the digital I/O so that the controller knows the mapping to I/O point wiring. The indices start at 0 for the least significant bit in the digital I/O map. See Setting up the External Interfaces for information on interfacing I/O boards to the software.

ESTOP_SENSE_INDEX = 1

The location of the input bit which is used to detect whether the system is in ESTOP.

LUBE_SENSE_INDEX = 2

The location of the input bit which is used to detect whether the lubrication level is OK or low.

SPINDLE_FORWARD_INDEX = 1

The location of the output bit which is used to drive the spindle forward. Only applicable to manual spindles.

SPINDLE_REVERSE_INDEX = 0

The location of the output bit which is used to drive the spindle in reverse. Only applicable to manual spindles.

MIST_COOLANT_INDEX = 6

The location of the output bit which is used to turn mist coolant on or off.

FLOOD_COOLANT_INDEX = 7

The location of the output bit which is used to turn flood coolant on or off.

SPINDLE_DECREASE_INDEX = 8

The location of the output bit which is used to decrease the spindle speed. Only applicable to manual spindles.

SPINDLE_INCREASE_INDEX = 9

The location of the output bit which is used to increase the spindle speed. Only applicable to manual spindles.

ESTOP_WRITE_INDEX = 10

The location of the output bit which is used to cause an ESTOP.

SPINDLE_BRAKE_INDEX = 11

The location of the output bit which is used to engage or release the spindle brake.

The following entries set the polarities for the digital I/O points. These can be set by trial-and-error, or by noting the levels and any inverting done by the electronics between the sensors and actuators and the electronics.

ESTOP_SENSE_POLARITY = 1

The polarity of the sensed estop input bit.

LUBE_SENSE_POLARITY = 1

The polarity of the sensed lube level bit.

SPINDLE_FORWARD_POLARITY = 0

The polarity of the sensed spindle forward bit.

SPINDLE_REVERSE_POLARITY = 0

The polarity of the sensed spindle reverse bit.

MIST_COOLANT_POLARITY = 0

The polarity of the sensed mist coolant bit.

FLOOD_COOLANT_POLARITY = 0

The polarity of the sensed flood coolant bit.

SPINDLE_DECREASE_POLARITY = 1

The polarity of the sensed spindle decrease bit.

SPINDLE_INCREASE_POLARITY = 1

The polarity of the sensed spindle increase bit.

ESTOP_WRITE_POLARITY = 1

The polarity of the sensed estop activation bit.

SPINDLE_BRAKE_POLARITY = 0

The polarity of the sensed spindle brake bit.

[EMCSERVER] section

EMCSERVER = emcsvr

Name of NML server, e.g., emcsvr; if not found then none will run.

[EMCSTRIP] section. Section for emc stripchart parameters.

EMCSTRIP = emcstripchart

Name of strip chart display program e.g. emcstripchart, if not found then none will run.

OPTIONS = -f emcstrip.conf.ferror

Options for emcstripchart ussually -f something.conf. This file says which variables to plot, colors etc. -u changes the update rate.

9.1

Chapter 10

EMC Stripchart

Stripchart can be a valuable tool for the machine integrator because it can show many of the EMC variables in much the same way that an external oscilloscope will.

The EMC stripchart program is a modified version of GNOME stripchart program "gstripchart". This document is a modified copy of the GNOME stripchart documentation. For the original source code and/or documentaton goto www.gnome.org.

10.1 Setting up Stripchart.

(Need to add required packages and setup and operating procedures here)

10.2 EMC Modifications

The following modifications were made to gstripchart program to turn it into emcstripchart.

- The "nml" configuration file option was added.
- The "emcmot" configuration file option was added.
- The "minimum" configuration file option was more completely implemented.
- An additional update timeout was added. This allows the data to be polled more often than display is refreshed. The text display is updated by a timer not just by clicking on it.
- The buttons "pause" and "go" were added. (They only pause and restart the plotting to give you more time to look at something, the EMC is not paused.)
- The "bottom" column was added. When the scale is readjusted, if the starting the starting minimum equaled the negative of the starting maximum the scale is modified so that it will maintain this symmetry about zero.
- Several of the defaults were changed.

- The bindtextdomain code was disabled.
- The original set of Makefile stuff using automake and autoconf was trashed to use a more EMC/RCS style makefile.
- Adds the -u –update-interval command line option.
- Linked in stripnmli.cc and stripemcmoti.cc and the RCS and EMC libraries.

10.3 Hints for EMC users.

Click on the graph with the left mouse button to bring up the "key". The horizontal scale is indicated with the little black tick marks. The distance between ticks is aproximately 1 second. The vertical scale is different for each variable. The top is the value that variable would have to have to be plotted at the top of the scale, the bottom is the value the variable would have to have to be plotted at the bottom. Initially the top and bottom equal the maximum and minimum from the configuration file, however if the maximum or minimum are exceeded the scale will change to keep the plot in the visible part of the graph. The file emcstrip.conf.ferror configures emstripchart to plot the following errors and high marks for the following errors. The normal build will skip this utility, to build it cd emc/src/stripchart and run "make". To make generic.run or ?.run start it every time you start EMC, base your run script on a version of generic.run that is newer than April 1, 2000 and add the following section to your .ini:

```
; section for emc stripchart parameters------------
----------
[EMCSTRIP]
EMCSTRIP = emcstripchart
OPTIONS = -f emcstrip.conf.position
```

10.4 GNOME Stripchart Documentation

The GNOME stripchart program charts various user-specified parameters as a function of time. Its main use is to chart system performance parameters such as CPU load, CPU utilization, network traffic levels, and the like. Other more ingenious uses are left as an exercise for the interested user.

The gstripchart program periodically reads data from a file, extracts a value, and displays these values in one of several formats. The default format is a graphical display similar to that of a stripchart recorder. Hence the name, "gstripchart".

On systems such as Linux, in which the system parameters are available in human-readable form in the /proc directory, the gstripchart program makes a dandy performance monitoring tool, similar to but more versatile than xload.

Instead of being limited to a few standard performance parameters, the gstripchart program can plot any time-variant parameter than can be read from a file or pipe. This ability to read data from a pipe provides a very versatile and easy to use method of setting up custom displays.

The gstripchart program determines the parameters to display by reading a configuration file. The gstripchart program will first look for a configuration file specified on the command line, then look for a file named gstripchart.conf in the current directory, then look for a file named .gstripchart.conf in the users home directory, then look for a file named /etc/gstripchart.conf.
If no configuration file is found, the program is terminated.

10.5 Options

There are a few command line switches that can be used to alter the behavior of the program.

-f, –config-file=FILE configuration file

-g, –geometry=GEOMETRY geometry

-i, –chart-interval=SECS chart update interval

-I, –chart-filter=SECS chart low-pass filter time constant

-j, –slider-interval=SECS slider update interval

-J, –slider-filter=SECS slider low-pass filter time constant

-M, –menubar add menubar

-S, –omit-slider omit slider

-t, –display-type=TYPE type of display none: no display is produced (for debugging); text: a textual numeric display is produced; graph: a textual graphic display is produced; gtk: use the default gtk-based graphic display. –class=CLASS FIXME –display=DISPLAY X display to use –gxid_host=HOST FIXME –gxid_port=PORT FIXME –name=NAME FIXME –no-xshm Don't use X shared memory extension –xim-preedit=STYLE FIXME –xim-status=STYLE FIXME

-u, –update-interval=SECS update interval to poll data

-?, –help Give this help list –usage Give a short usage message

-V, –version Print program version

A more detailed description of some switches follows.

configuration file Specifies a file from which to read configuration information. If unspecified, the current working directory is checked for a file named "gstripchart.conf", the user's home directory is checked for a file named ".gstripchart.conf", and the /etc directory is checked for a file named "gstripchart.conf". The first such file found is used.

geometry A standard X11 geometry specification of the form WxH+X+Y.

chart-interval

slider-interval Specifies the time interval in seconds between updates to the chart window and slider window. If unspecified, the chart window will be updated every 5 seconds and the slider window will be updated every 0.2 seconds.

chart-filter

slider-filter Specifies the time constant in seconds to be used in low-pass filtering the data displayed in the chart or slider windows. A time constant of 0 seconds turns low-pass filtering off, which can result in a jumpy display. A time constant in the same range as the interval parameter, described above, is usually a good choice. Much larger values cause display updates to become sluggish. If unspecified, no low pass filtering is performed in either window.

menubar Adds an application menubar to the main window. Normally this is omitted, and the menu is popped up by right-clicking on the chart window.

omit-slider Causes the display of the slider window to be suppressed.

10.6 Configuration

The configuration file has a paragraph of configuration information for each parameter to be ploted. Each of these paragraphs are comprised of a series of RFC-822 style "keyword: value" pairs, beginning with an "identifier:" line. A comment can be included by putting a sharp sign (#) in the first column of a line.

The following keywords are available. Some are optional; some are only used by certain display types; many have reasonable default values, as described below.

identifier: Introduces a parameter definition, and assigns a name to the parameter. This line *must* be the first line of a parameter description. active: If a parameter is marked "active = no", it will be ignored. This provides a convenient way to disable a parameter without deleting it from a parameter file.

id_char: Provides a single-character abbreviation for a parameter. Currently unused, this is intended for the non-existant character-graphics display mode.

color: Determines the color to be used in displaying a parameter. The color names and their RGB values are taken from X11/rgb.txt.

filename: The file from which a parameter value is read. When a filename beginning with a "|" is supplied, input lines will be read from a pipe.

pattern: The pattern which identifies the line from which a parameter value is to be extracted. If no pattern is provided, the first line of the file is used.

fields: The number of fields to be split out of the first line which matches the pattern. Splitting is done on whitespace.

equation: An equation used to obtain the value to be ploted for this parameter. maximum: The largest value that can be displayed. Any value in excess of the maximum will be plotted at the top of the display. If omitted, a default value of 1.0 is used.

minimum: The smallest value that can be displayed. Any value less than the minimum will be plotted at the bottom of the display. If omitted, a default value of 0.0 is used.

nml: The name of the variable to read from NML to use for this plot. If an equation is specified this variable becomes keyword one or $1. Adding this keyword makes the filename and pattern keywords irrelavant. See the list of variables available.

emcmot: The name of the variable to read from EMCMOT to use for this plot. If an equation is specified this variable becomes field one or $1. Adding this keyword makes the filename and pattern keywords irrelavant. See the list of variables available.

On each iteration, a value to be displayed is obtained for each parameter in the configuration file. The file named in the "filename" line is opened – either as a pipe if the filename begins with a pipe character (|), or as a regular file otherwise – and a line is read.

If a pattern was specified, lines are read until one is found that contains the pattern string anywhere in the line. This line is split into the number of whitespace seperated fields specified in the "fields" line. Each of these fields is interpreted as a floating point number.

A value is obtained by evaluating the "equation" line using these field values. The first (or only) value is denoted by $1, the next by $2, and so forth. The difference between the field values between the last and the current iteration is denoted by ~1, ~2, and so forth. The elapsed time in seconds between the last and current iteration is ~t. The requested update interval is $i (and the delta is ~i, but will always be zero). All the usual infix arithmatic operators are available.

10.7 libgtop

If libgtop support has been compiled into the gstripchart program, a value can be obtained from this library. This provides a portable method of obtaining many system performance parameters. These are all signed long integer quantities, except for uptime, idletime, and the three loadavg values which are floating point values.

The following libgtop parameters are available:

10.7.1 CPU Statistics

cpu_total, cpu_user, cpu_nice, cpu_sys, cpu_idle, and cpu_freq

10.7.2 Memory Statistics

mem_total, mem_used, mem_free, mem_shared, mem_buffer, mem_cached, mem_user, mem_locked

10.7.3 Swap Statistics

swap_total, swap_used, swap_free, swap_pagein, swap_pageout

10.7.4 Uptime Statistics

uptime, idletime

10.7.5 Loadavg Statistics

loadavg_running, loadavg_tasks, loadavg_1m, loadavg_5m, loadavg_15m

10.7.6 Network Statistics

net_pkts_in, net_pkts_out, net_pkts_total, net_bytes_in, net_bytes_out, net_bytes_total, net_errs_in, net_errs_out, net_errs_total

Note that the network statistics don't use the libgtop library. Instead, the values are read directly from /proc/net/dev, and so are only available under Linux.

10.8 NML Variables

This is the list of variables that can be read from NML. The array_index should be replaced with an appropriate integer.

10.8.1 Task

- task.heartbeat,
- task.mode
- task.state,
- task.execState,
- task.interpState,
- task.motionLine,
- task.currentLine,
- task.readLine,

10.8.2 Motion

- motion.heartbeat,
- motion.traj.linearUnits,
- motion.traj.angularUnits,
- motion.traj.cycleTime,
- motion.traj.axes,
- motion.traj.mode,
- motion.traj.enabled,

- motion.traj.inpos,
- motion.traj.queue,
- motion.traj.activeQueue,
- motion.traj.queueFull,
- motion.traj.id,
- motion.traj.paused,
- motion.traj.scale,
- motion.traj.position.tran.x,
- motion.traj.position.tran.y,
- motion.traj.position.tran.z,
- motion.traj.actualPosition.tran.x,
- motion.traj.actualPosition.tran.y,
- motion.traj.actualPosition.tran.z,
- motion.traj.velocity,
- motion.traj.acceleration,
- motion.traj.maxVelocity,
- motion.traj.maxAcceleration,
- motion.axis[array_index].units,
- motion.axis[array_index].p,
- motion.axis[array_index].i,
- motion.axis[array_index].d,
- motion.axis[array_index].ff0,
- motion.axis[array_index].ff1,
- motion.axis[array_index].ff2,
- motion.axis[array_index].backlash,
- motion.axis[array_index].bias,
- motion.axis[array_index].maxError,
- motion.axis[array_index].deadband,
- motion.axis[array_index].cycleTime,
- motion.axis[array_index].inputScale,
- motion.axis[array_index].inputOffset,
- motion.axis[array_index].outputScale,
- motion.axis[array_index].outputOffset,

- motion.axis[array_index].minPositionLimit,
- motion.axis[array_index].maxPositionLimit,
- motion.axis[array_index].minOutputLimit,
- motion.axis[array_index].maxOutputLimit,
- motion.axis[array_index].maxFerror,
- motion.axis[array_index].minFerror,
- motion.axis[array_index].homingVel,
- motion.axis[array_index].homeOffset,
- motion.axis[array_index].enablePolarity,
- motion.axis[array_index].minLimitSwitchPolarity,
- motion.axis[array_index].maxLimitSwitchPolarity,
- motion.axis[array_index].homeSwitchPolarity,
- motion.axis[array_index].homingPolarity,
- motion.axis[array_index].faultPolarity,
- motion.axis[array_index].setpoint,
- motion.axis[array_index].ferrorCurrent,
- motion.axis[array_index].output,
- motion.axis[array_index].input,
- motion.axis[array_index].inpos,
- motion.axis[array_index].homing,
- motion.axis[array_index].homed,
- motion.axis[array_index].fault,
- motion.axis[array_index].enabled,
- motion.axis[array_index].minSoftLimit,
- motion.axis[array_index].maxSoftLimit,
- motion.axis[array_index].minHardLimit,
- motion.axis[array_index].maxHardLimit,
- motion.axis[array_index].overrideLimits,
- motion.axis[array_index].scale,
- motion.axis[array_index].ferrorHighMark,

10.8.3 IO

- io.heartbeat,
- io.cycleTime,
- io.tool.toolPrepped,
- io.tool.toolInSpindle,
- io.spindle.speed,
- io.spindle.direction,
- io.spindle.brake,
- io.spindle.increasing,
- io.spindle.enabled,
- io.coolant.mist,
- io.coolant.flood,
- io.lube.on,
- io.lube.level,
- io.aux.estop,
- io.aux.estopIn,
- io.aux.dout[array_index],
- io.aux.din[array_index],
- io.aux.aout[array_index],
- io.aux.ain[array_index]

10.9 EMCMOT Variables

This is the list of variables that can be read from emcmot.

- heartbeat,
- computeTime,
- pos.tran.x, pos.tran.y, pos.tran.z,
- axisPos[0], axisPos[1], axisPos[2],
- ferrorCurrent[0], ferrorCurrent[1], ferrorCurrent[2],
- ferrorHighMark[0], ferrorHighMark[1], ferrorHighMark[2],
- output[0], output[1], output[2],
- input[0], input[1], input[2],
- actualPos.tran.x, actualPos.tran.y, actualPos.tran.z,

- id,
- depth,
- activeDepth,
- queueFull,
- motionFlag,
- axisFlag[0], axisFlag[1], axisFlag[2],
- axisPolarity[0], axisPolarity[1], axisPolarity[2],
- paused,
- overrideLimits,
- tMin, tMax, tAvg,
- sMin, sMax, sAvg,
- nMin, nMax, nAvg

Chapter 11

Remote GUIs

Running the GUI on a separate computer - From Will Shackleford

EMC usually consists programs from 4 categories:

GUI – (Graphical User Interface) xemc, yemc, tkemc, emcJavaGui, fpanel, etc. TASK – mmtask, shvtask, bridgeportask etc. MOTION – emcmotsim, steppermod.o, shavermod.o, minimod.o, etc. IO – mmio, shvio, simio, bridgeportio, etc.

You generally need to run at least one program from each category to get a working EMC, which to me would mean all 4. Most of the programs in the motion category are built as real-time kernel modules, and at the moment do not communicate using NML directly.

The most common way to split EMC across a network would be to put the TASK,MOTION and IO on computer1 and the GUI on computer2.

To do this I would find the .nml file used for my setup. It is usually the same as the .run file. The file needs to be changed with a text editor in two ways.

1. The host is listed as "localhost" for almost every thing which is fine if everything runs on the same computer but to run across TCP we'll need the real host names. I usually use a global search and replace of "localhost" to "myhost". You can also use IP address directly if you don't have host tables or a name server setup.

2. You need to change the process type flag for the appropriate process from LOCAL to REMOTE.

Then just copy over the .nml file, .ini file and the program you want to run on to computer2.

You need to start emcsvr on the host computer, The computer listed in the buffer lines (which start with B) before starting any remote programs. Attached is the file with xemc on computer2 and everything else on computer1.

When everything is finally run xemc will send messages via TCP to emcsvr which will then be placed in shared memory buffers to be read by IO and TASK. TASK will then forward messages as necessary to the MOTION kernel module.

Scott Stephens wants to run the TkEMC on a Windows PC. Here are the steps to do it, some of which may be confounding you all since the Windows zip/archive file may be out of date, unclickable, or whatever.

1. Get the EMC up and running on your Linux/RT Linux computer, controlling your machine.

2. In the .ini file for the EMC on the Linux/RT computer, make sure that the EMC server is uncommented, e.g.,

```
; section for external NML server parameters
---------------------------------
[EMCSERVER]
; Platform for NML server, e.g., nonrealtime
PLAT =                nonrealtime
; Name of NML server, e.g., emcsvr; if not found then none will run
EMCSERVER =           emcsvr
```

The emcsvr runs on the Linux/RT Linux computer. It opens connections to the EMC command and status buffers on the Linux/RT Linux computer, listens for network connections from remote clients (like the TkEMC on a Windows PC), and acts as a go-between.

3. Get the Windows version of the TkEMC and the emcsh program. These should be precompiled and zipped up along with a .nml file.

4. Edit the .nml file, near the top, and put the name of the Linux/RT Linux computer in the buffers' host name field, e.g.,

```
# Top-level buffers to EMC
B emcCommand     SHMEM   linuxpc.mydomain       8192    0    0    1    12 1001 TCP=5005 xdr diag
B emcStatus      SHMEM   linuxpc.mydomain       8192    0    0    2    12 1002 TCP=5005 xdr diag
B emcError       SHMEM   linuxpc.mydomain       8192    0    0    3    12 1003 TCP=5005 xdr diag queue
```

There are three entries, which I've put as "linuxpc.mydomain" in this example. This tells the TkEMC, which will be running on a Windows box, where to look for the EMC.

5. Edit the .nml file again, near the bottom, and make sure that the "xemc" process is connecting remotely, using the REMOTE access keyword,

e.g.,

```
P xemc        emcCommand      REMOTE  localhost      W      0        10.0    0      10
P xemc        emcStatus       REMOTE  localhost      R      0        10.0    0      10
P xemc        emcError        REMOTE  localhost      R      0        10.0    0      10
P xemc        toolCmd         REMOTE  localhost      W      0        10.0    0      10
P xemc        toolSts         REMOTE  localhost      R      0        10.0    0      10
```

You can leave "localhost" alone. This is the name of the Windows PC, but it's not used right now so it's mostly for documentation.

6. Start the EMC on the Linux/RT Linux box.

7. Run the TkEMC on the Windows box. It should connect across the network, and bring up a window showing the status of the EMC. You can use either the TkEMC on the Linux box, or the remote one, or both. Try one and watch the other update.

–Fred

11.1 Remote EMC with BDI

11.1.1 Introduction

This short paper describes the basics of getting EMC working across a network. Some initial assumptions are made with respect to the computers in the system.

- EMC is installed and configured on both computers.

- The computers are connected by a network that has been tested.
- The IP addresses and/or names are known

Much of the existing documentation refers to the two computers used as Remote and Local - This can lead to a fair degree of confusion when consulting with the experts. From here on, the terms Server and Client will be used.

- SERVER - The computer running the real time components of EMC.
- CLIENT - The computer running the Graphical User Interface.

BDI-2.14 contains copies of tcl-8.2 and emcsh.exe along an assortment of files required to run a remote GUI. All of these can be found in the Windows directory of the BDI CD along with a tarball of sample ini, nml, and run files used to run EMC. These files can also be found on the second CD in the BDI-TNG set.

11.1.2 Initial Tests

Before we start, some information is needed about the two computers to be used if not already known. First is the IP address of the Server which can be found by using ifconfig, or by pinging the intended client.

```
[paul@Talia paul]$ ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2) from 192.168.0.5 :  56(84) bytes of data.
64 bytes from 192.168.0.5:  icmp_seq=0 ttl=255 time=87 usec
```

Using ping has the dual purpose of checking the network is operating and also provides the IP addresses of the two computers to be used. From the above, the EMC server will be at 192.168.0.5, and the client at 192.168.0.2.

11.1.3 Setting up the Server

A couple of minor changes need to be made to emc.ini in the following two sections :

```
; General section -----------------------------------------------------------
[EMC]
; Name of NML file to use, default is emc.nml
NML_FILE = emc.nml

; section for external NML server parameters --------------------------------
[EMCSERVER]
; Name of NML server, e.g., emcsvr; if not found then none will run
EMCSERVER = emcsvr
```

The emc.nml now needs to be set up to allow the remote client GUI to “connect”:

```
P xemc          emcCommand      REMOTE  192.168.0.2     W       0       10.0    0       10
P xemc          emcStatus       REMOTE  192.168.0.2     R       0       10.0    0       10
P xemc          emcError        REMOTE  192.168.0.2     R       0       10.0    0       10
P xemc          toolCmd         REMOTE  192.168.0.2     W       0       10.0    0       10
P xemc          toolSts         REMOTE  192.168.0.2     R       0       10.0    0       10
```

All the other entries can be left as the default ’LOCAL localhost’ or ’SHMEM localhost’.

11.1.4 Setting up the Client

Now for the fun part - Getting the GUI to work on the remote or client computer. Check the emc.nml file for any occurrences of LOCAL and localhost and change them to REMOTE and your server IP address.

To test that it all works, EMC must first be started on the server with the usual ./emc.run command. On the client, first change to the EMC directory and use the following command:

```
plat/nonrealtime/bin/tkemc
```

All being well, tkemc should connect with EMC running on the server and give you full control over the machine. A few notes from some initial tests...

- If the IP addresses are resolved either from a DNS server or the local /etc/hosts files, then names can be used instead.
- EMC must be run as root on the server.
- The client GUI can connect as a user - It does not require root permissions.

11.1.5 Problems and Other Issues

One immediate problem that I have found with running EMC and Tkemc on different computers is with the emc/programs directory. When opening a file, Tkemc displays the local file system and then tells EMC to open the file based on the client directory tree. Should the file not exist on the server, or the directory tree differs, then an error is generated. One fix for this is to mount one common NFS directory in the same place on both computers.

By default, the client tkemc uses emc.nml and emc.ini when starting up. If the client computer is also used for running EMC locally, you may want to consider having a copy of the programs in a seperate directory. With some experimentation with ini parameters, it sould be possible to have two configurations co-existing in the same directory.

One area of particular concern in this day and age of network security - The EMC server will accept connections from any client on port 5005. If you are running the EMC server on a large network or connected to the internet, then you must look very carefully at the security risks. To have a remote user connect and turn the spindle on whilst you have your fingers in the way should be incentive enough.

11.1.6 EMC on a Microsoft Windows Computer

The final goal of my experiments with a remote GUI was to demonstrate a Windows computer in control at NAMES 2002. After downloading tkemc.exe from the NIST web site, matching versions of tcl and tk binaries were installed. The self extracting tkemc archive failed to do so at the first attempt, so another copy was downloaded, this time to be extracted successfully.

Samba will probably have to be set up to allow file sharing in order to prevent problems with opening G-code files. I'm not sure how well the use of '\' by DOS or Windows will convert to '/' used by linux.

With the client emc.nml file edited and the -queryhost flag removed from emc.bat, a connection was made with the EMC server. A serious problem has been found with the GUI in use - The X Y Z data is not being displayed correctly, in fact "Feed Override" appears to be used for one of the three axis ! The colours of the axis digits are also out of sync with reality, X is red, Y, green, and Z is shown as yellow. Most odd.

After persuading Don McLean to compile emcsh.exe against the current sources (24th March 2002), using Microsoft Visual C++, another attempt was made. This time all the digits responded to the correct axis and the machined could be homed and jogged. Once again, differences in the operating systems caused problems. This time with the way Tcl/Tk and the underlying OS handles paths. Linux will use /emc/programs/foo.ngc, and Windows, C:\emc\programs\foo.ngc. EMC expects the former format to locate and load a G-code file, and the latter just causes an error. For file browsing to work correctly, the emc/programs directory will have to be "shared" with the aid of Samba which will add further to the problem of path names.

In conclusion, Whilst operating EMC across a network is possible, using a Windows computer will require some editing of Tkemc to change the way file and path names are handled.

Appendix A

EMC FAQ

EMC FAQ is common effort of many individuals around the globe. Most of the FAQ has been collected from various postings at EMC-mailing list, emc@nist.gov `mailto:emc@nist.gov`. You can send your additions to this FAQ to FAQ-maintainer, Henry Palonen `mailto:emcfaq@yty.net`.

FAQ is available in HTML `http://www.linuxcnc.org/handbook/faq/`, PS `faq.ps`, PDF `faq.pdf`, RTF `faq.rtf`, SGML `faq.sgml` (DocBook) and LyX `faq.lyx` (LaTeX) formats. RTF, PDF and PS are perhaps most suitable for printing, while HTML is most suitable for searching / browsing the FAQ. LyX and SGML - formats are the "source code" formats where every other format are made from.

A.1 What is EMC ?

EMC stands for Enhanced Machine Controller. It's free Open Source software for controlling CNC machines and robotics. It's run on Linux-operating system with certain "real-time-modifications" applied.

A.2 How do I set the steps per unit for each axis?

This is done in the ".ini" file at your installation directory. INPUT_SCALE and OUTPUT_SCALE are the number of pulses for each unit as defined in the UNITS-parameter. UNITS should be "1" if you are operating with millimeter as your unit and " 0.03937007874016" for inch.

You can calculate needed INPUT_SCALE and OUTPUT_SCALE with following equation:

INPUT_SCALE = STEPS_NEEDED_TO_GO_ONE_REVOLUTION / LENGTH_OF_TRAVEL_IN_ONE_RE

So, for example if you have setup where milling head travels 1.5mm per one whole revolution, and your steppers require 400 pulses to go one revolution:

INPUT_SCALE = 400 / 1.5

so, in ".ini"-file we would write

INPUT_SCALE = 266.66666666667 0.000

If you are using “freqmod.o” as your motion module (eq. if you are using servos instead of steppers), you should write same number to INPUT_SCALE and OUTPUT_SCALE:

INPUT_SCALE = 266.66666666667 0.000

OUTPUT_SCALE = 266.66666666667 0.000

(I’m not sure where the second number “0.000” is used)

Appropriate way to think of these is that the input scale represents how many pulses you need to input to the motor amp to get one unit of motion. Output scale is the number of pulses the encoder will send for one unit of motion. Motor or axis in and out rather than PC in or out.

You can also calculate necessary values with a calculator `http://206.19.206.56/StepPerIn.asp`.

A.3 How does EMC handle home and limit switches ?

EMC provides inputs for Home, Low limit and High limit switches. These are fed into pins 12, 13 and 15 respectively of the principal parallel port. Because there is only one input for each purpose and yet we have at least two or three axes of motion, similar switches for all axes must be ganged together, i.e. all the Home switches are grouped together and fed into pin 12 etc.

It is normal for limit switches to be connected in the ’normally closed’ mode so that, if the switch fails, it will do so in a safe manner (and warn you that something is wrong). There is no definite convention for Home switches but, here again, it would make sense to connect this in the normally closed mode so that failure will be obvious. In this mode, the switches from all the axes should be connected in Series (i.e. in a chain with one terminal of one switch connecting to one of the next switch and the second terminal of this switch connecting to the next switch), thus, when any switch is activated it will open and break the chain.

If the switches are connected in the Normally Open mode, they will have to be connected in Parallel (i.e. the two terminals of one switch connecting directly to both the corresponding terminals of the next switch). When any of the switches are activated in this mode, they will close and provide a current path directly to the port pin. Depending on which mode is chosen, the relevant entry in the INI file will have to be set accordingly. To ensure positive operation of the switches, it is a good idea to provide a ’Pull-up’ voltage onto the port to make sure it defaults to a definite state. This voltage can conveniently be derived from any 5 volt supply, perhaps on the motor drive circuitry, or it can be provided separately from a small transformer and regulator (6 volt AC transformer feeding a small bridge rectifier and then to a 78L05 regulator with a 0.1uf capacitor across the input terminals and a 22uf tantalum capacitor across the output terminals.) Whatever voltage source is used it should be fed direct to pins 12, 13 and 15 via 5K6 resistors to limit the current to about 1mA. Do not forget to connect the negative side of the supply to the ground line of the parallel port (pins 18-25). The limit switches are then wired between pins 12, 13, 15 and the ground pins. Make sure that the cable you are using to connect from the equipment to the computer’s parallel port has all the

relevant pins connected - don't make the same mistake as I did at first and assume that a cable having 25 pin 'D' plugs on each end will have all pins connected, it is quite likely that this type of cable will be designed for serial use and will only have about half a dozen wires connected. It will drive the three axes OK but will not work with any limit switches. Where switches are connected in the Normally Closed mode (series), the settings in the INI file should be as follows:

MIN_LIMIT_SWITCH_POLARITY = 1

MAX_LIMIT_SWITCH_POLARITY = 1

HOME_SWITCH_POLARITY = 1

Where the switches are connected in Normally Open mode (parallel), the settings should be:

MIN_LIMIT_SWITCH_POLARITY = 0

MAX_LIMIT_SWITCH_POLARITY = 0

HOME_SWITCH_POLARITY = 0

The other settings which needs checking are the HOMING_POLARITY to send the axes in the right direction and the FAULT_POLARITY which will be 0 for Normally Closed switches and 1 for Normally Open switches. These settings need to be altered for all the axes of course.

To use the Home Switches with a stepper motor setup it will be necessary to make an entry in the INI file under HOME_OFFSET. This needs to be only a large enough distance to move the axis off the home switch (say 0.1" or 1mm). In use, one would select one of the axes and press the HOME button on the EMC screen (only in MANUAL mode). This will send that axis in the direction preset in the INI file as above and, when the Home switch is activated, the axis will stop, then move by the distance in HOME_OFFSET (the direction of which can be changed by altering the sign if needed) and will then stop and reset the axis display to 0.0000, changing the figures to green at the same time. Then another axis can be selected and Homed in a similar manner.

When any of the limit switches are activated, EMC will stop the machine and go into an ESTOP RESET state and also bring up an 'Error Box'. This tells you that a high or low limit has been activated and requires you to acknowledge the message before you can again turn the 'MACHINE ON'.

Types of switch: The most common type of switch for limit switch use is probably the mechanical microswitch. This is normally made as a single-pole change-over switch and can thus be used as either a normally open or normally closed switch. For machine environments, particularly on larger machines, it is probably a good idea to use the type which has a sealed protective enclosure. The microswitch should not be mounted directly in line with the carriage of the machine as any failure will cause the machine to crush the switch. Instead, it should be mounted to one side and activated by some form of arm which slides over the switch operating button, pressing it as it passes. This will prevent damage to the switch and allow you to mount the Home switch anywhere you wish on the axes (do remember, however, that you have only preset the axis to travel in one direction to seek the Home switch!). An alternative to the mechanical microswitch is the optical 'vane in a slot' type of switch which is found in many types of equipment. In this, a small plastic 'U'-shaped molding has an infra-red LED in one leg facing a photocell in the other. Passing a small opaque vane between the legs of the sensor interrupts the beam and this

can be used as a non-contact type of limit or home switch. Power needs to be supplied continuously to the LED and a little electronic circuitry is required to amplify the signal from the photocell but this does provide a rugged and very accurate means of detecting limits etc.

To make the Home switch even more accurate it is possible to 'gate' the switch with either the power to one pole of a stepper motor or with a signal produced by a simple, one-hole encoder on the motor shaft. By passing both these signals through an 'AND' gate (or 'NAND' gate) before feeding the result into the 'Home' pin of the parallel port, the motor can be made to always stop on exactly the same step. so guaranteeing the best accuracy possible.

Ian W. Wright

A.4 What do I need to do to run EMC as a user?

In a recent post, Jan raised the issue of security concerning the EMC software on a shop floor machine. IMO this issue is common to all PC based systems but since parts of the EMC must run as root, and now all of it runs as root in a standard setup, the system is extra vulnerable.

If you need to protect the system against inadvertent changing or removal of files, it's fairly easy. Protecting against determined hackers is more of a problem. Easy way is to set up a normal Linux user account, say "ray", with a home directory, say /home/ray. Ray won't be able to run the EMC for three reasons:

1. it requires running the /sbin/insmod, /sbin/rmmod, and /sbin/lsmod programs to install/remove/list the EMC motion controller
2. it requires accessing /dev/mem to use the shared memory interface
3. it requires running privileged inb/outb instructions for the parallel port IO.
4. it requires privileges to mbuff-device

You can get around the first problem by changing the permissions on /sbin/insmod and /sbin/rmmod so that they are "setuid root". This means that they run as if root is running them. Set this up, as root, by doing:

chmod u+s /sbin/insmod /sbin/rmmod /sbin/lsmod

Now, anyone can run insmod. So, for example, a talented programmer could write his own kernel module that ran through the file system looking for protected user files; remove files; etc. Writing a kernel module is not something you do inadvertently.

You can do the same sort of thing with /dev/mem, by making it read-write for everyone. As root, do:

chmod a+rw /dev/mem

Now, anyone can access /dev/mem and access Linux memory directly. This isn't something that can be done inadvertently. You can set up Unix pipes to write 0's into memory and clobber Linux, but you can also flip the power switch on the front.

You can change the permission on emc/plat/whatever/bin/bridgeportio so that it runs setuid root also. As root, do:

chmod u+s /usr/local/nist/emc/plat/whatever/bin/bridgeportio

You can then change permissions to mbuff-device:

chmod a+rw /dev/mbuff

Now it runs as root and can execute the privileged inb/outb instructions. There are better ways to set things up using groups of semi-trusted users so not everyone can run the EMC, and so those who can still aren't root. If anyone has any other ideas let me know.

Will's response

Would another possibility be to run the rtlinux stuff and the modules that directly connect to them in some .rc file at startup and then let "ray" run only the user interface? I think that this would mean "ray" would have to completely reboot the machine to restart the main controller, but not have access to the /dev/mem or insmod" Of course if "ray" is truly evil and talented and knows how to reboot in single-user mode with a floppy. The only thing that might prevent tampering would be to lock the PC with the EMC controller on it in a box and force "ray" to use a user interface connected remotely to the EMC controller.

– Will

Fred's post

Date: Mon, 17 Jul 2000 11:39:31 -0400 (EDT)

From: Fred Proctor frederick.proctor@nist.gov

Subject: Re: Secure EMC Run - 2

Ray,

You get this error: "not privileged to access IO– disabling IO" because the user is not root. To make this work, you can make the iosh program "setuid root" so that whoever runs the program effectively runs it as root. To do this, do (as root):

root> chown root emc/plat/linux_2_2_14/bin/iosh

root> chmod u+s emc/plat/linux_2_2_14/bin/iosh

This makes root the owner of the program (it should already be owned by root), and sets the "setuid" bit so that whoever runs the program has the identity of the program's owner, in this case root. If the program is recompiled, I think the setuid bit is cleared so you

have to do it again.

New Problem

Today Sept 25, 2000 I found an additional file that you will have to set to root in order to make the latest stuff run as a common user. That file is .bin/ls and it has to do with passing the value of the running ini file to scripts that run from tkemc. The message that I got when I tried to start the new version was:

running EMC DISPLAY PROGRAM – tkemc...

Error in startup script: /bin/ls: tcl/scripts: Permission denied

while executing "exec /bin/ls $scriptdir"

(file "plat/linux_2_2_14/bin/tkemc" line 570)

To fix this I ran the following command as root.

root> chmod u+s /bin/ls

That took care of it. You will need to add this file permission for new releases after the Aug release.

Ian's comment

Will Shackleford wrote:

> The only thing that might prevent tampering would be to lock the PC with the EMC controller on it in a box and force "ray"

> to use a user interface connected remotely to the EMC controller.

It must be something to do with the name - I had two 'Rays' work for me and I couldn't trust either if them not to fiddle with anything they could get their hands on! One was a positive Jonah but, fortunately, retirement has left the problem of controlling him to my successor!

Ian

Jon's response

> If you need to protect the system against inadvertent changing or removal of files, it's fairly easy. Protecting against determined hackers is more of a problem. Easy way:

Well, I actually think it can be pretty secure, assuming no gaping holes are found in Linux, itself. You can set the system up so root accepts no remote logins, period, and the only thing you can do remotely is ftp or telnet to one or two accounts, and then only from one of your own, local, machines. Set networking so that no connects are accepted except from that local machine, period. that should make it fairly hard for hackers to get in.

Now, if you only have outside network access on your other machines by dialup, and don't have a gateway on that machine, the Linux machine is pretty safe without much change. If you have a gateway, or keep one of your machines connected to the net all the time, you have to take some precautions. You can also set up your gateway to prohibit any remote access to the Linux machine. If you want to make EMC-related files available outside, you would need to move them to the 'public' machine. This makes sense anyway, as you wouldn't want outside demand for a file mentioned on your web page to bog down the CNC control machine while you were making parts.

Jon

A.5 How can I get EMC out of E-Stop?

This page is the result of a thread initiated by Max Heise on emc@nist.gov. It includes the relevant messages.

From: Max Heise mahemt01@fht-esslingen.de

Hi Everybody,

My Machine is now running fine for about 2 weeks. The problem was a incorrect connected amp fault. What finally did help me was Fred's "Why can't I come out of estop? and, parallel port debugger, Fri Mar 3 16:07:13 2000" .. just how many things affect estop. Aside from stupid things like wrong parallel port or motion board addresses, they are:

input from amp fault

input from positive hardware limit switch
input from negative hardware limit switch
positive software limit
negative software limit
input from estop sense

Only the pos/neg hard/soft limits turn the position digits red. Amp fault never shows up anywhere, so if you have this set wrong in your .ini file, you'll never know it.

And the following part from Will Shackleford's "Re: Several questions (Will Shackleford, Mon Mar 13 19:29:36 2000)"

Unfortunately there are a lot of reasons for not being able to come out of estop. Usually I start to look for the reason by running

plat/linux_2_2_13/bin/usrmot -ini mymachine.ini

And entering "show flags" to see most of the flags that might prevent you from coming out of estop.

Date: Wed, 12 Apr 2000 12:20:26 -0400 (EDT)
From: Max Heise mahemt01@fht-esslingen.de
To: Multiple recipients of list emc@nist.gov
Subject: Not getting out of estop

Hi,

I not getting out of estop. No matter what I try. My setup looks as follows.

Hardware:

AMD K6-2 at 300MHz
64 MB RAM
Only one (built in) par. port
STG Ver.1 Board - 4 Axis
2 Axis table plotter (x,y), with one rotary axis (z) for the 5 tools
(tool offset via G54-G58). Home and limit switches connected to STG board, estop connected to par. port. X and Y Axis connected to STG board too.

Software:

Redhat 6.1
linux 2.2.14
rtlinux-2.0
EMC-15-Mar

Some patches: ac patches and a patch for my ali 15xx IDE driver to do UDMA33. Everything compiles and runs fine. The simulation runs just great. Encoder feedback is working for all 3 axis. (what is the plural of

axis ?) I am using ioshow.tcl to monitor the par. port. If I tell tkemc to get out of estop I cannot see anything happen on digital out 10. If I switch digital out 10 using testppt tkemc goes into estop reset - but still not out of estop even if I switch digital in 1 using the estop button.

Or is the bridgeport PLC logic so different from my machine setup that I have to write my own one. Can I take tkio.tcl as an example ?

How can I see what's the reason for not getting out of estop ? Does anyone has a clue ?

Thanks for your help.

Max Heise

Date: Wed, 12 Apr 2000 15:44:37 -0400 (EDT)

From: Jon Elson jmelson@artsci.wustl.edu

To: Multiple recipients of list emc@nist.gov

Subject: Re: Not getting out of estop

The first thing to check is the address of the parallel port. If you do "more /proc/ioports" it will show the physical I/O address that is set for your parallel port. Make sure that corresponds with the one in the .ini file. OK, there are 2 things needed to get the machine 'running'. First, you need to get E-stop reset, which you have apparently done with testppt. If the external circuitry was set up to pull pin 13 to +5 Volts on the 25-pin parallel port connector, and to pull pin 16 to ground, then hitting the F1 key should do the same. Once in e-stop reset, you should be able to hit F2 to go to "machine on". But, that may require the limit switches to all show "not at limit" status. You can either override this (a screen button, I think) or change the polarity of the sense lines for the limit switches in the .ini file.

> Or is the bridgeport PLC logic so different from my machine setup that I have to write my own one. Can I take tkio.tcl as an example ?

I think you want to use the script that has minimill.io rather than Bridgeport.io. That should also use minimilltask rather than bridgeporttask. I think you HAVE gotten out of estop, but it is a 2-stage process. First you have to clear estop, then set machine on. I'm not sure of the reasons for this, and it has some drawbacks, but I have gotten it working, and am used to it. There are some diagnostic uses for the state between estop and machine on, like servo alignment.

Jon

Date: Fri, 3 Mar 2000 10:07:14 -0500 (EST)

From: Fred Proctor proctor@cme.nist.gov

To: Multiple recipients of list emc@nist.gov

Subject: Why can't I come out of estop? and, parallel port debugger

EMC users,

I had that frustrating "why can't I come out of estop" problem yesterday, and after I ran through all permutions of

ESTOP_SENSE_POLARITY and ESTOP_WRITE_POLARITY I discovered that I had
the wrong address for the parallel port (I clobbered my .ini file and had to redo it from scratch). In doing this I rediscovered just how many things affect estop. Aside from stupid things like wrong parallel port or motion board addresses, they are:
* input from amp fault
* input from positive hardware limit switch
* input from negative hardware limit switch
* positive software limit
* negative software limit
* input from estop sense

Only the pos/neg hard/soft limits turn the position digits red. Amp fault never shows up anywhere, so if you have this set wrong in your .ini file, you'll never know it.

The tcl interface to the EMC (emc/src/emctask/emcsh.cc) has a status command, "emc_joint_limit", that returns "ok", "minsoft", "minhard", "maxsoft", or "maxhard". Currently the digits are colored red only when it's not "ok". It would be better if there were some indication of what wasn't "ok". On the fossilized terminal-graphics GUI "keystick", the axis labels were bracketed by characters, like this:

-*X– –Y*- –Z**

where "*" meant "limit exceeded," and the characters indicated negative-hard, negative-soft, axis, positive-soft, positive-hard. In the above example, X is at a negative soft limit, Y is at a positive soft limit, and Z is at both positive soft and hard limits.

On the TkEMC it could be done similarly.

Anyone on the GUI committee want to try this? You simply need to use the "minsoft", etc. values explicitly to get the status.

Amp fault is not part of the emcsh.cc Tcl interface to the EMC. I will add this.

Regarding estop sense, the "ESTOP" label will still show "ESTOP" when you release the physical estop, since the controller has to acknowledge this. It would be nice if there were a label or light or something that showed the raw estop sense value directly. Then, when you reset the estop (pulled up on the estop button), the "ESTOP" label

would still show "ESTOP", but the raw label or light would show that the button was released. This would be nice for debugging.

Right now the raw estop isn't part of the status anywhere: not in NML, not in the Tcl interface. I will add this.

Regarding parallel port debugging, there is a text-based utility for this in the EMC distribution, called "testppt". It's compiled from emc/src/emcnml/parport.c, when MAIN is defined. Use it like this:

testppt -addr 0x378

Then type:

help/? print this help
s [#] set bit #
c [#] clear bit #
k Check outputs
ENTER print inputs
q quit

I also wrote a little Tk script that uses iosh to pop up green or red circles for the inputs. It's not in the EMC distribution yet, but it's appended here. (related version ioshow.tcl available from the dropbox)

–Fred

Date: Mon, 13 Mar 2000 13:29:37 -0500 (EST)
From: Will Shackleford shackle@cme.nist.gov
To: Multiple recipients of list emc@nist.gov
Subject: Re: Several questions

> Date: Mon, 13 Mar 2000 13:01:25 -0500 (EST)
> Originator: emc@nist.gov
> From: "Joel Jacobs" jj@netexp.net
>
>
> Hi all,
> I finally got EMC up and running on a Pentium 100mhz computer running
> 2.2.13/rtlinux-2.0. The computer seems grossly underpowered as I can

not
> get the motors to run smoothly. I had to set 'PERIOD' in freqmot to
100
> before I got the GUI working. I'm trying to decide what to do,
whether or
> not to try a faster computer. It seems like a really nice program if
I
> could get it working but I noticed some peculiar behavior that I'm
not sure
> is related to cpu speed.
>
> 1. Sometimes I start it and it won't come out of estop, I shut it
down and
> restart it and it may come out of estop but can't turn the machine
on.
> Restart again and everything turns on and I can move motors but
after a
> while it may stop responding to any input. When I first installed
it, it
> wouldn't work at all and the coordinates display was red - I changed
the
> polarity of the limit and home inputs and it started to work but the
input
> pins on the printer port are 'open' maybe I need to tie them low?
>

Unfortunately there are a lot of reasons for not being able to come
out of
estop. Ussually I start to look for the reason by running
plat/linux_2_2_13/bin/usrmot -ini mymachine.ini

And entering "show flags" to see most of the flags that might prevent
you from
coming out of estop.

> 2. Everytime I start the program the default jog speed is 300 imp no
matter
> what I put in the ini file for max velocity.
>
> 3. I can't edit the settings in the calibration dialog. I click on a
> setting and then try left and right arrows to see if the carrot
moves and it

> may move on another field.
>
> 4. I wanted to try emcmot but I get an un-resolved 'hrgettime'. Any clue?
> It doesn't seem to be in the system.map that was created when I compiled the
> rt kernel.

Did you run the install_rtlinux_base script? It should have insmoded rtl_time.o
which has gethrtime. (I assume hrgettime is a typo.)

>
> Thanks for any help.
>
> Joel
>
>
>
>
>

William Penn Shackleford III shackle@nist.gov
National Institute of Standards Technology Tel: (301) 975-4286
100 Bureau Drive Stop 8230 FAX: (301) 990-9688
Gaithersburg MD 20899 USA
http://www.isd.mel.nist.gov/personnel/shackleford/
Office Location: Bldg. 220 Rm A253

A.6 How can I set up auxiliary inputs and outputs?

Date: Fri, 22 Sep 2000 11:43:10 -0400 (EDT)
From: Fred Proctor frederick.proctor@nist.gov
To: Multiple recipients of list emc@nist.gov

Luc Vercruysse wrote:

> http://www.linuxplc.org `http://www.linuxplc.org` has a public domain plc emulation on a (real time)linuxbox of a siemens compatible plc.I wonder if it is posible to implement it in the EMC project a a replacement for the IO machine depended stuf (such as minimilio ..). Doing this

makes it possible to the end user to write the machine interface in a PLC-language.

Has somebody experience with the linuxplc ??

I don't have experience with the linuxplc, other than by clicking around the linuxplc.org web site and wondering what it actually is. You say it's a Siemens-compatible PLC. Does this means it's programmable using IEC-1131 languages? I'm very curious about this. It may be just what I'm looking for.

The PLC is possibly the weakest part of the EMC. Right now, there are two supplied "PLCs" that control the discrete I/O (estop, lube, coolant, spindle): bridgeportio and minimillio. minimillio just controls our NIST desktop minimill that doesn't have coolant or lube, so it's a stripped-down version of bridgeportio. bridgeportion is written in C++, not a very common PLC programming language. We did this since we have in-house tools for programming hierarchical control systems using C++, which we have used for many other projects, so it made sense for us. For EMC users, it's not a very popular choice.

Bridgeportio and minimillio are slightly customizable (without resorting to C++ programming) via the .ini file, by setting the location and polarity of input/output bits and some other parameters like delays for spindle brake engage/release. Adapting them for anything else, like an automatic tool changer, means coding in C++ or replacing them with something else.

Looking at the EMC I/O controller as a black box, there are three things it must do:

1. [TOP] Create NML buffers for commands to it from the EMC task controller and status from it to the EMC task controller. This is what connects it to the EMC.
2. [MIDDLE] Read NML commands that come in from the EMC task controller (e.g., coolant on, etc.), do what's requested, and write status back (done, executing, error, and values like coolant is on/off, etc.)
3. [BOTTOM] Talk to real-world I/O points.

As an alternative to the C++-based bridgeportio, we wrote a Tcl/Tk-based PLC. Here's what we did:

1. Extended the Tcl/Tk windowing shell "wish" with EMC-specific commands, creating the "IO shell" iosh. This was done similarly to how we built the EMC shell emcsh, used for building Tcl/Tk GUIs like tkemc.tcl. iosh creates NML buffers (requirement (1) above, the TOP), and adds Tcl words "inb" and "outb" for doing PC-bus byte I/O (requirement (3) above, the BOTTOM). iosh can be thought of as the PLC engine, that can be programmed in Tcl/Tk to implement any arbitrary PLC. It doesn't have any particular PLC logic program, for say the Bridgeport. You have to write this yourself, in Tcl/Tk, as you would write a ladder logic program for another PLC.
2. Wrote a Tcl/Tk script that does the actual PLC logic (requirement (2) above, the MIDDLE). This is tkio.tcl. It implements the Bridgeport I/O controller, just like bridgeportio. It's plug-compatible with bridgeportio: you can specify "tkio" instead of "bridgeportio" in the .ini file, e.g.,

[EMCIO]

EMCIO = tkio

I tried it out a while ago and it looked like it worked, but the "plug compatibility" might not be 100% due to bugs. I'll have to check it out on our machine.

Note that since we're extending wish to build iosh, we get some GUI stuff automatically. That is, in tkio.tcl, there's some script code for popping up a window with some red and green LEDs that show IO status. So, iosh lets you program both the PLC logic and the GUI appearance in the same script. So, unlike bridgeportio, which doesn't pop anything up, iosh pops up a blank canvas that your script can populate with whatever you program. If you choose to write just a straight PLC program, the canvas is just a little annoying blank square.

If you want to use a PC as a PLC without running the EMC, you can still use iosh and your own script. The NML buffers will be created and simply ignored, and you can write your PLC and accompanying GUI for any purpose whatsoever. Or, you can take iosh.cc, strip out all the NML stuff, and just leave the inb/outb commands there for your own stripped-down shell.

Or, you can use wish out of the box, write two short C programs that create "inb" and "outb" (we did this), put the in /bin, and call these within wish. This shows how a Linux box can be converted into a PLC in about 10 minutes. Of course, you program in Tcl/Tk, not ladder logic.

Now, I like Tcl/Tk and if I had to write a PLC with a gun to my head I'd use wish and the /bin/{inb,outb} programs real quick-like. My opinion on the end-all, be-all Linux PLC is something better:

1. It uses the Linux box for program development and actual PLC execution. So, you don't use the Linux PC for programming, then download the program to a commercial PLC through a serial port. It's the whole shebang. A PC can be overkill for simple PLC tasks, but in my case the PC is also running a CNC and I want to use the extra CPU cycles for a soft PLC.

2. It should be able to be programmed in any of the IEC-1131 languages (ladder logic, structured text, function blocks, instruction lists, and Grafcet), just like a commercial PLC. So, it should have a graphical programming environment, at least for ladder, function blocks, and Grafcet, where you can drag-and-drop icons for ladder symbols, logic blocks, etc. to build your PLC. Structured text and instruction list programmers could use plain old text editors.

3. It should support I/O board and I/O point configuration. You'd be able to associate logical inputs 0-4 with parallel port input bits 0x379/0-4, logical inputs 5-12 with 8 bits of input on your Keithley Metrabyte I/O board at 0x300/0-7, logical outputs 0-7 at 0x301/0-7, etc.

4. At least three options for program execution:

- a. Programs are executed directly via interpreters specifically written for each one.

- b. Programs are converted into some binary format, like Siemens proprietary format, and run on an emulator. This lets you write programs that could be downloaded to a commercial PLC, or run existing PLC programs from commercial PLCs on your PC in the emulator.
- c. Programs are converted from the IEC language into an intermediate form, compiled using native compilers for Linux, and then run as Linux executables.

I lean toward (c). (a) requires coding and maintaining five interpreters, and interpretation is slow. (b) requires building a PLC emulator, which means picking one, reverse engineering it, and suffering emulation overhead, although it does have the advantage of being able to run legacy PLC programs on a PC. Not a selling point for my needs.

With (c), I'd pick conversion to C for maximum portability. That is, I could write in any of the IEC-1131 languages, generate C code, compile for my Linux box or move the C code to another system and compile for other processors. With RT-Linux, the code could run in the restricted RT space and be deterministically real-time. Or, I could run the PLC as a user-level Linux process if strict determinism is not a requirement. I/O boards typically come with some C-language driver or interface code on DOS floppies, so this is easy to handle. It's also possible to link in arbitrary C libraries in order to deal with things like serial ports, custom boards, TCP/IP sockets, the EMC NML communication buffers, etc.

If the C code generator includes creation of NML and handling it in the read-execute-write PLC cycle, then we could write EMC I/O controllers in say ladder logic and plug them right into the EMC. The sequence would be read inputs, read NML commands, run logic, write outputs, write NML status, wait until next cycle...

The graphical console interface could be attached a variety of ways, many of which work with method (a). They could be written in X, Motif, Win32 (for C code compiled on a Windows box), and linked directly into the PLC code. I prefer separating the two as we did with the EMC, and having a separate process. In my case, I'd write something equivalent to emcsh, effectively iosh upside-down, for writing commands to the PLC, reading status from it, and displaying it nicely. This would let you build standalone PLCs. If you're plugging into the EMC, there need not be a separate GUI for the IO controller, since it could be part of the tkemc or a popup script.

The development cycle would begin with the programmer designing the PLC. Then, he'd bring up the programming tool, which might just be a text editor for structured text or instruction lists. Once the program is written, he'd run the code generation tool and get C code. Then, a make to build the executable. The GUI would be written separately, if it's needed at all.

As part of the development environment, we could write a GUI with buttons for each part of the cycle, e.g., [EDIT] [GENERATE] [COMPILE] [RUN]. If the skeleton were sophisticated enough, it could mark execution points so you could [PAUSE] [STEP] [EXAMINE] and [RESUME] execution. This is a separate thing from the PLC console GUI, which might just have LEDs showing I/O points, execution time health bars, etc. and would run for the tens of thousands of hours the PLC is alive.

First I want to find out what linuxplc.org has. It may be just what we want, except for the NML connectivity, which we could add. If we have to start from scratch, here's what I think needs to be done:

1. Decide on the structure of the C code skeleton that implements the read-execute-write-wait cycle, and handles NML.

2. Design the programming tool, the thing that lets you lay out ladder rungs or logic blocks. I would start with structured text or instruction lists, so a text editor would suffice for this. Later would come the integrated development environment with the [RUN] button.

3. Write the converters. We need one for each language, converting programs written in each to the flesh on the C skeleton. I'd start first with structured text or instruction lists. Will Shackleford has written code generation tools in Java. I don't know what to use here. I'd do it in C.

4. Lastly is just compilation using the Gnu compilers, which we get for free.

(1) is moderately hard, (2) is done if we start with the text languages, hard for the integrated development environment, (3) is hard, (4) is easy makefile stuff.

–Fred

A.7 Does EMC support probe digitizing?

Yes the EMC does support probing. There is a working probe routine under the tkemc scripts menu. If you are using steppers, you can access the probe pins on the parallel port. They are settable in the script.

A.8 How to alter the direction of axis ?

You can just swap the leads on your axis (stepper or servo) motor. You do it in the software by putting a negative (-) in front of your inputscale value. You may have to do the same to the outputscale if you are using some of the motion modules.

A.9 What happens when you press the home button ?

Numbers turn from yellow to green when each axis is homed. Assuming you can jog, when you select an axis and hit the home button, that axis should start seeking slowly towards the home switch (speed and direction are configurable), and should stop (or reverse, or continue a little further depending on the configuration in the .ini file) when the switch is activated. At that point the numbers turn green. If you set the home switch polarity such that it matches the state that the input bit naturally floats toward, then the axis should home immediately when you push the home button.

A.10 I'm wondering if there is an official pinout for EMC in six axes mode (hexapods)?

If there is no official assignment yet, could we agree on one?
Existing DB25 EMC pins:
2 - X dir
3 - X clk
4 - Y dir
5 - Y clk
6 - Z dir
7 - Z clk
Proposed 3 more axes:
8 - A dir
9 - A clk
16 - B dir
17 - B clk
1 - C dir
14 - C clk
(or should we be calling the axes: X, Y, Z, U, V, W ?) Another assignment goes like this:
Pin 1: Motor 4 DIR
Pin 2: Motor 0 DIR
Pin 3: Motor 0 STEP
Pin 4: Motor 1 DIR
Pin 5: Motor 1 STEP
Pin 6: Motor 2 DIR
Pin 7: Motor 2 STEP
Pin 8: Motor 3 DIR
Pin 9: Motor 3 STEP
Pin 10: (unused)
Pin 11: (unused)
Pin 12: ALL Motors REFERENCE SWITCH
Pin 13: ALL Motors NEG. LIMIT SWITCH
Pin 14: Motor 4 STEP
Pin 15: ALL Motors POS. LIMIT SWITCH
Pin 16: Motor 5 DIR
Pin 17: Motor 5 STEP
Pin 18-Pin 25: GND

A.11 Is there an easy way to change the accuracy used in EMC?

It seems that the g-code writer (Enrout) doesn't want to write distance and arc values to four decimal places. Is there an easy way to downdrade

the accuracy required by the EMC interpreter to .001 from .0001. This is mostly a problem when trying to match radii to begin and end points. The easiest way is if you can use the R word (radius) instead of I and J. This gets rid of most of the problem. If not, there are parameters that are compiled into the code.

/* numerical constants */

#define TOLERANCE_INCH 0.0002

#define TOLERANCE_MM 0.002

#define TOLERANCE_CONCAVE_CORNER 0.01

in emc/src/rs274ngc_new/rs274ngc.hh

Change these to what you need and recompile. This did solve most of the problem of using the EMC with Enrout files although Enrout still ocasionally throws in a line with g1 and nothing else.

A.12 How does autotune work ?

Max wrote:

On the EMC Handbook on http://www.linuxcnc.org `http://www.linuxcnc.org` under PID Tuning I found some lines of text from "Fred's PID Report":

"The student, Kees ("Case") Stolk, from the University of Twente in the Netherlands, wrote a Tcl/Tk script that automates much of the process, including going into machine-off, opening the log, running the DAC out command, saving the log, storing multiple runs, and popping up PID gains. It's pretty slick. I'll put this up on the FTP site once I verify that it works with the new release."

That sounds interesting, where can I find it? I found the pidtune zip files on the nist ftp server but not this Tcl/Tk script.

Some of Kees' Tk scripts have disappeared, I think since they were on a local disk instead of our file server. I've tried to track them down but can't find them. What we have is the analysis, and a TkEMC button to generate the logs used as input to the analysis. I'll try to resurrect Kees' Tk code. Here's a description of what we did.

The anaylysis presumes amplifiers in current mode. If you're in velocity mode, you'll have a tach fed into the amplifier, and tuning means setting the velocity feedforward gain to match the volts per units per second you get (e.g., you may measure 1 volt = 1.7 inches/sec, so FF1 = 1/1.7). Then you crank the P gain up to clean up the residual error.

In current mode things are a little different. For a given DAC voltage, the velocity v. time plot starts at zero and follows a curve that looks like capacitor charging, eventually attaining a steady-state velocity. The steady-state velocity is a function of the DAC voltage, and the time constant is about the same for all DAC voltages. If you plot the steady state velocity v. DAC voltage for several tests (1V, 2V, 3V, ..., 10V), you'll get a linear plot with an X offset. This offset is the threshold voltage, below which the axis doesn't move. The slope of the plot is the volts per units per second, as above.

So, you'll now have the slope "k" and the time constant "t" for your system. You then pick a design parameter, lambda, which basically means the crispness of your control. You can't choose this arbitrarily high since the resulting output voltages exceed your DAC limits. Given k, t, and lambda, then

P = 2 lambda + t / k lambda^2

I = 1 / k lambda^s

D = 2 t / k lambda

The "pidtune.zip" file is a ZIP archive of 6 MS-Word documents describing the analysis. I made little posters of these that we keep down by the Bridgeport for demos. There are some literature references provided.

To generate the logs in TkEMC, go into Settings-Logging..., select a log type of "Pos and Voltage", and start the log. The actual data logging won't begin until a DAC output command is received. Do this in the Settings-Testing... dialog. When you reach steady state, stop the log and zero the DAC.

Kees' Tk script basically did this for you.

–Fred

A.13 What is BDI ?

It is a CD on which Paul Corner has placed all of the Linux, Real Time Linux, and the EMC (Enhanced Machine Controller). It is called the Brain Dead Install, hence BDI. With this CD and a reasonable standard pentium, cyrix, or K6 computer, you can control machines. All kinds of machines from common mills and lathes to robot arms and hexapods. You can find BDI distribution places from from http://www.linuxcnc.org/bdi/ `http://www.linuxcnc.org/bdi/`

A.14 What is the minimum specification of the computer required?

An old P133 with 16-32Mb RAM, a 500Mb IDE hard drive, and a CDROM drive that can be booted from. If you want good performance from stepper motors, a Pentium 233 or better is recommended.

A.15 Can I use a laptop computer

This is allways a difficult question to answer. Most of the laptop computers use non-standard hardware for the LCD screen, some of which are supported under Linux. The CDROM drive can also be a major source of trouble especially if it can be swapped out of the same bay as the floppy drive.

In short, unless you are experienced in installing linux on a laptop, buy a cheap desktop computer. This is not to say it is impossible, rather it is outside the scope of this FAQ.

A.16 Do I need to install Linux first ?

No - The BDI automatically installs and configures linux along with EMC.

A.17 Will it run under Windows 98 ?

No. The install will remove the Microsoft virus for you. :)

A.18 Can I run a hexapod with it ?

Yes you can.

A.19 I get a screen full of error messages, but no graphical install screen when I try to install the BDI ?

The install CD is not recognizing some of your hardware. Try to change your display card to some other listed in http://www.redhat.com/support/hardware/ `http://www.redhat.com/support/hardware/` "supported-hardware"-list. You could also ask if there is version of BDI-CD that can handle your hardware. You can also look for answer from BDI's pages at http://www.linuxcnc.org/bdi/ `http://www.linuxcnc.org/bdi/`

A.20 I can't access the floppy drive with the Floppy icon.

The floppy icon on the KDE desktop need to point to the correct device/directory. Right click on the icon - Properties - Device.Set both 'Device' and 'Mount Point' to /mnt/floppy.

A.21 The computer crashes when I try to make a move in EMC

Try to edit your .ini file. Problem propably lies in there. Try to run sim.run script and if that's ok, use that .ini file as base for your own ini-file. Later on we'll put couple of working ini-files here. For now, you can find them from http://www.linuxcnc.org/dropbox `http://www.linuxcnc.org/dropbox`

A.22 Can I print from the BDI linux ?

Yes - But to configure the print driver, you will need two files found on the Programs page. Install control-panel followed by printtool then open up a console window and run printtool (you'll need to be root to do this).

A.23 How do I burn a CD from the downloaded image ?

You have downloaded the ISO image of the BDI from the web now want to burn a CD on a Windows box....

The following note was received from Mark Nudelman -

1) Download BDI .img file from www.yty.net/cnc `http://www.yty.net/cnc`

2) Change the .img extension to a .iso extension

3) From a CD burning program (I used CD Creator 4.0) create a CD using the "Create CD from CD Image" option (do not try and create a regular data CD) and make sure that the file type is "ISO Image Files:.

4) Insert this CD and a Linux boot floppy in the computer that is to be loaded. (I already had a boot floppy from my previous install, but I think that windows users can use a program called rawrite.exe to help them create this boot disk. There was information available on this process on the Linux site)

5) Reboot the computer and follow the BDI instructions.

A couple of extra notes from correspondence with other people that have had trouble with the downloaded image.

It would appear that one or more files can be corrupted during downloading or burning. Henkka has included a checksum so that the disk can be verifyed. Please follow the instructions before using the CD as a beer mat.

Step 4 should not normally be necessary unless the computer will not boot directly from the CDROM drive.

A.24 What are all these error messages when I try to compile?

Date: Fri, 15 Sep 2000 10:13:06 -0400 (EDT)

From: Fred Proctor frederick.proctor@nist.gov

To: Multiple recipients of list emc@nist.gov

Subject: Re: Steppermod.o problem

David Johnson wrote:

> I have just downloaded the 11 Aug Release and have installed it. My problem

> is that when I run emc.run it returns with a message that steppermod.o

> cannot be found.

> I've checked and it hasn't been compiled.

>

> Question is how or where do I set the

> -DSTEPPERMOTORS flag to make it compile correctly.

It should have been compiled, and since it wasn't there may be some Linux configuration problem.

I recompile using the makefiles, like this:

cd /usr/local/nist/emc/src
make PLAT=rtlinux_2_2 all

This should recompile everything needed for the rtlinux_2_2 platform. I
don't know which one you're using; it could be rtlinux_09J, or rtlinux_2_0.

You can trap the output of this by running "script", which captures terminal IO. Then, you can do a postmortem and figure out what error aborted the compile. Stick the script at the end of an email and we can
look it over. To script the above commands, do this:

you> script
Script started, file is typescript
you> make PLAT-rtlinux_2_2 all
..bunch of output...
you> exit
exit
Script done, file is typescript

The file "typescript" is created.

–Fred

A.25 Why does my PC slow down after running EMC?

This faq does not deal with EMC gui's being slow while EMC is running. If you are using freqmod.o that problem is most likely a PERIOD-parameter problem in your ini file. If EMC has been running okay and now it seems to be slow while running you may have a module problem.

EMC loads modules into the kernel whenever it runs. There will be a different set of modules running depending on which rtlinux kernel you

are using. Sometimes these modules do not get removed or idled properly when EMC shuts down. The following post from Will may help you understand and solve the problem.

The rtlinux-2.2 version of EMC uses these realtime modules:

mbuff
rtl_fifo
rtl_posixio
rtl_sched
rtl_time

The rtlinux-09J uses this realtime module.

rtl_sched

If you encounter a problem with modules please post your comments and results to emc@nist.gov

From: shackle shackle@cme.nist.gov
To: Multiple recipients of list emc@nist.gov
Subject: Re: mbuff in 2.2.14 rtlinux-2.2

On Mon, 11 Sep 2000 11:13:19 -0400 (EDT), Ray said:

>
>
> Folk
>
> This should only be a problem with rtlinux-2.0 or later.
>
> I've been running the EMC as a user rather than root for some time and
> have found an interesting problem that crops up now and again.
>
> After exiting the EMC, mbuff seems to get captured by another
> process. Then when I start EMC again it seems to share mbuff and after
> a few starts and stops it doesn't run properly at all. The sysmptoms are
> a very slow start of emc and there are big delays in the mouse functions
> when using tkemc.
>

> In order to run as a user, one thing I had to do is:
>
> chmod a=rw /dev/mbuff
>
> It would appear from xosview that when this problem occurrs, mbuff
is
> causing my system to use 100% of the processor's power. (perhaps
this is
> related to the problem noted in a recent post by Jan)
>
> After this error has occurred I get this report from lsmod:
>
> Module Size Used by
> mbuff 5356 3
>
> And when I run the EMC I get delayed responses from tkemc and the
> following report from lsmod.
>
> Module Size Used by
> steppermod 152652 0 (unused)
> rtl_fifo 7336 0 (unused)
> rtl_posixio 6684 0 [rtl_fifo]
> rtl_sched 36756 0 [steppermod]
> rtl_time 14080 0 [steppermod rtl_posixio
rtl_sched]
> mbuff 5356 7 [steppermod]
>
> The only way I seem to be able to fix the problem is to reboot.
Any hints
>
> Ray
>
>
>
>
>

The next time this happens try some/all of the following.

It's a good idea to save any files your editing, exit the editor and
run sync a
couple of times first since there is a fair chance we are going to
hang the

system completely.

Run top to see which processs are taking the most cpu time.

If the highest process on the list is not idle, top, or init.

Try killing it with killall

killall processname

If it doesn't disappear off the top list, try the same as root and/or try:

killall -9 processname
(Warning killing some system processes will hang your system)

Run ipcs to see what shared memory segments and semaphores were left around

Use "ipcrm" to delete them.

ipcrm -shm id

ipdrm -sem id

Try removing the modules one at a time.

rmmod steppermod
rmmod mbuff
. . .
—Will

–
———————————————-

William Penn Shackleford III shackle@nist.gov
National Institute of Standards Technology Tel: (301) 975-4286
100 Bureau Drive Stop 8230 FAX: (301) 990-9688
Gaithersburg MD 20899 USA
http://www.isd.mel.nist.gov/personnel/shackleford/
Office Location: Bldg. 220 Rm A253

A.26 What can I do to remove mbuff from the kernel after running EMC.

Sometimes when EMC shuts down, it does not clear out all of the sectors in mbuff and running the command (rmmod mbuff) gets a "device or resource busy" response. You can see what modules are loaded using the command

lsmod

or

/sbin/lsmod.

The problem is that when the EMC starts up, it attempts to install mbuff and then uses mbuff to allocate a block of SHMEM under the name emcmotStruct. Sometimes it creates several of these and does not remove them all.

Solution: Find the mbuff directory in the realtime stuff. For the box that I'm writing this on it is /usr/src/rtlinux-2.2/drivers/mbuff/. Change into that directory and issue the command

make test

This will compile some executable files that exercise mbuff. It also runs one of them and attempts to use mbuff. You will probably get a core dump but it will not stop your PC. Just delete the file named core. You will see the test program adding and deleting spaces using mbuff. After a bit the test program will just quit. Pressing enter will get you your terminal prompt again. Now enter the command

./mbuff_dealloc emcmotStruct

Below is a sample of what the command and response might look like:

[root@localhost mbuff]# ./mbuff_dealloc emcmotStruct

emcmotStruct shared memory area was 0 bytes long

The zero answer means that it found a block and removed it. If you get a "-1" answer it means that dealloc did not find the named block.

Most of the time, there will only be one emcmotStruct left laying around and you can issue the command

rmmod mbuff

or

/sbin/rmmod mbuff

and you should get a terminal prompt with no messsage. If you get the "0 bytes long" response and still rmmod says device or resource busy repeat the dealloc command until you get a "-1" response and try again. If you are not able to remove mbuff, you will need to reboot your computer.

A.27 emctaskmain.cc 2322: can't initialize interpreter

Check your .var file (The name of the .var file is in the .ini file) and see if

5220 0.0000

Should probably be

5220 1.0000

What is 5220? 5220 is the coordinate system number used by EMC. If it's "0", EMC doesn't know what coordinate system to use.

A.28 emctask.cc 245: rs274gnc_error: Radius to end of arc differs from radius to start

I had the same problem a while ago. Look at src/rs274ngc_new/rs274ngc.hh at the beginning an decrease these values.

/* numerical constants */

#define TOLERANCE_INCH 0.0002

#define TOLERANCE_MM 0.002

Once you have changed them => recompile.

Max

A.29 While booting the computer following error comes up: /dev/hda2 :unexpected inconsistency; run fsck manually (i.e, without -a or-p options.

You should give your root password and issue following command:

fsck /dev/hda2

Notice space between the command and the parameter. "fsck" is short for File System ChecK, it's quite same as scandisk in Microsoft-world. It's generally good idea to check all filesystems after crash. You can do that by checking what filesystem you have and them checking them all. First check what you have by issuing

cat /etc/fstab

[henkka@kilpikonna henkka]$ cat /etc/fstab

/dev/hda1 / ext2 defaults 1 1

/dev/hdg5 /varmistus ext2 defaults 1 2

/dev/hdg6 /home ext2 defaults 1 2

/dev/hda5 /usr ext2 defaults 1 2

We are looking for entries where 3rd item is "ext2" (filesystem type) and we should check them all by issuing fsck to them one by one. In my example, commands would be like this:

fsck /dev/hda1 [enter]

fsck /dev/hdg5 [enter]

fsck /dev/hdg6 [enter]

Replace those with your own partitions. You will be asked if you want to repair/fix the filesystem. You can almost allways answer "Y". The "fsck -a /dev/hda2" command would answer "Y" automatically, but it's not recommended in this situation as there could be some filesystem damage that is better to leave unrepaired. After you have run all the "fsck" commands, issue "reboot [enter]" command for rebooting your machine.

A.30 What to do when the system crashes and there's no choise but "cold boot" it ?

Here is my long list of things to try when the system crashes, although after you read it you might decide reinstalling the BDI is the best alternative after all.

After it crashes but before you hit the reboot/power switch you can try these:

CTRL + ALT and one of the keys F1 through F7

All pressed simultaneously are used to switch between 7 virtual screens. Usually X windows runs on 7 and text only consoles run on the rest, so for example CTRL+ALT+F1 should switch to a text console.

- If you can't get a text console skip to "A.30.1"
- If you can get to a text console it may be that X windows is hung.
- If X starts at bootup and you normally use a graphical login You can kill X and all of the programs running on it by switching run levels:

Log into the text console as root and run

init 3

to switch to run-level 3 and therefore kill X

init 5

to switch back to run-level 5 and therefore restart X. If you normally use

startx

from a text console to start X, you can try using

CTRL+ALT+BACKSPACE

to kill X or the command

killall xinit

or

killall -KILL xinit

A.30.1 No text console available

If you can't get a text console you can probably still do a softer reboot using the "magic sys request keys". These have to be enabled beforehand both in the kernel configuration and in /etc/sysctl.conf. The entry in sysctl.conf to add is

kernel.sysrq=1

You also need to run "sysctl -p" or reboot after modifying sysclt.conf. To actually use "magic sys request", you should press following keys:

ALT + SysRQ + S

ALT + SysRQ + U

ALT + SysRQ + B

Keys on the same line are pressed at the same time. The SysRQ key is ussualy the same as the "Print Screen" key.

If you have to hit power button without doing a controlled shut down, it is likely your harddrive will be left in an inconsistant state. If you boot

linux with an out of sync harddrive it should automatically run fsck. fsck is similar to scandisk it can take a long time and prints lots of cryptic messages, but the great majority of the time it cleans everything up and will eventually boot linux and everything will be ok. If not, see previous A.29.

On the hopefully very rare occasions that all of the above has failed. You can try using the install disk as a rescue disk. I belive the command is "rescue" typed at the first redhat install screen. This ussually dumps you in a text console. You could try running fsck with a variety of different options to fix problems fsck doesn't fix in automatic mode. You can also use fdisk to check the partition table. Another thing that has happened to me a couple of times was to modify the kernel and reboot without having run "lilo" or properly edited lilo.conf. You can run lilo from the rescue environment, just be sure to find the copy of lilo.conf that was on the harddrive and use

lilo -C /mountpoint/etc/lilo.conf

You can use the "mount" command to help figure out what mountpoint is. Look for something like /dev/hd?? or /dev/sd?? on /mountpoint

– Will

A.31 Is there any "commandline" version of EMC for debugging or error finding ?

Some long time ago I had a problem setting up EMC. I then read about usrmot in the NIST docs.

usrmot A terminal-based diagnostics program that connects to the EMC motion controller, not the top-level EMC, and lets you type in motion commands and view status. It only requires that the motion controller be running.

You can even move your machine with usrmot. help or ? will give you a summary of what is available. For running usrmot you do not have to run all modules or the gui, but see below. <shameless quote from http://www.isd.mel.nist.gov/projects/emc/emcsoft.html#USRMOT>

Using the USRMOT Motion Utility

USRMOT is an interactive text-based utility that is used to set and test motion parameters for the EMCMOT motion controller. To use

USRMOT, first run EMCMOT standalone (yourprompt> represents whatever your system prompt is):

yourprompt> emcmot

In another window, run the USRMOT utility:

yourprompt> usrmot

motion>

The motion> prompt is displayed by USRMOT when it runs. Entering a blank line lets you see the status:

motion>

mode: free

cmd echo: 0
split: 0
heartbeat: 605
compute time: 0.014992
traj time: 0.200000
servo time: 0.020000
interp rate: 10
axes enabled: 0 0 0
cmd pos: 0.000000 0.000000 0.000000
act pos: 0.000000 0.000000 0.000000
velocity: 10.000000
accel: 100.000000
id: 0
depth: 0
active depth: 0
inpos: 1
vscales: Q: 1.00 X: 1.00 Y: 1.00 Z: 1.00
logging: closed and stopped, size 0, skipping 0, type 0
homing: —
enabled: DISABLED

</quote> and another quote, again by Will, from this list from 13 March 2000 <quote>

Unfortunately there are a lot of reasons for not being able to come out of estop. Ussually I start to look for the reason by running plat/linux_2_2_13/bin/usrmot -ini mymachine.ini And entering "show flags" to see most of the flags that might prevent you from coming out of estop. </quote>

and yet another quote from Fred from 17 March 2000 <quote> You can run the "usrmot" motion controller utility when the whole controller is up, or just if the motion controller is loaded. If you just load the motion controller, you can send its .ini file parameters down to it using the "inimot" utility, like this:

you> cd /usr/local/nist/emc
you> insmod plat/rtlinux_09J/lib/stg2mod.o
you> plat/linux_2_0_36/bin/inimot -ini yourfile.ini
(some messages saying initing axis #...done, or something like that)
you> plat/linux_2_0_36/bin/usrmot -ini yourfile.ini
motion> help
(shows terse help list)
motion> [ENTER] shows status
motion> enable
motion> show pids
motion> [ENTER] shows just the PIDs
motion> show times
motion> [ENTER] shows the min/max/average compute times
motion> show

motion> [ENTER] shows full status
–Fred
</quote>
Max

A.32 How does EMC's copyright work ?

This page is compiled from responses to the first post requesting information about the EMC copyright.

Cheng-Chang Wu chengchangwu@yahoo.com
To: Multiple recipients of list emc@nist.gov
Subject: Copyright
X-To: emc@nist.gov

Hello,

I see from the home page linuxcnc.org that EMC is a
open source project. But I can't find the copyright of
it.
It seems to me that EMC is a product of the
government, so it is only for american firms? If it is
really so, it can not
a open source project according to the definition.
"Open Source" is already a trademark in USA with a
good definition.

I'm writing this e-mail because I like the idea of
EMC, but I want to respect the copyright law of USA.
I'm
considering to start a CNC project if EMC is not a
really open source project.

Best regards

Cheng-Chang Wu

Ray Henry rehenry@up.net

Cheng-Chang Wu

Interesting questions!

I am an EMC user here and don't pretend to represent NIST
or the US government. My comments are mixed into your post.

At 10:38 AM 5/24/2000 -0400, Cheng-Chang Wu wrote:
>
>Hello,
>
> I see from the home page linuxcnc.org that EMC is a
>open source project. But I can't find the copyright of
>it.

The EMC files themselves are all public domain and may be
used in any manner you choose.

> It seems to me that EMC is a product of the
>government, so it is only for american firms? If it is
>really so, it can not

No. The public NIST work on EMC is available to anyone
for any purpose whatever. There are users scattered all over
and there are almost as many ways of using parts of EMC as there
are users.

> a open source project according to the definition.
>"Open Source" is already a trademark in USA with a
>good definition.

You are correct that there are a number of valid definitions of
open source projects in the USA. NIST itself has not applied
any one of the "copyleft" definitions to EMC.

I reserved copyright on tkbackplot in it's original form but opened
the tkemc popup version as public domain. The scripts that I've
written have no explicit copyright.

Dan Falck, Matt Shaver, and I have talked a bit about this problem.
I think that Dan will soon place the contents of linuxcnc.org under
some copyleft definition so that it will be easier for others to
contribute to it without feeling that some big company can borrow
it for their own inordinate profit.

I personally would like to see you contribute to EMC rather that spend a lot of time writing your own machine control. But you should feel free to use any part of the current EMC in any way that you want.

Do you feel that you need gpl or some other copyleft protection for the work that you want to do? Public domain is no protection at all!

I'd like to hear some serious discussion of this issue? Is public domain holding us back from serious collaborative work on EMC?

Ray

Cheng-Chang Wu chengchangwu@yahoo.com
To: Multiple recipients of list emc@nist.gov
Subject: Copyright
X-To: EMC emc@nist.gov

Hi, Ray,

I'd like to contribute to EMC, that's why I wrote the e-mail:-)

>Do you feel that you need gpl or some other copyleft
>protection for
>the work that you want to do? Public domain is no
>protection at all!

Public domain is OK with me. I like GPL very much, but I don't think it is the best copyright for a project like EMC. Six years ago I worked for a small machine tool company for four years. They needed a NC controller for their special machine. The controller had to work with their technologies. I couldn't have opened ALL the source code, because that would have revealed their secrets. There are many such small companies. They have technologies and special machines, but they have very few programmers. An open-source solution is great for them, but they need to keep their technolgies in secret in order to be able to compete with large companies. A large company don't need open source.

I have just read FSF's comments about various licenses
at http://www.fsf.org/philosophy/license-list.html `http://www.fsf.org/philosophy/license-list.html`.
But I can still not decide which license is good for
EMC. Perhaps pulbic domain is already good enough for
me.

Best Regards

Cheng-Chang Wu

Will Shackleford shackle@cme.nist.gov
To: Multiple recipients of list emc@nist.gov
Subject: Re: Copyright
X-Mailer: dtmail 1.2.1 CDE Version 1.2.1 SunOS 5.6 sun4u sparc
X-To: emc@nist.gov

EMC is "public domain". Since it was produced primarily by the US government
by law it can't be copyrighted.

It can be used by non-US firms.

In fact unlike most GPL copyrighted code, it can be used as part of a commercial
application and there is no requirement that such an application release its
source code.

BTW. The term "Open Source" is not copyrighted anymore. See

In short, You can safely use EMC without fear of violating a
copyright.

However, EMC is not OSI certified, and probably won't be anytime soon.

– Will

Chris Stratton stratton@mdc.net
To: Multiple recipients of list emc@nist.gov
Subject: Re: Copyright

X-Mailer: ELM [version 2.4ME+ PL54 (25)]
X-To: emc@nist.gov

Because the core EMC code is in the public domain, you can release your extensions to it under any license you choose - from public domain, to open source, to fully proprietary.

If you use some of the extensions others have written which are not public domain, you will need to use a license compatibly with the terms already applied to those extensions.

Otherwise, it would be nice if you release as much of your work as you can under as open a license as your are comfortable with, but you are under no obligation to do so.

Perhaps the best thing to do would be to separate any work you do into different modules. Everything that is of general interest gets released under your favorite open source license. Anything specific to proprietary technology in a particular machine wouldn't be as useful to others anyway, so don't release that part at all.

Just some thoughts from someone who's played with EMC but is not currently running it.

Chris

Jon Elson's reply to James Owens

Date: Thu, 13 Jul 2000 16:03:18 -0400 (EDT)
Subject: Re: EMC - a practical implementation?

james owens wrote:
> Hi All,
>
> I have got to add my tuppence worth. Don't get carried away with
> indignity that someone who is looking to hi-jack the whole project
for
> commercial gain. These people wish to put it in a pretty package on
> one CD where the end user inserts and presses go. They are not in
the
> business of doing the hard work by writing the software then proving
> it, they are marketing salesmen and as such when and if the software

> gets to the stage of insert and go the likes of you and me will be
cut
> out by copy-write law suits as they claim it all as their own idea.
> EMC will no longer be FREE.

This cannot happen. The software may NOT be copyrighted by
a 3rd party, and anyone attempting to do so may become a violator
of federal law. The can sell it as part of a package. But, they
probably can't sell it (the software) standalone for more than a
reasonable cost of media and production. And, they absolutely,
certainly cannot prevent anyone else from using it, distributing
it or selling it. This would be almost like somebody coming
in the night, stealing your car and then leasing it back to you.
Property they CLEARLY DO NOT OWN cannot become theirs,
just because they say so. I'm sure NIST knows how to deal with
this sort of thing, it must have come up before.

Jon

Fred Proctor

Date: Fri, 14 Jul 2000 15:58:47 -0400 (EDT)
From: Fred Proctor frederick.proctor@nist.gov
Subject: EMC software copyright, etc.

EMC folks,

The EMC software was written by us U.S. government employees, and is
not
subject to copyright. It's public domain. If we tried to keep it out
of
your hands we couldn't, because of the Freedom of Information Act.

You can do anything you want with it, like give it away, sell it,
whatever. You can't copyright it yourself.

You can put an EMC on a machine tool and resell it, as some of you
have
done. This is great. You can make an easy-install CD and sell this for
big bucks, no problem.

We wrote this to support our work testing plug-and-play standards for

machine tools and robots. There's no warrantee, support, or anything like that. The emc@nist.gov mailing list is the support, and it's working great.

–Fred

Jon Elson's reply to Brian Bartholemew

Date: Fri, 14 Jul 2000 15:34:07 -0400 (EDT)
From: Jon Elson jmelson@artsci.wustl.edu
Subject: Re: EMC - a practical implementation?

Brian Bartholomew wrote:

> I'm confused. I thought EMC, as required for tax-funded government-
> developed software, was public domain. The design goal of such terms
> is that companies commercialize it, because this is thought to
> distribute the created value back to taxpayers the best.

Yes, clearly a company can take advantage of the code, use it in a product, and sell that product. Indeed, that's the idea. But, that company CAN NOT claim they now OWN the code, and no one else may use it! That is entirely OPPOSITE of the idea!

And, that is what someone was bemoaning as the fate of EMC, that some company would "take it over", start selling it on a CD, and everyone would have to buy the right to use it from that company. That WILL not happen!

Jon

A.33 How do I use Windows software with Linux (in case I need it for some reason) ?

There are couple of options for using Windows software at linux. First, and in my opinion most working solution, is to use VMWare `http://www.vmware.com` software. It allows user to install w95/w98/nt4/w2k/linux etc. "client" machines on top of "host" system, in this case linux. I'm using this every day in my work. It seems rock solid and with 196Mb memory I'm running NT 4 server on top of linux happily. Only thing about this solution is the price. But there is an free alternative, Plex86 `http://www.plex86.org`. It's evolving and it's getting better and better

all the time. It operates on similar priciples as VMWare. Third solution is to use Wine http://www.winehq.com. It operates with slightly different manner, you can just run one application at the time. But even this is sometimes enough. Fourth solution is to use VNCViewer/VNCServer http://www.uk.research.att.com/vnc/ combination. It allows user to use -remote- Windows-machine from linux machine and vice versa. It's also free. It operates quite nicely over network, and it saves the "state" of desktop between sessions. So you can continue your work where it was left, even if you are continuining the work in another computer.

A.34 I'm new to Linux, could you explain it a bit to Windows-user who has been using "point and click" -method ?

Point and click is a common affliction. Command lines do not take much more effort at the level we need you to use them.

Click on the little house icon on the bottom bar or panel. It will bring up an instance of kfm, the file manager. I assume that you are running as root and you will be in the /root directory. You can always see where you are in the file window. Press the up arrow near the top bar of kfm and it will move you to /. Now click on the usr directory icon then the emc directory icon. Now you will display the correct directory for running the emc. It should show you a bunch of files like TkEemc and such. Hey, that's not so different.

Clicking on a text based file icon should show its contents in a window. Right clicking on a terminal or executable file and selecting open with should let you look at those as well. Sometimes clicking on an unknown file type will get you an open with window much like you see in MS windows. At that point you can always type kwrite and it will show you the file, even if it is a binary.

A file named *.run, where the star has the usual definition of any or all, is a start the emc file. In one of these files about line 27 should be a variable which points that run file at an .ini file. It will say something like

INIFILE=ppmc.ini

except the file pointed to is the name of your ini file. Perhaps it will be emc.ini. This information is critical because that is the specific .ini file that you will need to edit to work with that .run file. I've made this mistake before, changing one ini while running another. I kept wondering why my changes weren't doing anything.

Now let's take one more little step away from the icons to running from a terminal because that way you can view the startup feedback and some of the run stuff. The Kfm window that you got to this directory with needs to have the focus. Kfm is the file browser and focus means that the top line is a different color than the background. (Click on the top line to change focus to that window) Now press and hold [control] and press t on the keyboard. This opens a new terminal so that you can enter text commands. This keystroke, [control]t , opens that terminal in the directory shown in the kfm window. This means that you should be ready to go to work entering text.

In the terminal window you will see something like

[ray@localhost emc]$

The $ means that it is ready to accept a command. the emc] just before it is the name of the directory the terminal is working in. Type in

./emc.run

(dot slash) and it should immediately say something like file not found. This is a good thing if it goes on with other stuff. It is this other stuff that will allow you to begin to debug your setup.

With the mouse, you can click and drag across the lines of the terminal window to highlight what you want. But you can't press [control]c to capture them. Just leave them highlighted and start kwrite, advanced editor, or text editor from the K-menu in the bottom left corner and select copy from the menu or press [control]v in the editor and you should have a copy of the highlighted text from the terminal. Copy that into your favorite email program and we can comment on what we see there.

Another option is to use "script"-program, if you manage to get into a shell. You can issue the command

script

and all input and output in the shell's window will be saved in a file named

typescript

After you are done running EMC you can enter the command

exit

and a history of the complete session should be in typescript-file. Emailing the complete file may be better than trying to paste pieces of it together.

A.35 What are the files that are needed with EMC ?

The REQUIRENAME header from emc.i586.rpm:

rtlinux

rcslib

ld-linux.so.2

libICE.so.6

libSM.so.6

libX11.so.6

libXaw.so.6

libXext.so.6

libXmu.so.6

libXt.so.6

libc.so.6

libdl.so.2

libm.so.6

librcs.so

libstdc++-libc6.1-1.so.2

libtcl.so

libtclx.so

libtk.so
/bin/bash
/bin/sh
libc.so.6(GLIBC_2.0)
libc.so.6(GLIBC_2.1)
libm.so.6(GLIBC_2.0)
libm.so.6(GLIBC_2.1)
And the same header from rcslib.1586.rpm:
rtlinux
ld-linux.so.2
libc.so.6
libdl.so.2
libm.so.6
librcs.so
libstdc++-libc6.1-1.so.2
libc.so.6(GLIBC_2.0)
libc.so.6(GLIBC_2.1)
libm.so.6(GLIBC_2.1)
If you do an rpm -q on the above libraries, rtlinux, rcslib and librcs.so will not be found - They are just references to the realtime kernel and rcslib that comes with EMC.

A.36 How do I put EMC on floppies with ???.aaa, ???.aab ... files ?

Probably just put them all on the hard disk and go:

cat fname.[a-z][a-z][a-z] > fname.tgz

to aggregate them all into a single file, where fname is the filename you have (want) and the .tgz is the extension you expect (ie might be .tar.gz or .tar or .Z or???). If it is an executable, you'll need to set the executable bit with something like:

split -b 1440000 bigfile

which will make as many as necessary 1,440,000 byte smaller files named: xaa, xab, xac .... The last file will probably be shorter.

A.37 What is CVS and how it's related to EMC ?

CVS is Concurrent Versions System, used to maintain easily different copies of evolving software. More information from http://cvshome.org/ `http://cvshome.org/`. You can use CVS to fetch the "bleeding edge" version of EMC. These versions are the one's that are developed most actively so they can easily be much more difficult to install than (standard) versions (like BDI). Neverthless, the CVS is there for you to fetch the sources for EMC and if you are interested in using it, just visit the http://cvshome.org/ `http://cvshome.org/` and grab a basic knoledge of the CVS system.

A.38 Programming in "TCL/Tk"-language, for example: "How can I get the ESTOP-button to be red when the estop is on?"

You can do it using a process, something like the toggleEstop process but for the standard tkemc, modifying the updateStatus process and adding the configure commands to make changes to the estop button. But there is a caution here that I will add after I go through the code.

—snip of updateStatus process—

```
# now update labels
if {[emc_estop] == "on"} {
set estoplabel "ESTOP"
} elseif {[emc_machine] == "on"} {
set estoplabel "ON"
} else {
set estoplabel "ESTOP RESET"
}
```

—end of snip -> suggested new—

```
# now update labels
global estopbutton
if {[emc_estop] == "on"} {
set estoplabel "ESTOP" $estopbutton configure -bg darkgray -fg white -relief sunken
} elseif {[emc_machine] == "on"} {
set estoplabel "ON" $estopbutton configure -bg red -fg white -relief raised
} else {
set estoplabel "ESTOP RESET" $estopbutton configure -bg green -fg black -relief raised
} —end of modifications—
```

The caution is that NIST has a philosophy concerning their motion software. They are building their API so that multiple HMI's and GUI's can connect to and take control of a running motion control system. I walked into Fred's office a couple years ago with some beautiful and functional drawings of a control panel, the logic of which lasted a few seconds because I was using hard wired switches and HMI software that limited the control of the EMC to only that panel.

Now you can do that for your machine and no one will really care but it will be more difficult to get help because most of us here are working with the standard EMC running within the NIST philosophy. Those of us that build custom stuff to go with it accept the fact that we are out on a limb by ourselves.

Let me apply this back to the code that I wrote for you above and see if we can understand the difference between updateStatus and toggleEstop. Since updateStatus executes several times a second as a loop, putting something in it that references the running EMC "emc_estop" the values of the button will always be reasonably current regardless of what has caused a change to estop condition.

If I were to make those changes only to a process called toggleEstop and then ran that process when I pressed the estopbutton, I would have created a bit of code that does okay as long as it is the only process that has access to estop. If estop changes state for some other reason, the gui will not reflect the current state of the EMC accurately.

The down side of doing this kind of stuff in a loop process is that it stretches the time required for the loop to complete or demands more speed in the processor to run the program.

Ray

A.39 Another example of "TCL/Tk": Could someone please send me a quick .tcl that will write a number to variable # 1000? I just can't seem to get it.

The most basic way to set a variable is g-code

n10 #1000 = 50

or you can iterate it using

n140 #1000 = #1000 + 0.050

Then you can command an axis using the variable

n150 X#1000

See http://linuxcnc.org.handbook/gcode/variables.html `http://linuxcnc.org.handbook/gcode/variables.html` for more.

Both genedit and Set_Coordinates.tcl show examples of how to control a parameter value from Tcl. You should note that you will need to re-read the var file after the set using emc_task_plan_init if you are running emcsh. And there are other effects of that command as well that will have to be restored after the read.

Appendix B

Glossary of Common Terms Used in the EMC Documents

A listing of terms and what they mean. Some terms have a general meaning and several additional meanings for users, installers, and developers.

Acme Screw A type of lead-screw B that uses an acme thread form. Acme threads have somewhat lower friction and wear than simple triangular threads, but ball-screws B are lower yet. Most manual machine tools use acme lead-screws.

Axis One of the computer control movable parts of the machine. For a typical vertical mill, the table is the X axis, the saddle is the Y axis, and the quill or knee is the Z axis. Additional linear axes parallel to X, Y, and Z are called U, V, and W respectively. Angular axes like rotary tables are referred to as A, B, and C.

Backlash The amount of "play" or lost motion that occurs when direction is reversed in a lead screw B. or other mechanical motion driving system. It can result from nuts that are loose on leadscrews, slippage in belts, cable slack, "wind-up" in rotary couplings, and other places where the mechanical system is not "tight". Backlash will result in inaccurate motion, or in the case of motion caused by external forces (think cutting tool pulling on the work piece) the result can be broken cutting tools. This can happen because of the sudden increase in chip load on the cutter as the work piece is pulled across the backlash distance by the cutting tool.

Backlash_compensation - Any technique that attempts to reduce the effect of backlash without actually removing it from the mechanical system. This is typically done in software in the controller. This can correct the final resting place of the part in motion but fails to solve problems related to direction changes while in motion (think circular interpolation) and motion that is caused when external forces (think cutting tool pulling on the work piece) are the source of the motion.

Ball-screw A type of lead-screw that uses small hardened steel balls between the nut B and screw to reduce friction. Ball-screws have very low friction and backlash B, but are usually quite expensive.

Ball-nut A special nut designed for use with a ball-screw. It contains an internal passage to re-circulate the balls from one end of the screw to the other.

bridgeportio An I/O task B designed to work with a Bridgeport milling machine, having a variable speed spindle B, coolant, and lube pumps, and some other stuff.

CNC Computer Numerical Control. The general term used to refer to computer control of machinery. Instead of a human operator turning cranks to move a cutting tool, CNC uses a computer and motors to move the tool, based on a part program B.

EMC The Enhanced Machine Controller. Initially a NIST B project. EMC is able to run a wide range of motion devices.

EMCIO The module within EMCB that handles general purpose I/O, unrelated to the actual motion of the axes. A couple examples are bridgeportio B and minimillio B.

EMCMOT The module within EMC B that handles the actual motion of the cutting tool. It runs as a real-time program and directly controls the motors.

Encoder

Feed Relatively slow, controlled motion of the tool used when making a cut.

Feedback

Feed-rate Override A manual, operator controlled change in the rate at which the tool moves while cutting. Often used to allow the operator to adjust for tools that are a little dull, or anything else that requires the feed rate to be “tweaked”.

G-Code The generic term used to refer to the most common part programming language. There are several dialects of G-code, EMC uses RS274/NGC B.

GUI Graphical User Interface.

- **General** A type of interface that allows communications between a computer and human (in most cases) via the manipulation of icons and other elements (widgets) on a computer screen.
- **EMC** An application that presents a graphical screen to the machine operator allowing manipulation of machine and the corresponding controlling program.

Home A specific location in the machine’s work envelope that is used to make sure the computer and the actual machine both agree on the tool position.

ini file A text file that contains most of the information that configures EMC B for a particular machine

Joint_Coordinates: These specify the angles between the individual joints of the machine. Kinematics

Jog Manually moving an axis of a machine. Jogging either moves the axis a fixed amount for each key-press, or moves the axis at a constant speed as long as you hold down the key.

kernel-space

Kinematics The position relationship between world coordinates B and joint coordinates B of a machine. There are two types of kinematics. Forward kinematics is used to calculate world coordinates from joint coordinates. Inverse kinematics is used for exactly opposite purpose.Note that kinematics does not take into account, the forces, moments etc. on the machine. It is for positioning only.

Lead-screw An screw that is rotated by a motor to move a table or other part of a machine. Lead-screws are usually either ball-screws B or acme screws B, although conventional triangular threaded screws may be used where accuracy and long life are not as important as low cost.

MDI Manual Data Input. This is a mode of operation where the controller executes single lines of G-code B as they are typed by the operator.

minimillio An I/O task B designed to work with small table-top mills.

NIST National Institute of Standards and Technology. An agency of the Department of Commerce in the United States.

Offsets

Part Program A description of a part, in a language that the controller can understand. For EMC, that language is RS-274/NGC, commonly known as G-code B.

Rapid Fast, possibly less precise motion of the tool, commonly used to move between cuts. If the tool meets the material during a rapid, it is probably a bad thing!

Real-time Software that is intended to meet very strict timing deadlines. Under Linux, in order to meet these requirements it is necessary to install RTAI B or RTLINUX B and build the software to run in those special environments. For this reason real-time software runs in kernel-space.

RTAI Real Time Application Interface, see http://www.aero.polimi.it/ rtai/ `http://www.aero.polimi.it/~rtai/`, one of two real-time extensions for Linux that EMC can use to achieve real-time B performance.

RTLINUX See http://www.rtlinux.org `http://www.rtlinux.org`, one of two real-time extensions for Linux that EMC can use to achieve real-time B performance.

RS-274/NGC The formal name for the language used by EMC B part programs B.

Servo Motor

Servo Loop

Spindle On a mill or drill, the spindle holds the cutting tool. On a lathe, the spindle holds the workpiece.

Stepper Motor A type of motor that turns in fixed steps. By counting steps, it is possible to determine how far the motor has turned. If the load exceeds the torque capability of the motor, it will skip one or more steps, causing position errors.

TASK The module within EMC B that coordinates the overall execution and interprets the part program.

Tcl/Tk A scripting language and graphical widget toolkit with which EMC's most popular GUI's B were written.

World_Coordinates: This is the absolute frame of reference. It gives coordinates in terms of a fixed reference frame that is attached to some point (generally the base) of the machine tool.

Appendix A

Legal Section

Handbook Copyright Terms

A.1 GNU Free Documentation License Version 1.1, March 2000

A.1.1 GNU Free Documentation License Version 1.1, March 2000

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a

way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission. B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five). C. State on the Title page the name of the publisher of the Modified Version, as the publisher. D. Preserve all the copyright notices of the Document. E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices. F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms

of this License, in the form shown in the Addendum below. G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice. H. Include an unaltered copy of this License. I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence. J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission. K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein. L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles. M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version. N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine

any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http:///www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Appendix B

Index

Index