This is my updated simplified version of building from source for LinuxCNC with Ethercat as Scott's code has changed a few months back. Now you need to add the "golang" package, or you need to modify the "Makefile" in the "src" folder.

This is not the preferred way to install LinuxCNC with EtherCAT as Rod Webster (rodw) has put together a much easier way of installing everything from repo's. The source build is the option if you want a different variation than the repo install (that's me), or you want the newest version (such as dragon_hd a few years back), or your specific hardware… or its your last resort. (I have had all of these reasons happen at some point since 2013)

To clarify, make sure you only perform the source build and not perform the install from repos on top of the source build. Do NOT mix both, only take one path not both… Don't start with a source build and try to finish with the repo install, this will not work and you will only waste everyone's time on the forum, and your own time over a dirty build that will not work. If performed correctly, either way will give you a clean build.

If you are not verse with LinuxCNC, you need to go Rod's route as it is the preferred easy method and will work out of the box. Plus, he is on the forum practically all day every day to help you. I am rarely on the forum because of time… (CORBETT)

This was updated on December 7, 2024
This works for the following as I have checked all of the combinations:

- Debian 11 Bullseye -or- 12 Bookworm (32 bit -or- 64 bit)
- LinuxCNC 2.9 -or- 2.10 (master)
- Ethercat 1.5 -or- 1.6
- Sasha Ittner -or- Scott Laird's Ethercat Driver (I prefer Scott's)

Scott has advanced his code and that is the preferred choice. Sascha still does updates as of 2024, but Scott has been producing much more code and he is on the LCNC forum to help with problems.

Be aware that Scott's code needs a tiny workaround in the Makefile as I have not been able to build cleanly with only adding "golang", but that is most likely because I use a very minimal install with "--no-install-recommends lxqt-core" (qtdragon_hd looks awesome). I'm sure if you do a full install of Debian, then the "golang" package alone will probably work, I just haven't wasted the time to check.

For an ethercat source build to work, you must do a source build of LinuxCNC, this is called a Run-In-Place (RIP) as described in Part 2A. Do not install a LinuxCNC ISO or do a repo install as this will not work later on for the linuxcnc-ethercat driver part of the install. Just remember, this is a source build and all of the parts are built from source, so don't be tempted to try and get past a section with just installing that part and trying to get it to work with the other parts. An example would be to install the LinxuCNC ISO and then trying to do the source build of ethercat and the glue driver, this will not work.

THERE ARE 5 MAIN PARTS TO PUTTING A SYSTEM TOGETHER.

1) Install Linux Debian OS with RT kernel
2) Build LinuxCNC from source (rip build)
3) Build IgH ethercat from source
4) Build Sascha Ittner -or- Scott Laird's driver from source
5) Xml and configure LinuxCNC

PART 1 INSTALLING DEBIAN OS

If you already have your Debian OS setup and a RT kernel patched that you like, then you can skip on down to part 2a and start the build.

If you are starting fresh, start by downloading the Debian ISO image that you want for the build. Just do a search on the internet for downloading bullseye 11 or bookworm 12 and you will find the Linux download cd image page. Download the preferred ISO that you want or need, I like a net install without a GUI. You can use other OS's for your build, but these instructions are setup for Debian based systems.

When installing the OS, just do a simple graphical install or whatever you prefer. Just make sure that when it ask for "root" password, to leave this blank and skip this section so that you have "sudo" rights.

If your verse with Linux, then you already know what to do, just go do the commands… but if you are new and not sure of anything then follow along reading and only type the "**blue**" commands when prompted below.

Once you have Debian installed and have booted up the computer again, open a command line and bring the machine up to date with the latest packages. (for buster and older you will need to do "**apt-get**" for everything below.)
sudo apt update
sudo apt dist-upgrade

Now install the Preempt-RT kernel and modules. (This can be done from Synaptic, but I don't have it installed in my systems, and you will be on the command line anyway.)
USE THIS FOR 64 BIT **sudo apt install linux-image-rt-amd64**
USE THIS FOR 32 BIT **sudo apt install linux-image-rt-686-pae**

Next install the kernel headers
USE THIS FOR 64 BIT **sudo apt install linux-headers-rt-amd64**
USE THIS FOR 32 BIT **sudo apt install linux-headers-rt-686-pae**

After headers finish, then reboot
sudo reboot
Once computer is booted and ready, check that the RT kernel is installed. You will see "preempt_rt" in part of the full name. That sure is a lot easier than patching an "RTAI" kernel…
uname -a
If you do not see "preempt-rt", then reboot and go into "GRUB" and choose the rt kernel to boot up with.

After confirming you are using the RT kernel, then remove the old kernels by finding their name with grep
dpkg --list | grep linux-image
Then purge each of them with the following (pay attention and do not remove the RT kernel that you are using, just remove the other unused kernels)
sudo apt --purge remove linux-image-(*THE REST OF THE KERNEL NAME FROM GREP*)
Then reboot
sudo reboot
If you didn't install a GUI, you can choose anything you want but I usually stay with XFCE or LXQT (the merge of LXDE). I like a simple setup without bloat and you can do a somewhat minimal install with
sudo apt install sddm lxqt-core
If you are more verse with Debian then you can get lighter with
sudo apt install --no-install-recommends sddm lxqt-core
After you reboot and get your GUI working, then keep going…
(As a side note, if you do the "--no-install-recommends" throughout the build, you will get much less packages but you will need to do a few tweaks here and there such as "python3-gi" for the "pango" dependency. I just never said anything in the past as you need to know a little more to get the DE working)
THIS COMPLETES PART 1 OF GETTING THE REALTIME OS READY

PART 2A - LINUXCNC BUILD FROM SOURCE (RUN IN PLACE)

LinuxCNC 2.9 will only run on Debian 11 or newer

After reboot, open a command line window.
Next install the build packages "git" and "build-essential"
sudo apt install git build-essential
Make sure you are in your home directory, now clone the linuxcnc source code to a new directory called "linuxcnc-dev"
git clone https://github.com/LinuxCNC/linuxcnc.git linuxcnc-dev
Move to the "linuxcnc-dev" folder
cd linuxcnc-dev
Change to LinuxCNC version you want to build, you can choose "master" also for the latest LCNC version (2.10).
git checkout 2.9
Now change to the "debian" folder
cd debian
Then configure
./configure uspace no-docs
Then move back to the "linuxcnc-dev" folder
cd ..
Check for any missing build dependencies
dpkg-checkbuilddeps
(There will be several dependency packages to install. Just copy selection and paste to save all that typing, otherwise just take time and type them all out, it's quite a few)

Install the build dependencies with
sudo apt install "*LIST OF DEPS COPIED FROM DPKG-CHECKBUILDDEPS*"
Install all of the packages needed, then check deps again
dpkg-checkbuilddeps
Once all the dependencies have been installed then move to the "src" folder
cd src
Then after getting into "src" folder, generate the configure script
./autogen.sh
then configure with realtime uspace
./configure --with-realtime=uspace --disable-build-documentation-translation
Then make
make
Allow access to hardware
sudo make setuid
Setup rip environment
. ../scripts/rip-environment
Start up linuxcnc
linuxcnc
When the linuxcnc configuration box appears, the "axis" sim will already be highlighted. Choose this highlighted "axis" config and also check the box below that has "created desktop shortcut". When you click "ok" it will try to start up linuxcnc but it will **fail** if using lcnc2.9 or master as the below qtvcp packages have not been installed yet. Do not close the command line, just keep going below for the script.

PART 2B – QTVCP PACAKAGES FOR DRAGONHD – RIP

Start the qtvcp script for the installation
~/linuxcnc-dev/lib/python/qtvcp/designer/install_script
Choose #2 when prompted for the rip install.
2
After this is installed, linuxcnc will start up and work. As a side note, qtdragon_hd will work at this point. Don't close out the command line window, just keep going down to part 3
THIS COMPLETES PART 2, LINUXCNC IS SETUP READY FOR ETHERCAT

PART 3 GETTING ETHERCAT INSTALLED FROM SOURCE

Change back to home folder
cd ~
Once back in the home folder, clone the ethercat repository. Just make sure you are in the home folder and not in another location so that the "ethercat-master" folder is created in the correct location. (Use the "pwd" command to check)
git clone https://gitlab.com/etherlab.org/ethercat.git ethercat-master
Change to the ethercat folder
cd ethercat-master
Change to the stable 1.6 branch
git checkout stable-1.6

Build the software as regular user, not as root
Start up the build
./bootstrap
Get ethernet infomation
sudo lspci -v
Example: My network is Intel Pro/100, Kernel driver in use: e1000e (I use generic myself as I build for several setups that don't have the "e1000e" driver and just clone the drives for each machine, I need to start building specific for each device though to get that little extra speed)
Configure to the type of module needed (there are other switch options, but this is what I use for a general build)
./configure --sysconfdir=/etc/ --disable-8139too --enable-userlib --enable-generic
Then make with modules
make all modules

Then install the software as root
Become root
sudo su

Install the software as root
make modules_install install
Dependency modules command
depmod
Exit root
exit

Then get service working
Get ether address – write down or copy & paste the mac address
ip a
Open the ethercat configuration file with pico and add your eth0 address
sudo pico /etc/ethercat.conf
Once the file is open, set your master0 device to your mac address and device modules to generic. Scroll down until you see these two parts. (they are not together, but separated throughout the code)

MASTER0_DEVICE="**a1:b2:c3:d4:e5:f6**" (whatever your mac address is)
DEVICE_MODULES="**generic**" (whatever module you configured and built with)

Then save and exit pico with "ctl o", then hit the "enter" button, and then "ctl x".
ctl o
ctl x

After you have exited pico and back on the command line
You must do the following and not skip as it sys-links
Enable the ethercat service
sudo systemctl enable ethercat.service
Start the ethercat service
sudo systemctl start ethercat.service
Check the ethercat status
sudo systemctl status ethercat.service
Change permissions
sudo chmod 666 /dev/EtherCAT0
Check that its working with.
ethercat master
You should see numbers in the lines, if not and it's all zero's then something is not linked correctly in ethercat or linux. Make sure it shows a change after doing the "ethercat master" command. Make sure to do the following below to get permission setup properly.

Open in pico to give ethercat port startup permission
sudo pico /etc/udev/rules.d/99-ethercat.rules

Once the file is open, add the following:
KERNEL=="EtherCAT[0-9]", MODE="0777"
Then save and exit pico with "ctl o", then hit the "enter" button, and then "ctl x".
ctl o
ctl x

Once back on the command line, then reload the rules
sudo udevadm control --reload-rules
Now you have ethercat… next we just need to glue it to LCNC
THIS COMPLETES PART 3 OF GETTING ETHERCAT INSTALLED

PART 4 - SETUP LINUXCNC-ETHERCAT DRIVER

Change back to home folder
cd ..
Clone the ethercat repository. Just make sure you are in the home folder and not in another location so that the "linuxcnc-ethercat" folder is created in the correct location for these instructions.
git clone http://github.com/linuxcnc-ethercat/linuxcnc-ethercat.git linuxcnc-ethercat

Note: You do not have to perform the "realtime.mk" fix anymore, as both sets of code have the file fixed.

But you do have to comment out a few lines in the source code for Scott Laird's driver. He is using GoLang and it will not build even with getting the dependencies met, but that could be from me using a minimal install. It's a simple easy modification of the "Makefile" in the "src" directory.

Change to the "src" directory
cd linuxcnc-ethercat/src/
Once you are in the "src" folder, now open the "Makefile" with pico.
sudo pico Makefile
Now comment out part of line 83, 92, and all of lines 113 through 121. Just add the "#" where shown below:

At line 83 of the code, comment out "lcec configgen"
```
user: lcec_conf lcec_devices #lcec_configgen
```
At line 92 of the code, comment out the whole line
```
#	cp lcec_configgen $(DESTDIR)/usr/bin/
```

At lines 113 - 121 of the code, comment out the whole line
```
#lcec_configgen: configgen/*.go configgen/*/*.go
#	(cd configgen ; go build lcec_configgen.go)
#	cp configgen/lcec_configgen .

#configgen/devicelist: configgen/devicelist.go
#	(cd configgen ; go build devicelist.go)

#configgen/drivers/drivers.go: configgen/devicelist lcec_devices
#	(cd configgen ; go generate)
```

Then save and exit pico with "ctl o", then hit the "enter" button, and then "ctl x".
ctl o
ctl x

Now move back to the "linuxcnc-ethercat" folder.
cd ..
Make sure you are in the "linuxcnc-ethercat" folder, then make clean
make clean
Then make the package
make
NOTE: IF YOU HAVE A FAIL ON THE MAKE, THEN GO BACK TO THE "linuxcnc-dev/src" FOLDER AND DO THE ". ../scripts/rip-environment" AGAIN AND START LINUXCNC TO GET THE FILES LOADED

Then do the install
make install
Now we need to edit the file /etc/ld.so.conf and add your path
sudo pico /etc/ld.so.conf
Once the file is open in pico, add the following down below the "include" line
/usr/local/lib
Once the above had been added,
Then save and exit pico with "ctl o", then hit the "enter" button, and then "ctl x".
ctl o
ctl x

After saving and exiting pico, change to root
sudo su
Load the library (with -v so it prints all the links)
ldconfig -v
Exit root
exit
Even though it looks like you should do something else for the install, you are finished and everything should be working. Now it is time to go make an axis config and add the ethercat xml file to get it all linked and a working system.
THIS COMPLETES PART 4 OF GETTING ETHERCAT DRIVER INSTALLED

PART 5 - SETUP LINUXCNC AND XML

The following info is much easier to understand if you are verse with LinuxCNC. Should you get lost or confused, don't hesitate to reach out to one of us on the forum and we will help you get your config working.

***NOTE* (Grotius) has built an XML configurator that has been pinned on the forum…** (I will be trying this in the future to check it out, just too busy at moment)

SETTING UP THE XML FILE
There are several examples on the web, but you simply put the following code in a plain text file and rename it "ethercat-conf.xml". Then place the xml file in your "/linuxcnc-dev/configs/" folder.

```
<masters>
  <master idx="0" appTimePeriod="1000000" refClockSyncCycles="1">
    <slave idx="0" name="D1" type="EK1100"/>
  </master>
</masters>
```

With the above code you can get linuxcnc with ethercat to startup and work even if you do not have a beckhoff ek1100 coupler or any other ethercat hardware. This helps you learn a good bit before investing in hardware.

If you have an ek1100 coupler and other beckhoff terminals, you will need to make your xml match what you have for the physical hardware. My recommendation is to get everything working with the ek1100 only and then add 1 terminal slice at a time, while re-working your xml until you get the hang of getting it working… then in the future you can just build the xml to what you have for equipment.

One thing to keep in mind, if you have a "bad" beckhoff terminal piece, when you try to start up linuxcnc it will fail and definitely confuse you on what is wrong. That is a good reason to add 1 terminal slice at a time if your parts are used from ebay or other sources, just in case you have a bad non-working piece. Don't forget that you can reset the terminal and sometimes that will get it working again. If it's a used piece, you never know what the person before you did and what they were using the terminal specifically for. You can use TwinCAT or the command line to reset the terminal, but TwinCAT is failsafe. You can "brick" a terminal from the command line, but I usually don't waste time on switching over just for doing a factory reset. A shout out to Albert (chimeno) for teaching me the procedure on TwinCAT… (ArcEye) for the command line…

SETTING UP LNUXCNC
Once you have your xml setup and in place, then you simply need to make a linuxcnc "axis" config and add the ethercat code to your "hal" file. For simplicity, you can also add the code to your "postgui.hal" file as this will work and get linuxcnc to startup.

Once you figure out how it works, you can move the code to your main "hal" file as you like. When moving the code to your other "hal" file, make sure to follow linuxcnc rules and put the components and functions in the correct places. Don't forget to correct the file name marked in red below to your correct name.

```
loadusr -W lcec_conf /home/NAME/linuxcnc-dev/configs/ethercat-conf.xml
loadrt lcec

addf lcec.read-all servo-thread
addf lcec.write-all servo-thread

net ec-slaves-responding lcec.slaves-responding
net ec-link-up lcec.link-up
net ec-all-op lcec.all-op
```

Once you have a configuration working where linuxcnc axis starts up without failing, you can check to see if ethercat (lcec) is working by clicking on the "machine" tab at the top, then from the drop down menu click on "show hal configuration", then the "hal show" box will appear. Look on the left side and click on the "pins" tab, when the list drops down you will see "lcec". When you click on "lcec" the halshow box will show all of the ethercat (lcec) pin info. Just seeing "lcec" alone tells you that you have it fully installed and working. If "lcec" is not in the "pins" list, then you have something wrong. First, if you are able to see your terminals with "ethercat slaves", then it is most likely that the "linuxcnc-ethercat" build did not fully build correctly, so just go try installing that part again and then re-try everything. If it was the ethercat build, then once you get it built correctly, linuxcnc will then show the (lcec) under pins section. This is the fastest way to get it all diagnosed and working.

PART 6 - LET'S SEE ETHERCAT WORK IN THE REAL WORLD.

Let's add a real physical external "emergency stop" button wired to an el1008 for input to LinuxCNC.

(with power off) add the Beckhoff el1008 terminal to your ek1100 coupler. Next, wire your emergency stop button looped from +24v power to the el1008 terminal input-1. Next you need to re-work your xml file and add the el1008 terminal. Make your code look like below by adding the section in "blue" then save the file:

```
<masters>
  <master idx="0" appTimePeriod="1000000" refClockSyncCycles="1">
    <slave idx="0" name="D1" type="EK1100"/>
    <slave idx="1" name="D2" type="EL1008"/>
  </master>
</masters>
```

Once your xml file is re-worked and you are able to start up LinuxCNC without it failing, then you are ready to write more ethercat code to your "postgui.hal" file.

Before going to the "postgui.hal" file, you must find the following code section in the "core.hal" file or whichever "hal" file it is located in.
net estop-loop iocontrol.0.user-enable-out iocontrol.0.emc-enable-in
Now let comment out the line with adding # at the beginning.
#net estop-loop iocontrol.0.user-enable-out iocontrol.0.emc-enable-in

Now let's write one ethercat pin to your "postgui.hal" file. You will only add one line to the previous code and it must be below all of the previous code. Make your code look like below by adding the section in "blue" then save the file and start lcnc. Don't forget to change the "name" in red just like you did before.

```
loadusr -W lcec_conf /home/NAME/linuxcnc-dev/configs/ethercat-conf.xml
loadrt lcec

addf lcec.read-all servo-thread
addf lcec.write-all servo-thread

net ec-slaves-responding lcec.slaves-responding
net ec-link-up lcec.link-up
net ec-all-op lcec.all-op

net voltage-on  iocontrol.0.emc-enable-in   lcec.0.D2.din-0
```

Now start up LinuxCNC and operate the E-stop to see if it changes on the screen. If everything is correct, you should see it operate in the LinuxCNC window. If it does not work, try using the mouse and clicking the e-stop to see if it works. If it works by the mouse and not the physical button, then the code is not correct or something at the terminal. Look to see if the green led at the top of the terminal changes when you operate the button. If it does, the it is most likely the code is at fault.

If for some reason that LinuxCNC fails to start, backtrack by removing the terminal and change your xml back to make sure you can get LinuxCNC to start again with just the ek1100 coupler, if so, then double check the xml code and that you saved the file… then double check that you have a good terminal that is not a defective bad unit.

<<<**ADDING SOME MORE ETHERCAT PINS**>>>
Here is some extra code to add combined home/limit switches to that same el1008 that you wired your e-stop switch into. Same concept as the e-stop, just loop your +24v power to the switch and then you can wire them into the el1008 input-2 & input-3 connections for using the code below:

```
net both-home-x => joint.0.neg-lim-sw-in joint.0.home-sw-in <= lcec.0.D2.din-1
net both-home-z => joint.2.pos-lim-sw-in joint.2.home-sw-in <= lcec.0.D2.din-2
```

<<<**ADDING EXTERNAL PENDANT PUSH BUTTONS WITH ETHERCAT**>>>
Here is some extra code to add external physical push button switches for your pendant to that same el1008 that you wired your e-stop switch into. Same concept as the e-stop & limit switches, just loop your +24v power to the switch and then you can wire them into the el1008 input-2 & input-3 connections for using the code below, just add below all the other code in your "postgui.hal":

```
net jog-speed             halui.jog-speed
net remote-jog-x-plus     halui.jog.0.plus      lcec.0.D2.din-3
net remote-jog-x-minus    halui.jog.0.minus     lcec.0.D2.din-4
```

Don't forget if you add another Beckhoff terminal such as an el2008, then you need to not only add the "pins" in your "hal" file, you also need to re-work the "xml" file to add the "slave".

<<<**ADDING OUTPUT WITH EL2008**>>>
Let's add an el2008 and then set it up for outputting to a relay or some other function. You will need to go find this code in your other hal files, as if you put this in your "postgui.hal" it will most likely fail as you would have the "net & motion" code already linked in another "hal" file.

```
net spindle-on     <=  motion.spindle-on        =>   lcec.0.D3.dout-0
net spindle-cw     <=  motion.spindle-forward   =>   lcec.0.D3.dout-1
net spindle-ccw    <=  motion.spindle-reverse   =>   lcec.0.D3.dout-2
net spindle-brake  <=  motion.spindle-brake     =>   lcec.0.D3.dout-3
```

Next you need to re-work your xml file and add the el2008 terminal. Make your code look like below by adding the section in "blue" then save the file:

```
<masters>
  <master idx="0" appTimePeriod="1000000" refClockSyncCycles="1">
    <slave idx="0" name="D1" type="EK1100"/>
    <slave idx="1" name="D2" type="EL1008"/>
    <slave idx="2" name="D3" type="EL2008"/>
  </master>
</masters>
```

OK, IF YOU MADE IT TO THIS POINT, THEN CONGRADULATIONS ON GETTING LINUXCNC WITH ETHERCAT WORKING!!! IT'S AN AWESOME FEELING TO SEE IT FIRE UP AND WORK!!!

ONCE AGAIN, THANKS TO ALL OF THE PLAYERS THAT HAVE MADE THIS HAPPEN OVER THE YEARS. IT JUST KEEPS GETTING BETTER AND BETTER!!! (CORBETT)