How-to Setup EtherCAT for LinuxRT

<This page is currently under active development/revision. Please contact Jesus Vasquez <jvasquez@slac.stanford.edu> for more information or questions.>

This page covers how to setup a new EtherCAT device and integrate into the EPICS Control System. The following topics are included:

EtherCAT Resources & Reference Material

The following resources and reference material provide an introduction to the EtherCAT technology:

- EtherCAT Overview
- EtherLAB Master & Kernel Modules
- Bus Scanner & EPICS Asyn Driver

linuxRT PC Set-up

In the SLAC environment, we are currently running EtherCAT on industrial PC's running linux with a pre-emptive real-time patch (linuxRT). Before configuring for EtherCAT, the very first step is to set-up a new linuxRT host.

- For new users, please reference the following page: Diskless Boot of LinuxRT or Centos7 - Quick Start-up Highly recommend viewing this diagram: LinuxRT_Diagram

Table 1

Facility	LinuxRT Account	LinuxRT Group	Comment
Development	laci	lcls	
LCLS	laci	lcls	
LCLS-II			not yet available
FACET	flaci	facet	linuxRT is built with laci as the default account, users must add facet group, user and password
Test-Facility	acctf	?	linuxRT is built with laci as the default account, users must add test-facility group, user and password

Verification checks on linuxRT machines:

- Please see $IOC/cpu-<bld>-<name>/README on instructions on how to power cycle machines remotely through the use of ipmi commands
- Once you set-up your linuxRT machine and boot up on development for instance, log into laci on your linuxRT system (ref: Table 1). (*ssh laci@cpu-<bld>-<name>*)
- Test if your machine is running real-time linux using *uname -a*

```
$ uname -a
Linux cpu-b34-mc23 3.18.11-rt7 #5 SMP PREEMPT RT Wed Mar 16 19:03:38 PDT 2016 x86_64 GNU/Linux
$ /lib/libc.so.6
GNU C Library (Buildroot) stable release version 2.20, by Roland McGrath et al.
Copyright (C) 2014 Free Software Foundation, Inc.
```

EtherCAT linuxRT Configuration

We are currently using Etherlab's EtherCAT Master module, which can be found under the package area. The kernel-modules.cmd file needs to be updated to include the following:

1. Location of EtherCAT kernel drivers
2. Install kernel drivers
3. Network configuration
4. EtherCAT system configuration & set-up
5. Start scanner process

Update the following lines of code in your $IOC/cpu-<>-<>/kernel-modules.cmd. The following code excerpts are from $IOC/cpu-b084-hp03/kernel-modules.cmd

This current code stanza defines the location for the EtherCAT kernel drivers.

```
# ================================================================

# Define kernel module driver location

# ================================================================



ETHERCAT_DRIVER=$KERNEL_DRIVER_HOME/ethercat/buildroot-2015.02-x86_64
```

The following stanza provides the network configuration, installs kernel drivers, and performs some EtherCAT system configuration & set-up

```
# ========================================

# Install EtherCat kernel drivers + Set-up

# ========================================

# Need to configure another ethernet port on cpu-b084-hp03 for ethercat

# MUST configure another ethernet port for ethercat

# ifconfig eth1 <insert_your_ethercat_ip_address> netmask <insert_your_netmask>

# Recommend using the following logic to configure based on the port's MAC Address

EthCat_MAC_ADDR=<insert_your_MAC_addresss_for_your_etherCAT_port>

eth_cat=$(ifconfig -a | grep "HWaddr ${EthCat_MAC_ADDR}" | awk '{print $1}')

ifconfig ${eth_cat} <insert_your_ethercat_ip_address> netmask <insert_your_netmask>



# insert MAC address for eth1 after main_devices

# the ethercat slaves/devices are connected at ${eth_cat} (for this specific configuration)


insmod $ETHERCAT_DRIVER/master/ec_master.ko main_devices=${EthCat_MAC_ADDR}

insmod $ETHERCAT_DRIVER/devices/ec_generic.ko


/sbin/ifconfig ${eth_cat} up


ln -s $ETHERCAT_DRIVER/tool/ethercat /sbin/ethercat


sleep 5

chmod o+rw /dev/EtherCAT*

# ============================================================
```

Next, need to add the following lines to start the scanner process up automatically.

```
# ================================================================
# Now we start the ethercat scanner process under the linux screen program:
# This will allow us to attach/detach to our Ethercat scanner process
# ================================================
cd $EPICS_IOCS/cpu-b084-hp03
su laci -c $EPICS_IOCS/vioc-b084-tmp1/scannerStartup.sh
# ================================================
```

Verification checks on linuxRT machines:

- Test if your machine has the Etherlab EtherCAT driver module loaded.

```
$ ethercat version
IgH EtherCAT master 1.5.2 2eff7c993a63
$ lsmod | grep ec
ec_generic            3453  1
ec_master           240382  3 ec_generic
```

- If the slaves are connected with the setup, the *ethercat* command line tool can help visualize it.

```
$ ethercat slaves
0  0:0  OP  +  E1250-EC-UC V1.7b1
```

- Another helpful *ethercat* command line took is *ethercat master*

Configuring the $IOC/sioc-<>-<> Directory

The $IOC/vioc-<>-<> directory created for this EtherCAT application needs the follow items:

- scannerStartup.sh (where should we put a "master" scannerStartup.sh for people to copy/point to)
- Proposal: put generic scannerStartup.sh under $IOC/common/ethercat. Script has been modified to pass in the vioc name from when the script is called from the kernel-modules.cmd
  - so vioc specific information (for vioc-ltu0-mc01) was moved to the kernel-modules.cmd => have to think about use case where multiple subsystems share cpu (see above solution)
  - OR do we keep it in kernel-modules.cmd (this should be apart of the linuxRT discussion)
- Create a bin directory (currently under discussion to rename this "bin" directory to avoid confusion)
- Proposal: rename "bin" to "scanner" and use symbolic links to point to scanner binary and scanner.xml in epics application
- In the bin directory, create a symbolic link "scanner" that points to your scanner binary in your epics application (<top>/bin/linuxRT_glibc-x68_64 /scanner)
  - ~~OH! scanner binary is NOT specific to your application. Should point to the package area.~~
- In the bin directory, create a symbolic link "scanner.xml" that points to your scanner.xml in your epics application (<top>/etc/scanner.xml)

Example from vioc-b084-tm01:

```
$ ls -all

README          bin             iocSpecificRelease  scannerStartup.sh   screenrc        startup.cmd

$ ls -all bin

total 6

drwxrwxr-x    2 13620    1006         2048 Jul  8 15:14 .

drwxrwxr-x    3 13620    1006         2048 Jul 14 15:18 ..

lrwxr-xr-x    1 13620    1006          123 Jul  8 11:21 scanner -> /afs/slac/g/lcls/epics/R3-14-12-4_1-1/iocTop/users/ababbitt/LCLS-II
/TEMPERATURE/MAIN_TRUNK/bin/linuxRT_glibc-x86_64/scanner

lrwxr-xr-x    1 13620    1006          106 Jul  8 15:14 scanner.xml -> /afs/slac/g/lcls/epics/R3-14-12-4_1-1/iocTop/users/ababbitt/LCLS-II
/TEMPERATURE/MAIN_TRUNK/etc/scanner.xml
```

EtherCAT Template for Creating Epics Application

When starting a new EtherCAT application, it is highly recommended to start a new epics application and vioc using the *slac_ethercat* template.

The following command creates a new epics command using the *slac_ethercat* template.

```
makeBaseApp.pl -t slac_ethercat <insert_your_application_name>
```

Generate a vioc for your application. We are using the following format for linuxRT IOC's: vioc-<area>-<subsystem>

```
makeBaseApp.pl -i -t slac_ethercat vioc-<>-<>
```

Choose your target architecture:

```
The following target architectures are available in base:
    linux-x86
    vxWorks-ppc604_long
    vxWorks-mpc8540
    vxWorks-ppc604_altivec
    linux-x86_64
    linuxRT_glibc-x86_64
    linuxRT_glibc-i686
    RTEMS-beatnik
    RTEMS-mvme3100
    RTEMS-uC5282
What architecture do you want to use? linuxRT_glibc-x86_64
```

Choose the application that you want your new vioc to boot:

```
The following applications are available:
   <insert_your_application_name>
What application should the IOC(s) boot?
>> <insert_your_application_name>
```

You have now created a baseline EtherCAT application.

(For more reference material on getting started with EPICS: CH. 2 Getting Started)

Configuring EtherCAT Epics Application

The EtherCAT application has two parts.

1. The scanner process which scans the bus
2. The EPICS IOC application

Before being able to successfully run the application, some clean-up and configuration are necessary in the following areas:

1. <top>/etc/
2. <top>/configure/RELEASE
3. <top>/<app_name>/Db/Makefile
4. <top>/iocBoot/vioc-<>-<>/st.cmd

The first step specific to an EtherCAT application, is creating a chain.xml file for the scanner process. The scanner process will open up a unix socket for communication with the IOC. If using the *slac_ethercat* template, you should already have a directory named /etc under <top>. Otherwise, you will been to create a directory called /etc. This directory will consist of a Makefile, chain.xml, and a scanner.xml. The scanner.xml will be automatically generated upon building the application.

The Makefile should consist of the following lines:

```
TOP=..

include $(TOP)/configure/CONFIG

install:
    /usr/bin/python $(ECASYN)/etc/scripts/expandChain.py  chain.xml > scanner.xml

clean:
    rm -f scanner.xml
    rm -f EK1101-EtherCAT.template
distclean: clean
realclean: clean
uninstall:
```

However, the chain.xml needs to be manually created based off the devices in your EtherCAT chain.

This is an example of a chain with just one motion control device (ref: $APP/users/namrata/EtherCATest/etc):

```
<chain>
    <device type_name="E1250-EC-UC" revision="0x00010007" position="0" name="LINMOTOR0" />
</chain>
```

Here is another example with two slave devices for an EtherCAT coupler and an analog output device for reading temperature from RTD's (ref: $APP/users /ababbitt/LCLS-II/TEMPERATURE/etc):

```
 <chain>
     <device type_name="EK1101" revision="0x00120000" position="0" name="COUPLER0" />
    <device type_name="EL3202" revision="0x00100000" position="1" name="ANALOGINPUT" />
</chain>
```

The device name needs to match the name given by the xml device templates found here: $PACKAGE_TOP/ethercat/<ethercat_version_number>/etc/xml (need to confirm....).

To obtain the revision number for the device you will need to use the slaves *ethercat* command line tool on your linuxRT machine (aka the EtherCAT Master).

```
$ ethercat slaves -v -p0

=== Master 0, Slave 0 ===

Device: Main

State: OP

Flag: +

Identity:

  Vendor Id:       0x4c4e5449

  Product code:    0x009606e3

  Revision number: 0x00010007

  Serial number:   0x1bd200e7
```

Upon making your application, the scanner.xml file will be automatically generated in the /<top>/etc directory. A scanner binary will be generated under /<top>/bin. You will need to create symbolic links to both of these items under $IOC/vioc-<>-<>/bin directory as previously mentioned.

NOTE: If you are using a brand new module, it may not currently exist in the EtherCAT module. You can check whether you device is supported by looking in the epics modules ethercat area under the /etc/xml directory.

Epics ethercat module: ECASYN=$(EPICS_MODULES)/ethercat/$(ECASYN_MODULE_VERSION)/etc/xml - contains the xml files from the vendor for the device families currently supported.

Any new devices should be integrated into the EtherCAT module. Please contact the EtherCAT module owner:

- Jesus Vasquez: jvasquez@slac.stanford.edu

Now that the chain.xml file has been created to generate the scanner.xml file, the second part is to configure and clean-up your epics application to only include the elements that you need.

- Under <top>/configure/RELEASE, remove any modules you are not using for your application (the *slac_ethercat* is a motion control based example)
- Under <top>/<app_name>/Db/Makefile, only include the templates to be installed that you are using (each slave device should have an associated template)
  Here is an example of using the *slac_template* for a non-motor application:

  ```
  ## Install Protocol File:

  #DB_INSTALLS += $(CAENN1470)/db/caenN1470.proto

  ## ================================================

  DB_INSTALLS += $(ECASYN)/db/EK1101.template

  DB_INSTALL S+= $(ECASYN)/db/EL3202-0010.template

  #DB_INSTALLS +=$(MOTOR)/db/basic_asyn_motor.db


  DB_INSTALLS += $(ASYN)/db/asynRecord.db
  ```

  NOTE: Should be using DB_INSTALLS to be using the common templates found in the epics modules (want to avoid unnecessary copy and paste of template files or database records)

- <top>/iocBoot/vioc-<>-<>/st.cmd
  Update the st.cmd to only load the database records specific to your application. The st.cmd for all EtherCAT applications must include the following items. (Please reference the st.cmd file after using the *slac_ethercat* template for additional motion control related steps.)

```
# ====================================================
# Init EtherCAT: To support Real Time fieldbus
# ====================================================
# EtherCAT AsynDriver must be initialized in the IOC startup script before iocInit
# ecAsynInit("<unix_socket>", <max_message>)
# unix_socket = path to the unix socket created by the scanner
# max_message = maximum size of messages between scanner and ioc
ecAsynInit("/tmp/sock1", 1000000)
```

Load the records for the EtherCAT components and pass in the appropriate macros as defined in the template files.

NOTE: The "PORT" name is the name defined in the chain.xml file (also is located at the end of the scanner.xml file).

```
# ============================================================================================================
# Load Additional databases:
# ============================================================================================================
# Load the database templates for the EtherCAT components
# dbLoadRecords("db/<template_name_for slave_module>, <pass_in_macros>)
dbLoadRecords("db/EK1101.template", "DEVICE=TEMP:BCM_EK1101,PORT=COUPLER0,SCAN=1 second")
dbLoadRecords("db/EL3202-0010.template", "DEVICE=TEMP:BCM_EL3202,PORT=ANALOGINPUT,SCAN=1 second")
```

Verification checks on development machine (preferably lcls-dev3):

Check that your python version is correct (the scripts used to generate the chain.xml are python-based):

- Python version: Python 2.7.9
- export the correct version to the path

```
 $ python --version
 Python 2.7.4
$ export PATH=/afs/slac/g/lcls/package/python/python2.7.9/linux-x86_64/bin/:$PATH
$ python --version
Python 2.7.9
```

Debugging Tips

- Check that the ethercat kernel module is installed: *ls /dev/EtherCAT0*
- Manually start scanner process on EtherCAT Master (linuxRT machine)

```
On a linuxRT machine:
<path_to_scanner_binary> -f <rate_in_hz> <path_to_scanner_xml> <path_to_unix_socket_created_by_scanner>
<max_message_size>

 Example (from directory containing scanner binary):
./scanner -f 100 ../../etc/scanner.xml /tmp/sock1 100
```