WHEN YOU NEED TO BE SURE

CyFlex® EtherCat Setup

Version 8

October 18, 2023

Developed by SGS North America, Inc.

Version History

Version	Date	Revision Description
1	2/16/2016	Initial publication
2	8/23/2018	Format with SGS brand
3	9/30/2019	Miscellaneous revisions throughout
4	4/3/2020	Retrofit to new template Revamp *Section 3 Installing EtherCat* on page 4 with installation instructions for the new RPM
5	12/15/2020	Add *Appendix A. Probing and Configuring Beckhoff I/O Modules* on page 24
6	6/6/2022	Updated hypertext link to Gantner website `test.commander` information in *Section 7 Test.Commander Settings* on page 21
7	6/12/2023	Added step 3, update the `ifcfg` file to enable the port in *Section 3.3 Assigning an EtherCat Network Interface* on page 4.
8	10/18/2023	In *Section 2 Setup Requirements* on page 2: • Added hypertext linked cross-references to usage help on cyflex.com • Added category Channel type drivers

Document Conventions

This document uses the following typographic and syntax conventions.

- Commands, command options, file names or any user-entered input appear in Courier type. Variables appear in Courier italic type.

  Example: Select the `cmdapp-`*`relVersion-buildVersion`*`.zip` file….

- User interface elements, such as field names, button names, menus, menu commands, and items in clickable dropdown lists, appear in Arial bold type.

  Example: **Type**: Click **Select Type** to display drop-down menu options.

- Cross-references are designated in Arial italics.

  Example: Refer to *Figure 1*…

- Click intra-document cross-references and page references to display the stated destination.

  Example: Refer to *Section 1* Overview on page 1.

  The clickable cross-references in the preceding example are *1*, *Overview*, and on page 1.

CyFlex Documentation

CyFlex documentation is available at https://cyflex.com/. View **Help & Docs** topics or use the **Search** facility to find topics of interest.

Table of Contents

List of Figures

LIST OF TABLES

1 Overview

This document describes basic setup on a CyFlex test cell for using Ethernet Control Automation Technology (EtherCAT) over a network to control engine testing. EtherCAT works with standard Ethernet technology and is compatible with Ethernet protocols, including Transmission Control Protocol/Internet Protocol (TCP/IP).

An open-source driver from EtherLab works as a real-time kernel module attached to the Linux OS on the test cell – and functions as the EtherCAT master (controller). The master has a Media Access Controller (MAC) address, identifying its Ethernet port on the network.

Note:

EtherLab is a technology that combines hardware and software for testing and automation. For more information about how the EtherLab driver works, refer to documentation for the current release: http://www.etherlab.org/en/ethercat/index.php

CyFlex tasks (applications) interface with the master to exchange process data with EtherCAT slaves (I/O and measuring modules). A CyFlex task called `ecat_mon` interprets EtherCAT channels into REAL CyFlex variables. Another CyFlex task called `ecat_srvr` can act as an EtherCAT master and start Process Data Object (PDO) transfer of all input and output channels (at 100 Hz).

Modules (slaves) may be configured for various engine testing requirements using a Windows program from Gantner called `test.commander`. This program functions as the FTP client for reading and writing the configuration files and so has access to all measuring system parameters. `Test.commander` is packaged with ICP100 configuration software and an integrated viewer called `test.viewer`. Refer to *Section 7 Test.Commander Settings* on page 21.

2 Setup Requirements

The following are required to set up EtherCAT:

- Test cell computer running CyFlex and Linux OS
- CyFlex version 6.2.14 and up
- Network connection (Ethernet)
- EtherCAT Master driver
- EtherCAT specification file(s)
- CyFlex applications:
  - `ecat_mon`, refer to `ecat_mon` usage help on cyflex.com.
  - `ecat_srvr`, refer to `ecat_srvr` usage help on cyflex.com.
- Channel type drivers:
  - `ai_ecat`
  - `di_ecat`
  - `ao_ecat`
  - `do_ecat`
- `Test.commander` software
- PC running Windows OS (for `test.commander`)
- Go-script

2.1 Hardware Connection

The EtherCAT device driver application supports connecting the test cell to a TCP network via the computer's Ethernet port and cable.

2.2 Communication Parameters

Having the MAC address of the master (Ethernet port) on-hand may save time while configuring the setup:

At a shell window, enter:

```
/sbin/ifconfig
```

The display should list the MAC for each adapter device. For example:

```
eth1 Link encap:Ethernet HWaddr 00:03:FF:FF:FF:F9
```

Note:

Configure slave modules and the EtherCAT interface using the `test.commander` program. Refer to *Section 7 Test.Commander Settings* on page 21.

2.3 Files

The files needed for setting up EtherCAT on the test cell are typically kept in the directories shown in the table below. Once CyFlex is installed and the cell is up and running, users typically create or copy a Go-script from another cell; customized for the tasks the cell performs. This saves time modifying the file manually.

Users can create more than one instance of a configuration (specification) file. This means one specification file for each copy of the CyFlex task being run.

Table 1: EtherCat Setup Files

File Types	Filenames	Test Cell Directories
EtherCAT drivers (EtherLab)	`ec_gerneric.ko` `ec_master.ko`	`/lib/modules/$(uname -r)/kernel/devices/`
EtherCAT utility	`ethercat`	`/opt/etherlab/bin`
Configuration	`ethercat`	`/etc/sysconfig/`

CyFlex tasks	`ecat_mon`
	`ecat_srvr`
	`ai_ecat`
	`di_ecat`
	`ao_ecat`
	`do_ecat`
		`/cyflex/bin/`

Specification (example files for CyFlex tasks)	See filenames above.	`/cell.def`

Specification (device attributes for drivers)	`ai_attr.ecat`
	`di_attr.ecat`
	`ao_attr.ecat`
	`do_attr.ecat`
		`/cell`

Go-script	`go.scp`	`/cell`

3 Installing EtherCat

Execute the following steps to install the EtherCat library:

1. Download and install the EtherCat RPM.
2. Revise the task sequence in the `go` script.
3. Assign a network interface.

Refer to the following sub-sections for details.

3.1 Downloading and Installing the RPM

Enter the following to download and install the RPM:

```
sudo yum install cyflex-ethercat
```

Note:

If a new kernel is installed, enter the following to reinstall the rpm to ensure the kernel calls are compatible.

```
sudo yum reinstall cyflex-ethercat
```

3.2 Task Sequence in the Go Script

When adding CyFlex task programs to the Go-script:

1. Ensure the `ecat_srvr` task is started before running device attributes (`device_attr`).
2. Run `ecat_mon` for intent to store values with a simpler setup. Refer to *Section 5 Specification File Example* on page 11.
3. Run `ai_attr` for intent to use calibration tables. Refer to *Section 6 Creating Device Attributes* on page 13.

3.3 Assigning an EtherCat Network Interface

If this the first setup of the EtherCAT driver, assign a network communication interface for EtherCAT. Disregard this step if already done on the system.

Execute the following steps for first use:

1. Edit the EtherCAT configuration file to identify the interface dedicated to EtherCAT:
   a. Open the file:
      ```
      sudo nedit /etc/sysconfig/ethercat
      ```
   b. Specify the MAC address (hexadecimal with colons) of each master. Refer to *Section 2.2 Communication Parameters* on page 2 for the MAC address.
      The `MASTER<X>_DEVICE` variables also determine how many masters are created. For example: A “non-empty” variable `MASTER0_DEVICE` creates one master. Adding a “non-empty” variable `MASTER1_DEVICE` creates a second master, and so on.

Example: `MASTER0_DEVICE="8C:89:A5:2F:CC:6E"`

Note:

The broadcast address `ff:ff:ff:ff:ff:ff` has a special meaning. It tells the master to accept the first device offered by any Ethernet driver.

2. Start the EtherCAT master (service):
   ```
   sudo service ethercat start
   ```
3. Update the `ifcfg` file to enable the port. These files are located in `/etc/sysconfig/network-scripts` . Determine the name of the `ifcfg` file associated with the port, which will vary depending on whether on a 32 or 64-bit system. Below is an example content for the file. No IP address is required. It is recommended to avoid use of an IP address since doing so could cause a routing conflict.
   ```
   DEVICE=eth2
   BOOTPROTO=none
   ONBOOT=yes
   TYPE=Ethernet
   HWADDR=68:05:CA:B7:D1:DE
   ```

4 Determining Connected Devices

Display devices connected to the EtherCAT interface/network by issuing the following command at a test cell terminal window:

```
/opt/etherlab/bin/ethercat cstruct
```

Note:

The `test.commander` Windows software also shows devices connected to the network. Refer to *Section 7 Test.Commander Settings* on page 21 for supplemental information.

The command output shows the possible channels, the slave interfaces that are used to set up the spec files or device attributes. Comments have been added to the output example below to include additions descriptions with a #. Contact SGS for assistance in understanding the command output.

Example Output:

```
#For the spec/attribute file, the main 3  things from this output we need is
# the master, slave, and channel number

#This indicates it is talking with master 0, and it is Slave 0. This also
# says what the device is,which is a Beckoff “EK1101” module

/* Master 0, Slave 0, “EK1101”

* Product code: 0x044d2c52
* Revision number: 0x00110000
*/
ec_pdo_entry_info_t slave_0_pdo_entries[] = {       #PDO information, we will
cover this further

#down with the gantner module

{0x6000, 0x01, 16}, /* ID */
};
ec_pdo_info_t slave_0_pdos[] = {
{0x1a00, 1, slave_0_pdo_entries + 0}, /* ID */
};
ec_sync_info_t slave_0_syncs[] = {
{0, EC_DIR_INPUT, 1, slave_0_pdos + 0, EC_WD_DISABLE},
{0xff}
};
/* Master 0, Slave 1, "EL1034"
#Next slave, which is a Beckoff “EL1034”
* Vendor ID: 0x00000002
* Product code: 0x040a3052
* Revision number: 0x00110000
*/
ec_pdo_entry_info_t slave_1_pdo_entries[] = {
{0x6000, 0x01, 1}, /* Input */
{0x6010, 0x01, 1}, /* Input */
{0x6020, 0x01, 1}, /* Input */
{0x6030, 0x01, 1}, /* Input */
};
```

```
ec_pdo_info_t slave_1_pdos[] = {
{0x1a00, 1, slave_1_pdo_entries + 0}, /* Channel 1 */
{0x1a01, 1, slave_1_pdo_entries + 1}, /* Channel 2 */
{0x1a02, 1, slave_1_pdo_entries + 2}, /* Channel 3 */
{0x1a03, 1, slave_1_pdo_entries + 3}, /* Channel 4 */
};
ec_sync_info_t slave_1_syncs[] = {
{0, EC_DIR_INPUT, 4, slave_1_pdos + 0, EC_WD_DISABLE},
{0xff}
};
/* Master 0, Slave 2, "Q.station 101 D"
#Next slave, which is a gantner “Q.station”
* Vendor ID: 0x0000050a
* Product code: 0x0002ab9b
* Revision number: 0x00030000
*/
ec_pdo_entry_info_t slave_2_pdo_entries[] = {
#each of the below lines is a channel that can be communicated with. The
mapping to the
#channels with the spec file is a little further down
{0x7000, 0x01, 32},
{0x7001, 0x01, 32},
{0x7002, 0x01, 32},
{0x7003, 0x01, 32},
{0x7004, 0x01, 32},
{0x7005, 0x01, 32},
{0x7006, 0x01, 32},
{0x7007, 0x01, 32},
{0x7008, 0x01, 32},
{0x7009, 0x01, 32},
{0x700a, 0x01, 32},
{0x700b, 0x01, 32},
{0x700c, 0x01, 32},
{0x700d, 0x01, 32},
{0x700e, 0x01, 32},
{0x700f, 0x01, 32},
{0x7010, 0x01, 32},
{0x7011, 0x01, 32},
{0x7012, 0x01, 32},
{0x7013, 0x01, 32},
{0x7014, 0x01, 32},
{0x7015, 0x01, 32},
{0x7016, 0x01, 32},
{0x6000, 0x01, 64},
{0x6001, 0x01, 32},
{0x6002, 0x01, 32},
{0x6003, 0x01, 32},
{0x6004, 0x01, 32},
{0x6005, 0x01, 32},
{0x6006, 0x01, 32},
{0x6007, 0x01, 32},
{0x6008, 0x01, 32},
{0x6009, 0x01, 32},
{0x600a, 0x01, 32},
{0x600b, 0x01, 32},
```

```
{0x600c, 0x01, 32},
{0x600d, 0x01, 32},
{0x600e, 0x01, 32},
{0x600f, 0x01, 32},
{0x6010, 0x01, 32},
{0x6011, 0x01, 32},
{0x6012, 0x01, 32},
{0x6013, 0x01, 32},
{0x6014, 0x01, 32},
{0x6015, 0x01, 32},
{0x6016, 0x01, 32},
{0x6017, 0x01, 32},
{0x6018, 0x01, 32},
{0x6019, 0x01, 32},
{0x601a, 0x01, 32},
{0x601b, 0x01, 32},
{0x601c, 0x01, 32},
{0x601d, 0x01, 32},
{0x601e, 0x01, 32},
{0x601f, 0x01, 32},
{0x6020, 0x01, 32},
{0x6021, 0x01, 32},
{0x6022, 0x01, 32},
{0x6023, 0x01, 32},
{0x6024, 0x01, 32},
{0x6025, 0x01, 32},
{0x6026, 0x01, 32},
{0x6027, 0x01, 32},
{0x6028, 0x01, 32},
{0x6029, 0x01, 32},
{0x602a, 0x01, 32},
{0x602b, 0x01, 32},
{0x602c, 0x01, 32},
{0x602d, 0x01, 32},
{0x602e, 0x01, 32},
{0x602f, 0x01, 32},
{0x6030, 0x01, 32},
{0x6031, 0x01, 32},
{0x6032, 0x01, 32},
{0x6033, 0x01, 32},
{0x6034, 0x01, 32},
{0x6035, 0x01, 32},
{0x6036, 0x01, 32},
{0x6037, 0x01, 32},
{0x6038, 0x01, 32},
{0x6039, 0x01, 32},
{0x603a, 0x01, 32},
{0x603b, 0x01, 32},
{0x603c, 0x01, 32},
{0x603d, 0x01, 32},
};
```

```
ec_pdo_info_t slave_2_pdos[] = {
#Here is where it is easiest to convert to device attributes/specfiles. The
number right after the
#‘+’ for this device is the channel number we would put in the spec/attribute
file.

{0x1600, 1, slave_2_pdo_entries + 0},
{0x1601, 1, slave_2_pdo_entries + 1},
{0x1602, 1, slave_2_pdo_entries + 2},
{0x1603, 1, slave_2_pdo_entries + 3},
{0x1604, 1, slave_2_pdo_entries + 4},
{0x1605, 1, slave_2_pdo_entries + 5},
{0x1606, 1, slave_2_pdo_entries + 6},
{0x1607, 1, slave_2_pdo_entries + 7},
{0x1608, 1, slave_2_pdo_entries + 8},
{0x1609, 1, slave_2_pdo_entries + 9},
{0x160a, 1, slave_2_pdo_entries + 10},
{0x160b, 1, slave_2_pdo_entries + 11},
{0x160c, 1, slave_2_pdo_entries + 12},
{0x160d, 1, slave_2_pdo_entries + 13},
{0x160e, 1, slave_2_pdo_entries + 14},
{0x160f, 1, slave_2_pdo_entries + 15},
{0x1610, 1, slave_2_pdo_entries + 16},
{0x1611, 1, slave_2_pdo_entries + 17},
{0x1612, 1, slave_2_pdo_entries + 18},
{0x1613, 1, slave_2_pdo_entries + 19},
{0x1614, 1, slave_2_pdo_entries + 20},
{0x1615, 1, slave_2_pdo_entries + 21},
{0x1616, 1, slave_2_pdo_entries + 22},
{0x1a00, 1, slave_2_pdo_entries + 23},
{0x1a01, 1, slave_2_pdo_entries + 24},
{0x1a02, 1, slave_2_pdo_entries + 25},
{0x1a03, 1, slave_2_pdo_entries + 26},
{0x1a04, 1, slave_2_pdo_entries + 27},
{0x1a05, 1, slave_2_pdo_entries + 28},
{0x1a06, 1, slave_2_pdo_entries + 29},
{0x1a07, 1, slave_2_pdo_entries + 30},
{0x1a08, 1, slave_2_pdo_entries + 31},
{0x1a09, 1, slave_2_pdo_entries + 32},
{0x1a0a, 1, slave_2_pdo_entries + 33},
{0x1a0b, 1, slave_2_pdo_entries + 34},
{0x1a0c, 1, slave_2_pdo_entries + 35},
{0x1a0d, 1, slave_2_pdo_entries + 36},
{0x1a0e, 1, slave_2_pdo_entries + 37},
{0x1a0f, 1, slave_2_pdo_entries + 38},
{0x1a10, 1, slave_2_pdo_entries + 39},
{0x1a11, 1, slave_2_pdo_entries + 40},
{0x1a12, 1, slave_2_pdo_entries + 41},
{0x1a13, 1, slave_2_pdo_entries + 42},
{0x1a14, 1, slave_2_pdo_entries + 43},
{0x1a15, 1, slave_2_pdo_entries + 44},
{0x1a16, 1, slave_2_pdo_entries + 45},
{0x1a17, 1, slave_2_pdo_entries + 46},
{0x1a18, 1, slave_2_pdo_entries + 47},
{0x1a19, 1, slave_2_pdo_entries + 48},
```

```
{0x1a1a, 1, slave_2_pdo_entries + 49},
{0x1a1b, 1, slave_2_pdo_entries + 50},
{0x1a1c, 1, slave_2_pdo_entries + 51},
{0x1a1d, 1, slave_2_pdo_entries + 52},
{0x1a1e, 1, slave_2_pdo_entries + 53},
{0x1a1f, 1, slave_2_pdo_entries + 54},
{0x1a20, 1, slave_2_pdo_entries + 55},
{0x1a21, 1, slave_2_pdo_entries + 56},
{0x1a22, 1, slave_2_pdo_entries + 57},
{0x1a23, 1, slave_2_pdo_entries + 58},
{0x1a24, 1, slave_2_pdo_entries + 59},
{0x1a25, 1, slave_2_pdo_entries + 60},
{0x1a26, 1, slave_2_pdo_entries + 61},
{0x1a27, 1, slave_2_pdo_entries + 62},
{0x1a28, 1, slave_2_pdo_entries + 63},
{0x1a29, 1, slave_2_pdo_entries + 64},
{0x1a2a, 1, slave_2_pdo_entries + 65},
{0x1a2b, 1, slave_2_pdo_entries + 66},
{0x1a2c, 1, slave_2_pdo_entries + 67},
{0x1a2d, 1, slave_2_pdo_entries + 68},
{0x1a2e, 1, slave_2_pdo_entries + 69},
{0x1a2f, 1, slave_2_pdo_entries + 70},
{0x1a30, 1, slave_2_pdo_entries + 71},
{0x1a31, 1, slave_2_pdo_entries + 72},
{0x1a32, 1, slave_2_pdo_entries + 73},
{0x1a33, 1, slave_2_pdo_entries + 74},
{0x1a34, 1, slave_2_pdo_entries + 75},
{0x1a35, 1, slave_2_pdo_entries + 76},
{0x1a36, 1, slave_2_pdo_entries + 77},
{0x1a37, 1, slave_2_pdo_entries + 78},
{0x1a38, 1, slave_2_pdo_entries + 79},
{0x1a39, 1, slave_2_pdo_entries + 80},
{0x1a3a, 1, slave_2_pdo_entries + 81},
{0x1a3b, 1, slave_2_pdo_entries + 82},
{0x1a3c, 1, slave_2_pdo_entries + 83},
{0x1a3d, 1, slave_2_pdo_entries + 84},
};
#The section below is how you would determine what channels are input/output.
From the
#section above we can see that there are 85 channels on this slave (starts
with 0).
#Now below we can see from the line:
#{2, EC_DIR_OUTPUT, 23, slave_2_pdos + 0, EC_WD_DISABLE},
#that the output channels start at channel 0, and that there are 23 of them.
Then the line
#below that we can see that there are 62 input channels, and that they start
at index 23
ec_sync_info_t slave_2_syncs[] = {
{0, EC_DIR_OUTPUT, 0, NULL, EC_WD_DISABLE},
{1, EC_DIR_INPUT, 0, NULL, EC_WD_DISABLE},
{2, EC_DIR_OUTPUT, 23, slave_2_pdos + 0, EC_WD_DISABLE},
{3, EC_DIR_INPUT, 62, slave_2_pdos + 23, EC_WD_DISABLE},
{0xff}
};
```

5 Specification File Example

In the example specification file below, the `ecat_mon` task interprets EtherCAT channels into REAL CyFlex variables.

The mapping uses master index; (currently only one master is supported), slave position, and slave channel. `ecat_mon` is a quicker and easier setup to communicate with CyFlex but lacks the ability to use calibration tables. For that ability, use the device attribute portion of the driver. Refer to Section 6 Creating Device Attributes on page 13.

Note:

The registered name `EcatMon` (below) is an instance of the program, `ecat_mon`. The CyFlex convention uses a unique name for each instance of a task. In order to run another instance of `ecat_mon`, define another registered name for that instance.

Important:

`ecat_mon` and the device attributes setup using any of the attributes tasks (`ai_attr`, `ao_attr`, `di_attr`, `do_attr`) CANNOT be used together. Select one or the other. This can be changed later.

The following specification file example is used with the output from *Section 4 Determining Connected Devices* on page 6.

```
# Example spec file for EtherCAT monitor task
#
# History:
# clf - 11/2014 - initial version
#
# Registered Name
@Reg_Name
EcatMon
$Debug
yes
#
$InputVariables
# Currently only input variables are supported, tab or comma separated.
# Variable   CyFlex    Device Units    Update     Master    Slave     Slave
# Name       Units     OR Event Name   Rate [ms]  Index     Position  Channel
Q1timestamp  usec      usec            1000       0         2         23
EBDI0        none      none            1000       0         1         0
EBDI1        none      none            1000       0         1         1
EBDI2        none      none            1000       0         1         2
EBDI3        none      none            1000       0         1         3
QDI00        none      none            1000       0         2         2
QDO00i       none      none            1000       0         2         3
QDI01        none      none            1000       0         2         4
QDO01i       none      none            1000       0         2         5
QSAI17_T     deg_F     deg_C           1000       0         2         42
QSAI21_T     deg_F     deg_C           1000       0         2         46
$
$OutputVariables
QDO00o       none      none            1000       0         2         0
QDO01o       none      none            1000       0         2         1
QDO2         none      none            1000       0         2         3
```

```
#Q1DO0       none      none            1000       0         2         0
#Q1DO1       none      none            1000       0         2         1
$
# List MUST end with a '$'
```

6 Creating Device Attributes

Create device attribute specification files for the EtherCAT drivers based on the devices connected to the EtherCAT network.

Create the device attribute specification files for the EtherCAT drivers based on the devices connected to the network. Refer to the output example in *Section 4 Determining Connected Devices* on page 6.

The following attribute files are examples:

- Analog input (`ai_attr.ecat`)
- Digital input (`di_attr.ecat`)
- Analog output (`ao_attr.ecat`)
- Digital output (`do_attr.ecat`)

6.1 Analog Input

The following is an attribute example for `ai_attr.ecat`.

```
# Example:
# ai_attr ai_attr.408 &
#
# key_name
ECATai

# error_name
ECATaierr

# driver_name      driver_priority
ai_ecat          24

# handler name     handler priority
inpt_handler            19

# chan_offset
 0

# num_boards
# number of EtherCAT masters  - currently only 1 supported
   1

# board_address     interrupt_number    dma_channel
# master index
   0

# pacer_clock_freq (Hz)
 1000

# time_out(sec)
 10
```

```
# conversion_time(micro-sec.)   actually time from channel load to
# conversion signal, must be longer than
# the longest required settling time of the attached multiplexers
 0

# num_intervals
# 4
3
# intervals (milli-seconds) ( one per line ) - (Hz = 20,10,5,4,2,1)
#1
10
100
1000

# num_crit_intervals
1

# crit_intervals (micro-seconds) (one per line)
100000

# max_num_chan  max supported is 2048 total input and output EtherCat
channels
  100

# Channel Definitions
# unused channels may be skipped
# channel numbers, master indexes, slave position and channel start
from 0
# largest possible channel number is (max_num_chan - 1)
# use dash (-) for unsupported features
#
#  use /opt/etherlab/bin/ethercat cstruct or ethercat xml to
# understand channel number mapping
#
# chan num  master index  slave pos  channel    fixed gain  fixed
# offset
#  0-i        0           0-n         0-m          1.0       0.0
  0         0             3         24             1.0       0.0
  1         0             3         25             1.0       0.0
  2         0             3         26             1.0       0.0
  3         0             3         27             1.0       0.0
  4         0             3         32             1.0       0.0
  5         0             3         33             1.0       0.0
  6         0             3         34             1.0       0.0
  7         0             3         35             1.0       0.0
  8         0             3         36             1.0       0.0
  9         0             3         37             1.0       0.0
 10         0             3         38             1.0       0.0
 11         0             3         39             1.0       0.0
```

```
 12          0               3          40              1.0         0.0
 13          0               3          41              1.0         0.0
 14          0               3          42              1.0         0.0
 15          0               3          43              1.0         0.0
 16          0               3          44              1.0         0.0
 17          0               3          45              1.0         0.0
 18          0               3          46              1.0         0.0
 19          0               3          47              1.0         0.0
 20          0               3          48              1.0         0.0
 21          0               3          49              1.0         0.0
 22          0               3          50              1.0         0.0
 23          0               3          51              1.0         0.0
 24          0               3          52              1.0         0.0
 25          0               3          53              1.0         0.0
 26          0               3          54              1.0         0.0
 27          0               3          55              1.0         0.0
 28          0               3          56              1.0         0.0
 29          0               3          57              1.0         0.0
 30          0               3          58              1.0         0.0
 31          0               3          59              1.0         0.0
 32          0               3          60              1.0         0.0
 33          0               3          61              1.0         0.0
 34          0               3          62              1.0         0.0
 35          0               3          63              1.0         0.0
 36          0               3          64              1.0         0.0
 37          0               3          65              1.0         0.0
 38          0               3          66              1.0         0.0
 39          0               3          67              1.0         0.0
  -          -               -           -              -
# NOTE: last line of channel definitions must be all dashes.

# num_prog_gains
 0

# programable gains
# 1.0
# 2.0
# 4.0
# 8.0

# min_count  max_count  zero_count
   -10000       10000       0

# max_volt_in  min_volt_in
   -10000.0     -10000.0

# ext_trigger_allow ext_clock_allow ext_enable_allow ssh_allow
# num_ssh_chan
      0               0               0              0
      0
```

```
# pre_trigger_allow  post_trigger_allow  pre_trig_samples
analog_trig_allow
        0               0                   0
        0

# num_trig_modes
  0

# trigger modes - since 0 none needed
#

# num_trig_chan
  0

# trigger channels - since 0 none needed      status = get_next_line(
f1, line );
#
# relative sample position flag &
# raw data format (0 ULONG, 1 LONG, 2 short, 3 ushort, 4 float)
 0     4
```

6.2 Digital Input

The following is an attribute example for `di_attr.ecat`.

```
#
# key_name
ECATdi

# device error event name
ECATerr

# driver name  task priority
di_ecat            20

# handler name    task priority
inpt_handler      19

# chan_offset
  128

# num_boards
# number of EtherCAT masters
   1

# board_address   interrupt_number  dma_channel  # master index
   0
```

```
# pacer_clock_freq - maximum update rate
 100

# time out in seconds
 10

# num_intervals
 3

# intervals (milli-seconds) ( one per line ) - (Hz = 8,4,2,1)
# 100 Hz is maximum for EBLOX modules
 10
 100
 1000

# num_crit_intervals
 1

# crit_intervals (micro-seconds) (one per line)
 10000

# max_num_chan
  24

# Channel Definitions:
# unused channels may be skipped
# channel numbers, master indexes, slave position and channel start
# from 0
# largest possible channel number is (max_num_chan - 1)
# use dash (-) for unsupported features
#
# use /opt/etherlab/bin/ethercat cstruct or ethercat xml to understand
# channel number mapping
#
# chan num  master index  slave pos  channel  fixed gain  fixed offset
# 0-i        0           0-n         0-m          1.0         0.0
  0         0             3          28           1.0         0.0
  1         0             3          29           1.0         0.0
  2        0     3          69    1.0   0.0
  3        0     3          70    -     -
  4        0     3          71    -     -
  5        0     3          72    -     -
  6        0     3          73    -     -
  7        0     3          74    -     -
  8        0     3          75    -     -
  9        0     3          76    -     -
 10        0     3          77    -     -
 11        0     3          78    -     -
 12        0     3          79    -     -
 13        0     3          80    -     -
```

```
 14        0     3          81    -     -
 15        0     3          82    -     -
 16        0     3          83    -     -
 17        0     3          84    -     -
  -         -              -          -           -
# NOTE: last line of channel definitions must be all dashes.
```

6.3 Analog Output

The following is an attribute example for `ao_attr.ecat`.

```
#
# NOTE:  ecat_srvr task must be running BEFORE driver is run:
#
# key_name
 ECATao

# driver_name
 ao_ecat

# driver priority
 25

# chan_offset
 0

# num_boards
 1

# io_address  - repeat this entry num_boards times, not used by driver
# not really used by driver, since accessed through optomux_srvr task
# just put board index here for address
   0

# max_num_chan
 6

# Channel Definitions:
# channel numbers, board indexes, and board chan start from 0
#
# largest possible channel number is (max_num_chan - 1)
# use dash (-) for unsupported features
#
# The mux chan is the module address.
#
# NOTE:  It is possible to mix AI, AO, DI, DO on a given board.
#
```

```
# The following are the analog output channel types supported by the
eblox driver.
#
# Valid types:
# A9-1 Voltage          2110
# A9-1 Current          2111
# D1-1 PWM             2142
#
# The type code is just a combination of the module type and the
sensor type
# from the Ganter manual.  (Module type * 10 + Sensor type)
#
# NOTE:  The A9-1 & D1-1 must be setup with the ecommander.
# The driver requires that the data type be set to REAL, and the
SOURCE range be from 0 to 1.0
# This is NOT the default.  You can set the output range to whatever
# you want:
# 0-5V, +/-5V, 0-10V, +/-10V, 0-20mA, 4-20mA, but the source range
# MUST be set to 0 for min and 1.0 for maximum.
#
# chan num  egate index  mod addr+  mod chan     gain  offset  type
#                       interface
# A9-1  module 11  on interface 1
#      master index    slave index     channel
  0         0              3              2        -     -        -
  1         0              3             19        -     -        -
  2         0              3             20        -     -        -
  3         0              3             21        -     -        -
  4         0              3             22        -     -        -
 -          -              -              -        -     -        -
# NOTE: last line of channel definitions must be all dashes
```

6.4 Digital Output

The following is an attribute example for `do_attr.ecat`.

```
#
# NOTE:  ecat_srvr task must be running BEFORE driver is run:
#
# key_name
 ECATdo

# driver_name
 do_ecat

# driver priority
 20

# chan_offset
 0
```

```
# initial_state  - initial state for all outputs
 0

# number of masters currently only 1
 1

# master
# kSNAPIO modules are accessed via the TCP/IP ports.
# board address in index into SNAPIO brain module addresses passed
# to snapio_srvr at startup, starting with index 0
# io_address(hex)
  0

# max_num_chan
 20

# Channel Definitions
# channel numbers, board indexes, and board chan start from 0
#
# largest possible channel number is (max_num_chan - 1)
# use dash (-) for unsupported features
#
# module address is mux_chan
#
# NOTE:  It is possible to mix DI and DO channels on a give module.
#        However, for effecient operation of the software, DOs should
# be grouped together.
#
# chan num  master index   slave index channel   fixed gain
  0         0              3           0              -
  1         0              3           1              -
  2         0              3           3              -
  3         0              3           4              -
  4         0              3           5              -
  5         0              3           6              -
  6         0              3           7              -
  7         0              3           8              -
  8         0              3           9              -
  9         0              3          10              -
 10         0              3          11              -
 11         0              3          12              -
 12         0              3          13              -
 13         0              3          14              -
 14         0              3          15              -
 15         0              3          16              -
 16         0              3          17              -
 17         0              3          18              -
 -          -              -           -              -
# NOTE: last line of channel definitions must be all dashes
```

7 Test.Commander Settings

The EtherCAT interface, controller (master) and modules (slaves) are configured from a Windows machine on the network, using a program from Gantner called `test.commander`.

Instructions for installing and using the program are available on the Gantner Web site and include a setup wizard. Refer to:

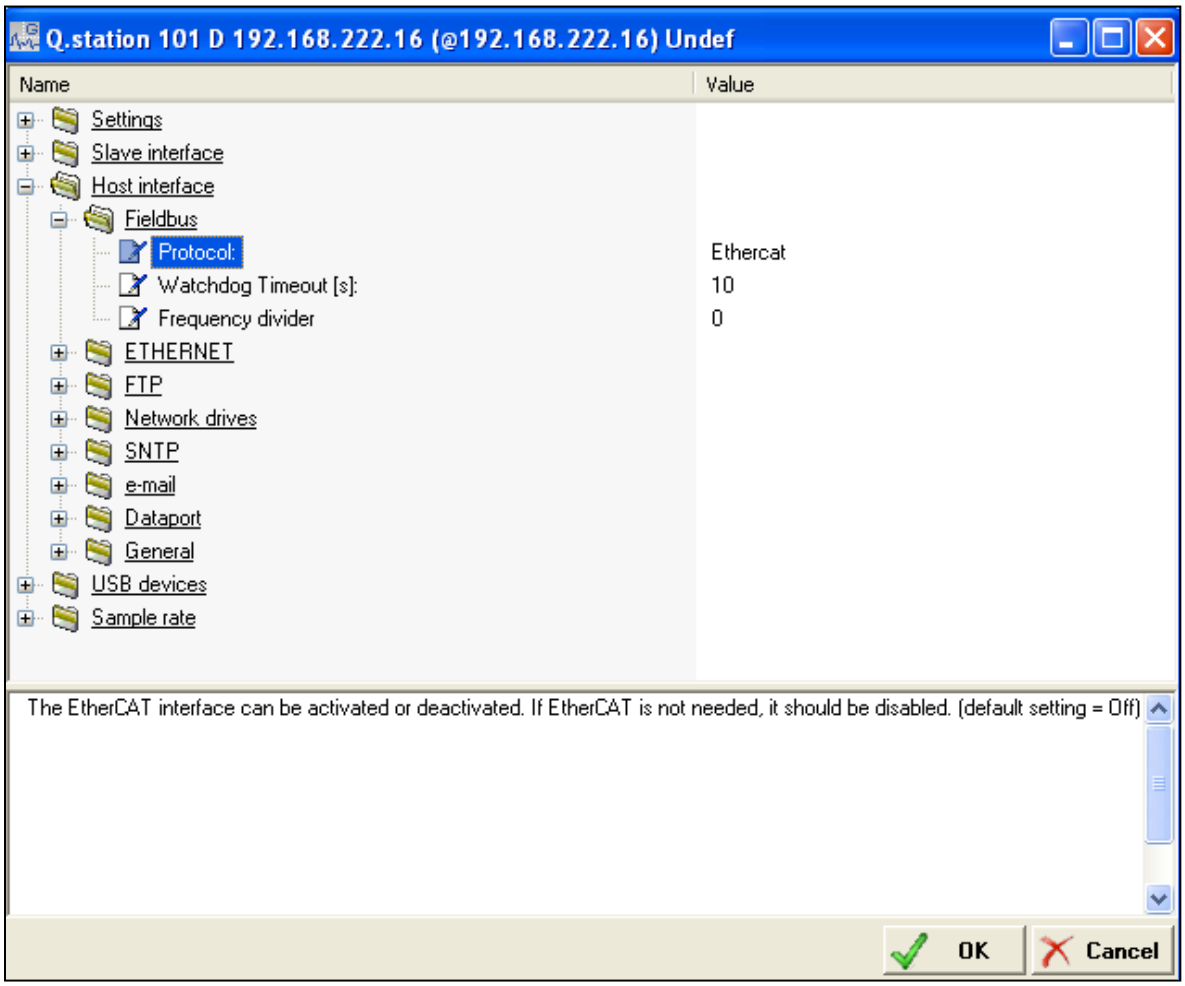
https://www.gantner-instruments.com/product-group/test-commander/

All other settings done via `test.commander` should be the same as for `eblox` UDP streaming driver. All channels from `q.station` are assumed to be float as for `eblox` UDP streaming driver.

7.1 Activating the EtherCAT Interface

Select **Host Interface** - **Fieldbus** - **Protocol** as in *Figure 1* to enable the EtherCat fieldbus.

Figure 1: Enable EtherCAT Fieldbus

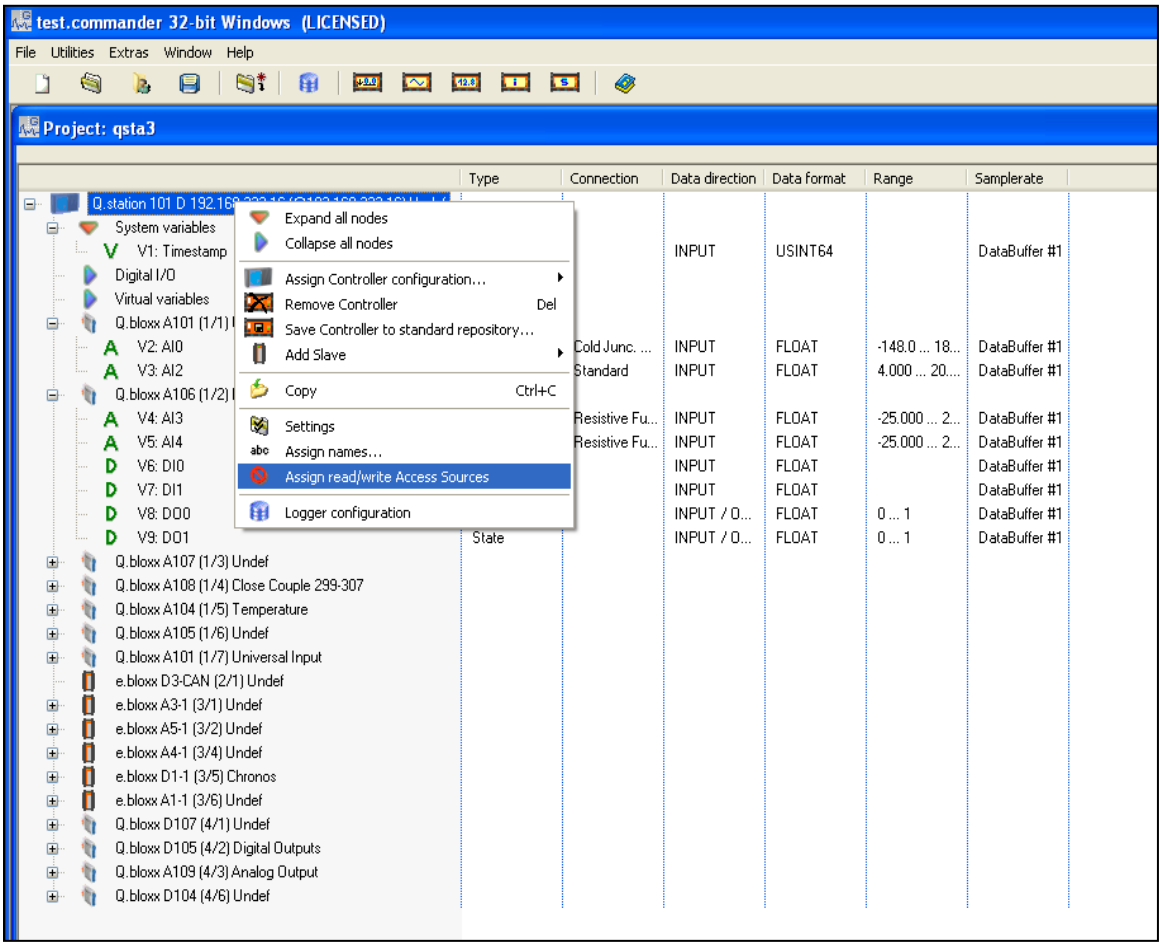


7.2 Assigning Read and Write Access

Execute the following steps to assign read and write access sources for the controller (master):

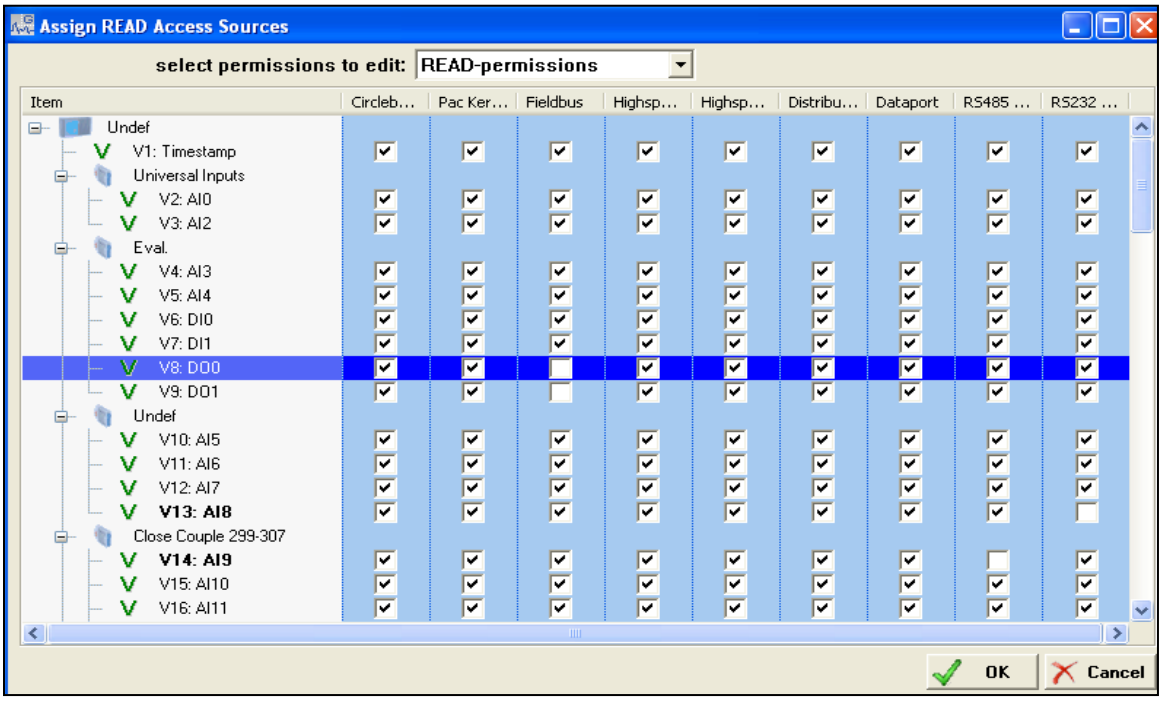
1. Right-click the controller to open menu options and then select Assign read/write Access Sources as in .
2. *Figure* 2.

Figure 2: Assign read/write Access Sources



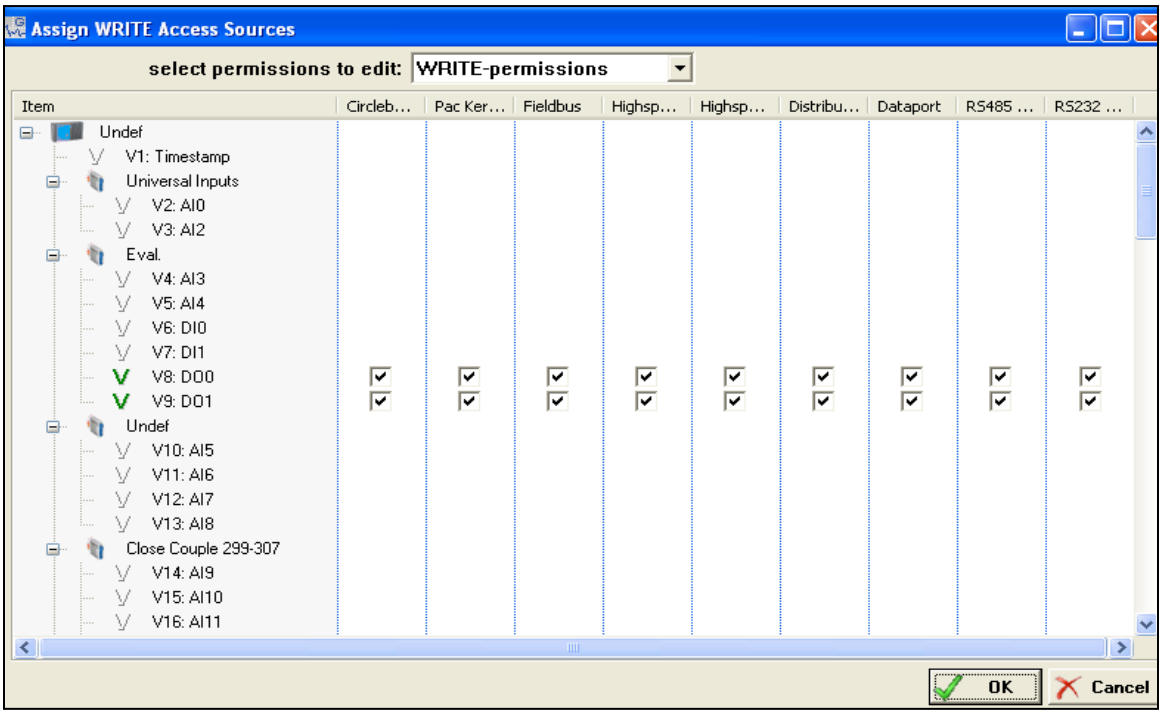
3. Set the read option for input channels on the fieldbus as in *Figure 3*.

Figure 3: Read Option for Fieldbus Input Channels



4. Set the write option for output channels on the fieldbus as in *Figure 4*.

Figure 4: Write Option for Fieldbus Output Channels



Appendix A. Probing and Configuring Beckhoff I/O Modules

Use the IgH EtherCAT `ethercat` command-line tool `graph` option to produce a file that can be processed by the `graphviz` graph visualization software into a viewable image showing the topology of the Beckhoff network.

First, install the following software:

1. IgH EtherCAT software; refer to *Installing the IgH EtherCAT Software* below
2. graphviz graph visualization software; refer to *Installing and Using the graphviz Graph* Visualization Software below

Installing the IgH EtherCAT Software

To probe and configure Beckhoff I/O modules, first install the IgH EtherCAT software. Refer to IgH EtherCat Master 1.5.2 Documentation, section *9 Installation*.

Installing and Using the graphviz Graph Visualization Software

Enter the following to download and install the graphviz package on a CyFlex node:

```
sudo yum install graphviz
```

Enter the following to view the multiple image format options:

```
dot -Txxx
```

Refer to the graphviz documentation for detailed usage instructions.

Producing a Graph

Enter the following to use the graphviz dot utility and produce a graph of the EtherCAT network in jpeg format:

```
/opt/etherlab/bin/etherlab graph | dot -Tjpeg > /tmp/my_graph.jpeg
```

Refer to the Test Cell 3 example located at the SGS CyberMetrix National Road facility.