DEWETRON

EtherCAT Slave Configuration 1.5

WELCOME TO THE WORLD OF DEWETRON!

Congratulations on your new device! It will supply you with accurate, complete and reproducible measurement results for your decision making.
Look forward to the easy handling and the flexible and modular use of your DEWETRON product and draw upon more than 25 years of DEWETRON expertise in measurement engineering.

ISO9001

EN ISO 14001

THE MEASURABLE DIFFERENCE.

The information contained in this document is subject to change without notice.

Technical Support

Please contact your local authorized DEWETRON representative first for any support and service questions.

For Asia and Europe, please contact:

DEWETRON GmbH
Parkring 4
8074 Grambach
AUSTRIA

Tel.: +43 316 3070
Fax: +43 316 307090
Email: support@dewetron.com
Web: http://www.dewetron.com

The telephone hotline is available Monday to Friday between 08:00 and 17:00 CET (GMT +1:00)

For America, please contact:

DEWETRON, Inc.
2850 South County Trail, Unit 1
East Greenwich, RI 02818
U.S.A.

Tel.: +1 401 284 3750
Toll-free: +1 877 431 5166
Fax: +1 401 284 3755
Email: us.support@dewetron.com
Web: http://www.dewetron.com

The telephone hotline is available Monday to Friday between 08:00 and 17:00 GST (GMT -5:00)

Restricted Rights Legend:

Printing History:

Before updating your software please contact DEWETRON. Use only original software from DEWETRON.

Please find further information at www.dewetron.com.

Table of content

1 PREFACE

This document describes the first configuration of the EtherCAT slave module on the DEWETRON device and in OXYGEN, an Oxygen system can provide timestamped periodic measurement values to an EtherCAT master. The most important control features as well as some status information are provided as well.

Before the installation is begun, there are three key elements, which are needed during the configuration and installation (See EtherCAT Configuration). The location of the following files should be noted.

2 EtherCAT Configuration

2.1 Selection of transferred channels

The names of all channels that are mapped into the cyclic input data of the EtherCAT frame have to be specified in a list file before starting Oxygen. This file can be created and modified using any text editor and must be located in the following path:
Windows: %PUBLIC%\Documents\Dewetron\Oxygen\ethercat_slave_channels.txt
UBUNTU: \home\dewetron\Dewetron\OXYGEN

- In this list the channels must be defined, which should be transmitted over EtherCAT. Of course, the channels in the list must exist in the current OXYGEN setup.
- The file has to contain a single Oxygen channel name per line. Empty lines can be used to group channels into different PDOs.
- At least one existing channel of the loaded OXYGEN Setup must be contained within the ethercat_slave_channels.txt file, otherwise the plugin will not be loaded correctly in OXYGEN.

It is necessary to use the long channel name in this file in order to uniquely reference all channels. These names are visible in Oxygen when switching to flat view using the icons on top of the channel lists. Additionally, a list containing the long names of all Oxygen channels is generated after every configuration change in the following location:
Windows: %TEMP%\ oxygen_all_channels.txt
UBUNTU: Ubuntu\tmp\ oxygen_all_channels.txt

- This list contains all channel names of the current OXYGEN setup with the long channel names. These names must be used in the ethercat_slave_channels.txt file.
- This list is generated after every configuration.

The maximum number of channels that can be transferred depends on the installed hardware; channels exceeding this limit are ignored.

2.2 ESI File

An ESI file representing the active configuration is generated during startup of the EtherCAT subsystem. To allow reliable and consistent offline configuration, its con- tent only depends on the specified channel configuration and does not consider which channels are available / valid in the active measurement setup.

This file can be found at the following location:
Windows: %PUBLIC%\Documents\Dewetron\Oxygen\oxygen_esi.xml
UBUNTU: \home\dewetron\Dewetron\OXYGEN

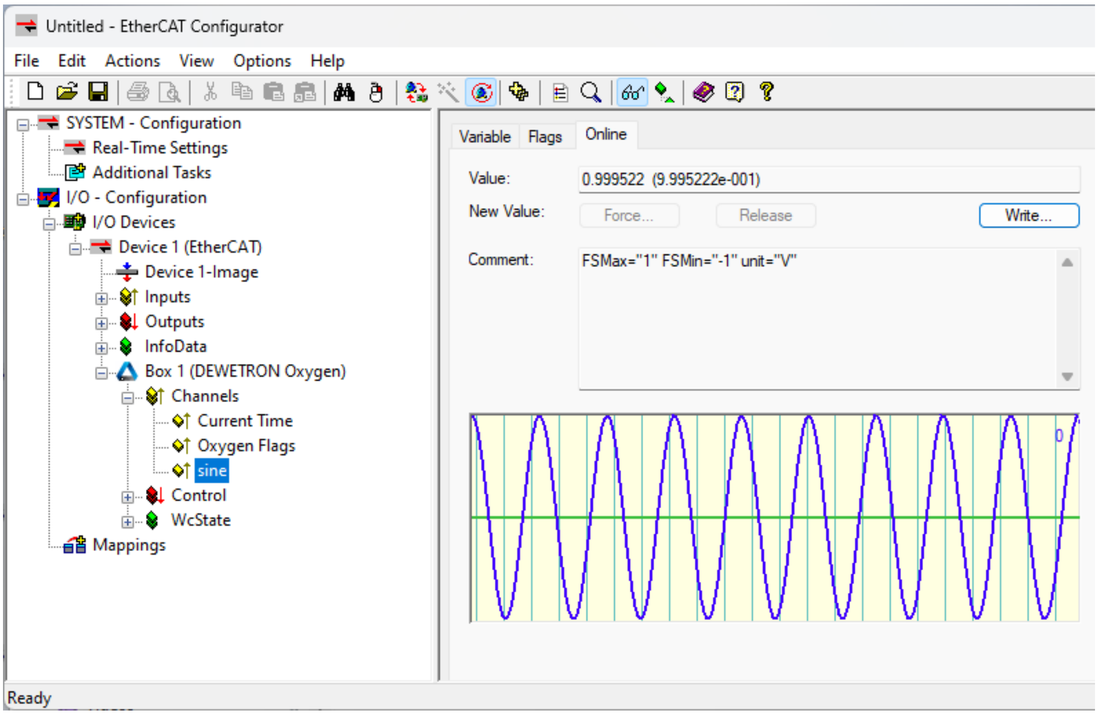
2.2.1 ESI File comments

Measurement range and unit information of transferred channels will be saved in *.esi-file as <Comment>.

- Informaiton ca be read by EtherCAT-Master
- Can be passed through to Beckhoff EtherCAT configuration tool for *.eni-file generation

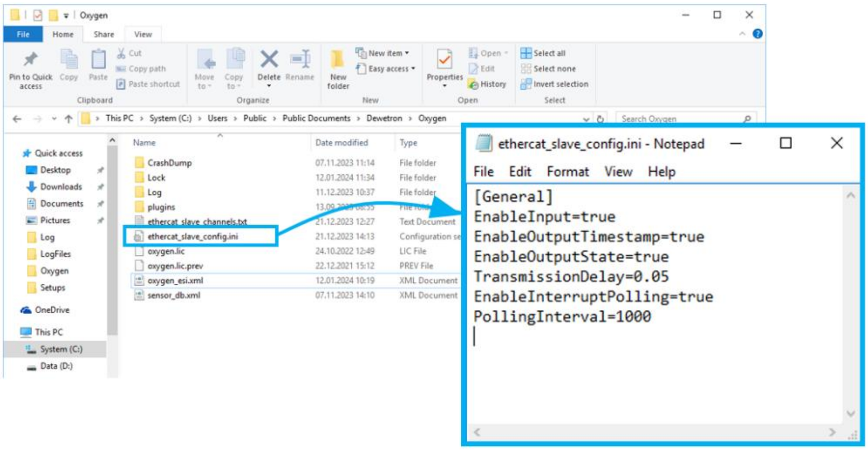
```xml
<Entry>
    <Index>#x3001</Index>
    <SubIndex>0</SubIndex>
    <BitLen>32</BitLen>
    <Name>sine</Name>
    <Comment>FSMax="1" FSMin="-1" unit="V"</Comment>
    <DataType>REAL</DataType>
</Entry>
```

Example in EtherCAT configurator.



2.3 EtherCAT Slave configuration file

A configuration file for the EtherCAT Slave configuration can be found in the following path:
Windows: %PUBLIC%\Documents\Dewetron\Oxygen\ ethercat_slave_config.ini



Following parameters are available and can be changed manually within the file. Note that changes only take effect after OXYGEN has been restarted.

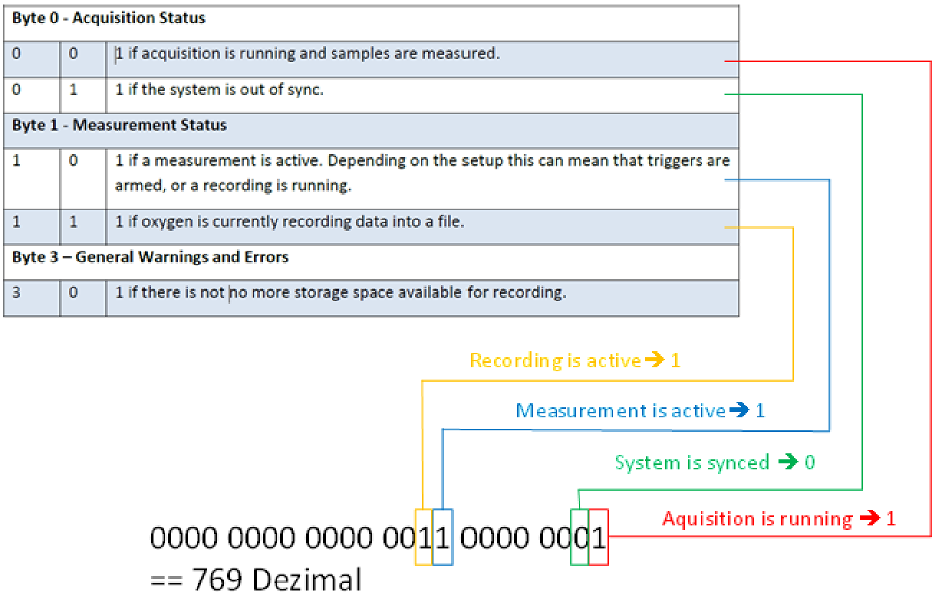
List and description of available configuration parameters:

- **EnableInput=true**
  Input variable “Control” will be enabled if true and disabled if false.
  The EtherCAT master can control Oxygen using several flags in the “Control” value in the output PDO. The effect of these control flags can be monitored using the corresponding Oxygen flags. The following flags are defined:

Byte	*Bit*	*Content*
1	0	Set to 1 while the measurement should be active.

- **EnableOutputTimestamp=true**
  The timestamp which will be transmitted with the variable “Current Time” will be transferred if true and will not be transferred if false. The absolute measurement time of the samples present in the buffer is communicated in the 8 byte field “Current Time”. It contains the time in nanoseconds since 0:00 of 1/1/1970 as an unsigned value. Each day is treated as being 8.64e+13 nanoseconds long, but once an acquisition is running the value is updated in the style of TAI (counting every elapsed second) to prevent any discontinuations caused by leap seconds.

- **EnableOutputState=true**
  The OXYGEN state which will be transmitted with the variable “Oxygen Flags” will be transferred if true and will not be transferred if false. Cyclic input data contains four bytes for flags (starting at offset 8) that indicate the current state of the Oxygen measurement. These values are not synchronized to the channel samples, therefore they always contain the most recent status and the timestamp does not apply!



Byte 0 - Acquisition Status		
0	0	1 if acquisition is running and samples are measured.
0	1	1 if the system is out of sync.
Byte 1 - Measurement Status		
1	0	1 if a measurement is active. Depending on the setup this can mean that triggers are armed, or a recording is running.
1	1	1 if oxygen is currently recording data into a file.
Byte 3 – General Warnings and Errors		
3	0	1 if there is not no more storage space available for recording.

- TransmissionDelay=0.5
  Data is sent after this delay (in seconds). Depending on the transmitted signals, this value can be adjusted. When using only analog input or fast math channels, this delay can be shortened. But if computations with large block sizes (e.g. FFT) or network transfers are involved in signal acquisition, a larger delay value may be required.

- EnableInterruptPolling=true
  When the EtherCAT hardware has interrupts enabled, data polling will match PollingInterval better. Note that this setting will increase the overall system CPU load significantly.

- PollingInterval=1000
  Oxygen will attempt to update data with this rate in [Hz]. Actual update rates may be slower.

3 Input Data

Cyclic input data delivers a synchronized snapshot of all values of the selected measured and calculated channels.
To determine the exact time of occurrence of the sample value, a timestamp is provided via PDO and valid for all samples contained.

Despite being timestamped exactly at the time of occurrence, as a result of blocked data processing and dependencies of software calculations, data samples are only ready with a varying delay in relation to the clock source. In order to guarantee that all data samples contained in the input PDO are in sync at any time an additional output delay is used over all channels.
This delay is configured at 500ms.

3.1 Channel Samples

Channel samples are always sent as 4-byte floating point values that have all necessary scaling applied, and in the order they are listed in the configuration file. The value of the first channel can be found at offset 12 in the input PDO; successive channels follow without any padding between them.

A channel can report the special value NaN (not-a-number; IEEE 754 NaNq) in the following cases:

- For calculated channel NaN is a legal value and may occur at any point
- The channel is configured for transfer but not available in the measurement report
- The channel is lagging behind and exceeds the maximum allowed delay

3.2 Timestamp Format

The absolute measurement time of the samples present in the buffer is communicated in the 8 byte field "Current Time".
It contains the time in nanoseconds since 0:00 of 1/1/1970 as an unsigned value. Each day is treated as being $8.64e^{+13}$ nanoseconds long, but once an acquisition is running the value is updated in the style of TAI (counting every elapsed second) to prevent any discontinuations caused by leap seconds.

3.3 Oxygen Flags

Cyclic input data contains four bytes for flags (starting at offset 8) that indicate the current state of the Oxygen measurement.
These values are not synchronized to the channel samples, therefore they always contain the most recent status and the timestamp does not apply!

Byte	*Bit*	*Content*
Byte 0 - Acquisition Status		
0	0	1 if acquisition is running and samples are measured.
0	1	1 if the system is out of sync.
Byte 1 - Measurement Status		
1	0	1 if a measurement is active. Depending on the setup this can mean that triggers are armed, or a recording is running.
1	1	1 if oxygen is currently recording data into a file.
Byte 3 – General Warnings and Errors		
3	0	1 if there is not no more storage space available for recording.

4 Measurement Setup Selection

Oxygen always starts as specified in the application settings, which can mean automatic loading of a setup. This is currently not communicated via EtherCAT.

Masters can switch the active Oxygen measurement setup using the SDO object "Setup" (address 0x6000; uint32) in PreOp state.

Writing a new SetupIndex to this object triggers loading of the corresponding setup from the following path:
Windows: <Oxygen data folder>/ECAT<SetupIndex>.dms

The initial value of the object is 0 which is reserved and may not be used as a SetupIndex.

EtherCAT state changes are only possible if the setup was successfully loaded and may be delayed (loading takes longer than expected) or even prevented (setup file not found or loading failed).

5 INSTALLATION

NOTE: The following steps can be ignored by UBUNTU users.
For UBUNTU "dewetron-ethercat_2.3.0~release-bookworm_amd64.deb" must be installed.
After installation of the Debian File, no further installation steps are necessary.

All the needed files can be downloaded or found here:

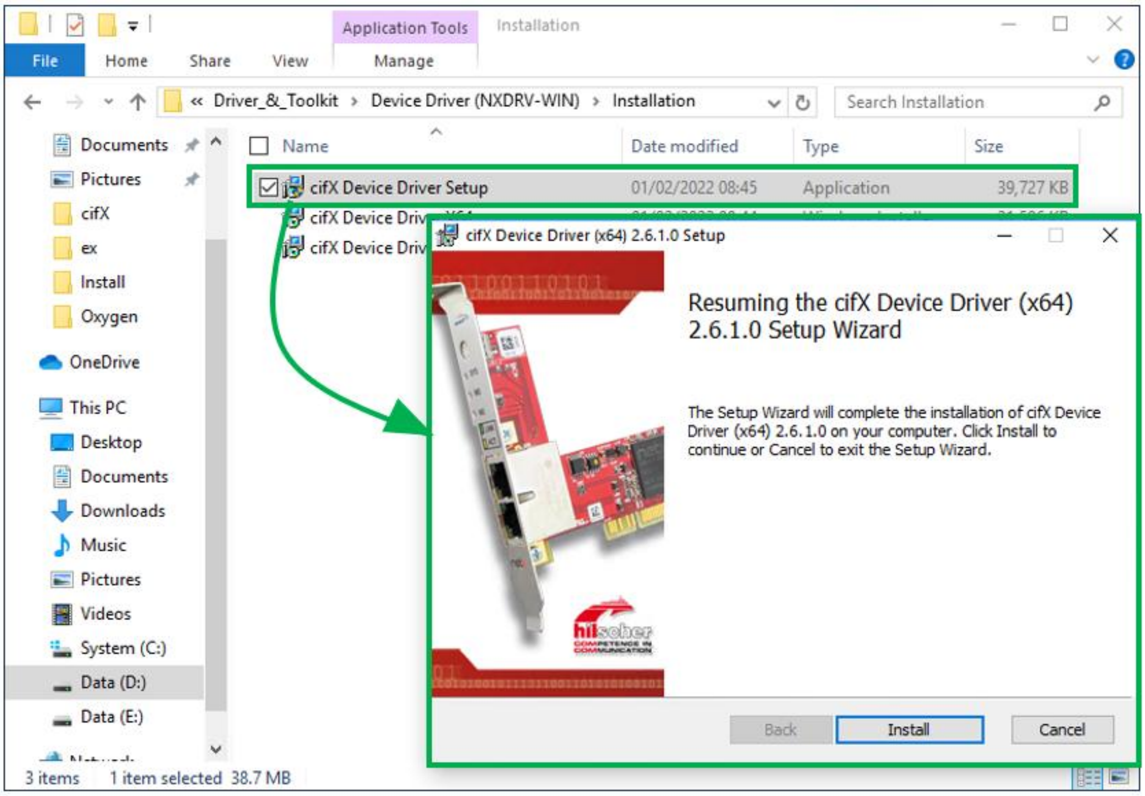
https://www.hilscher.com/fileadmin/user_upload/global/downloads/dvd/Communication_Solutions_2022-03-1.zip
Or
\\dewedata\Internal\01_Install\08_Software_Messtechnik\Hilscher\Communication_Solutions_90_2022-03-1.zip Must be on the VPN and have access to engineering directory.

5.1 INSTALLATION INSTRUCTIONS

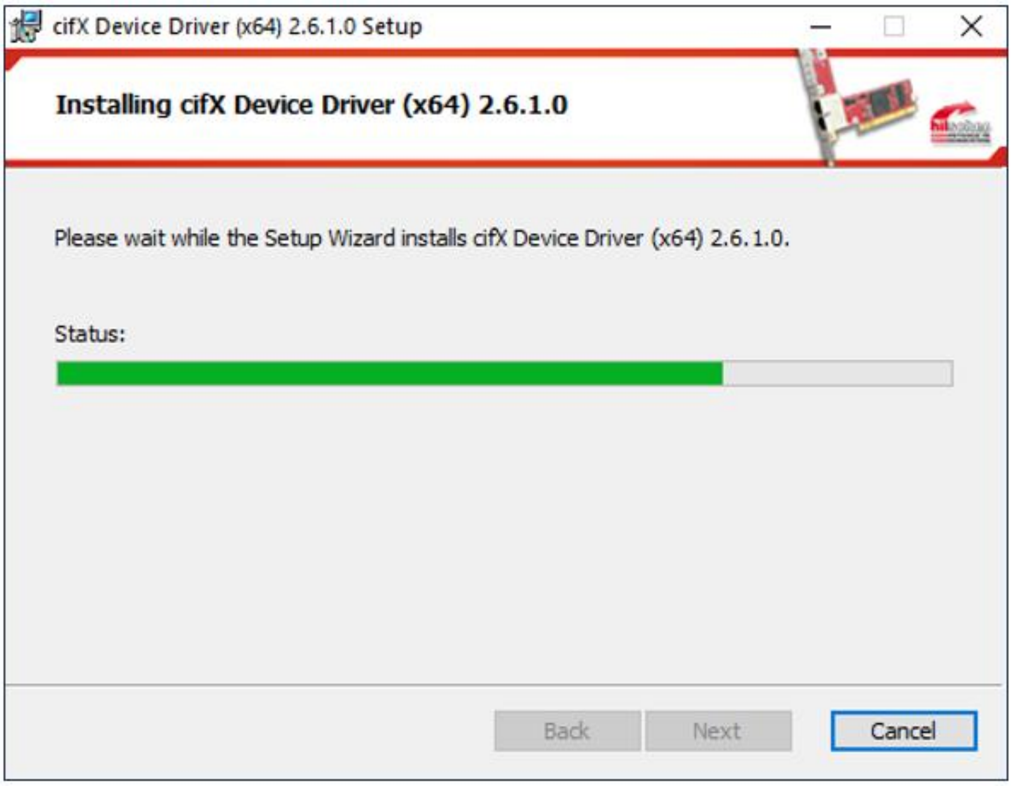
5.1.1 INSTALLATION HILSCHER DRIVER FOR TRION-ETHERCAT-S1

1. Extract the downloaded .zip file from Hilscher (Communication_Solutions_90_2022-03-1.zip)
2. Start the installation from the extracted files

Start the cifX device driver setup: …..\Driver_&_Toolkit\Device Driver (NXDRV-WIN)\Installation\cifX Device Driver Setup.exe. Press the button "Install" to start the automatic installation of the cifX driver setup.



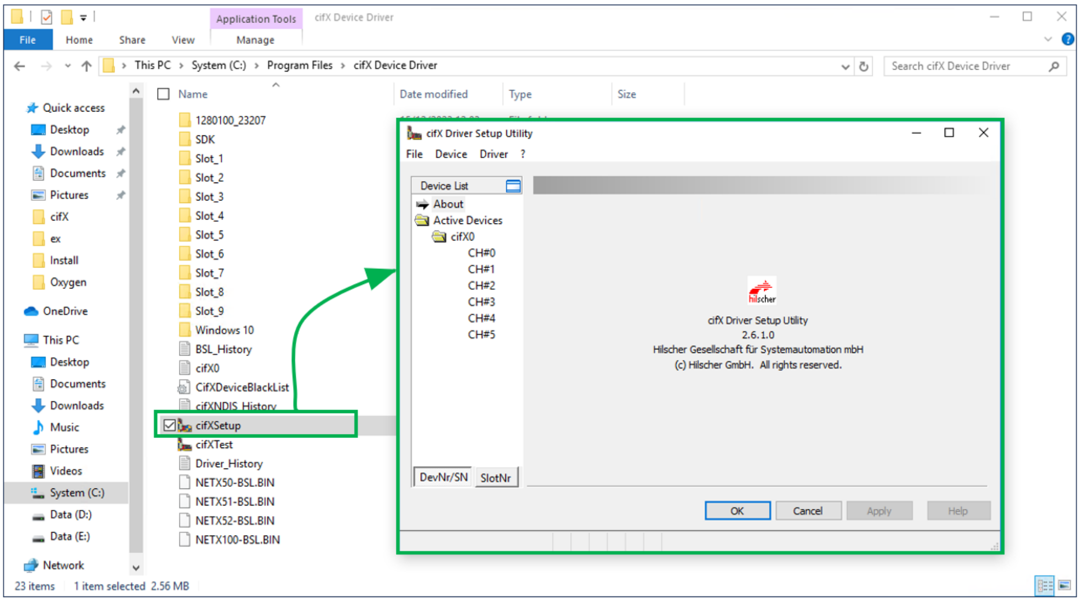
3. Wait until the installation of the device driver for cifX is finished.



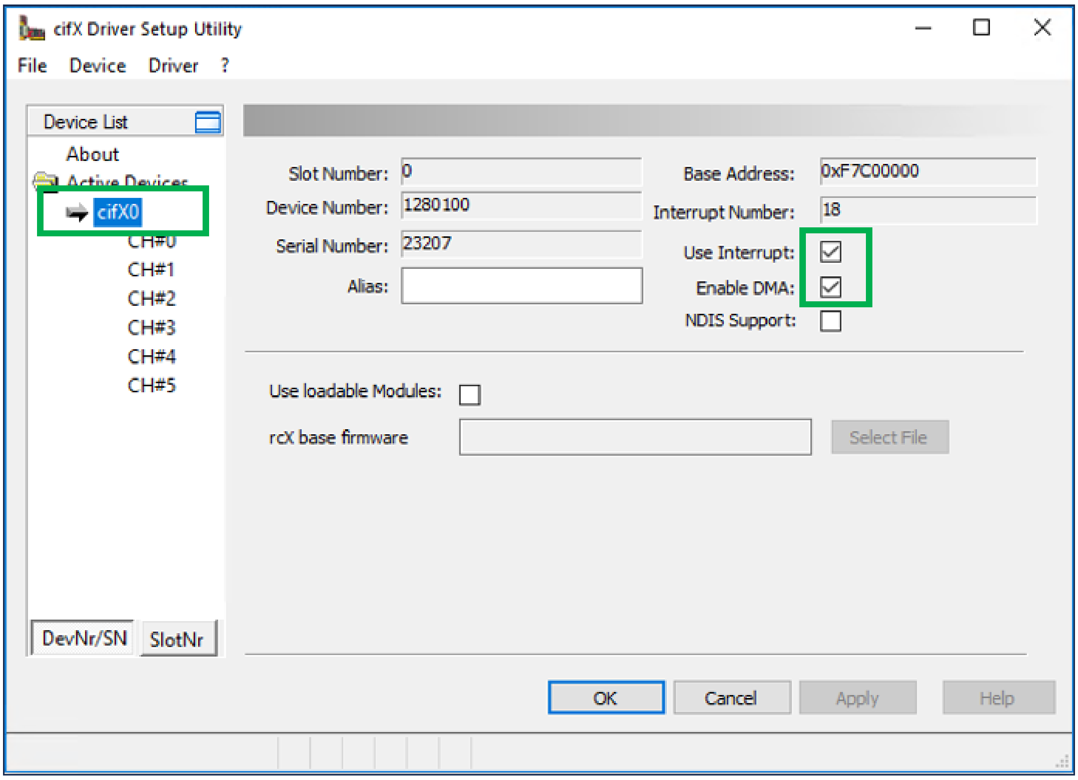
4. When the installation is finished, press the button “Finish” to exit he setup.


cifX Device Driver (x64) 2.6.1.0 Setup
Completed the cifX Device Driver (x64) 2.6.1.0 Setup Wizard
Click the Finish button to exit the Setup Wizard.
Back
Finish
Cancel

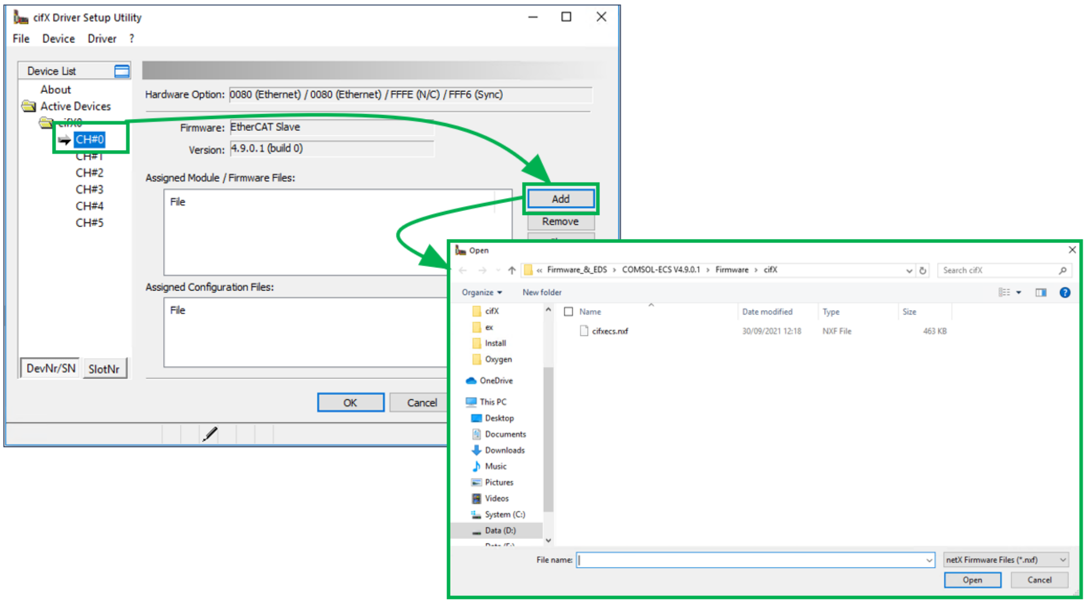
5. Start cifX Driver Setup Utility.
IMPORTANT! The TRION-ETHERCAT-S1 module must be attached to the device.
After the successful installation of the cifX device driver the setup utility can be found in the installation path which is normally: C:\Program Files\cifX Device Driver\cifXSetup.exe



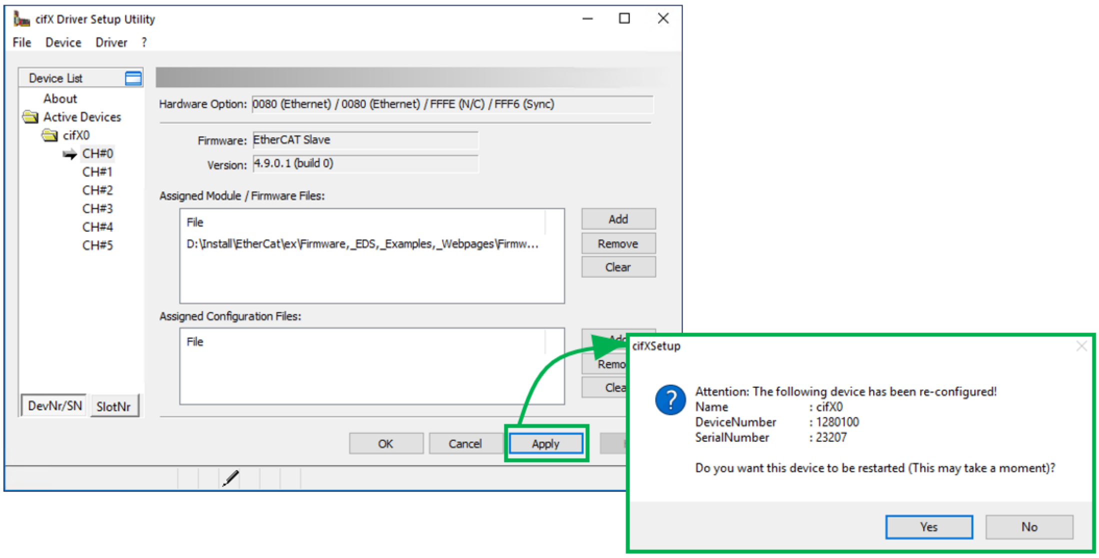
6. Select cifX0 and check "Use Interupt" and "Enable DMA" as shown in following screenshot. If there is displayed "Unknown" instead of "cifX0", there is no TRION-ETHERCAT-S1 module attached to the device.



7. Select CH#0 and add the correct firmware, which can be found within the extracted files of "Communication_Solutions_90_2022-03-1.zip" under following path:
.......\Firmware,_EDS,_Examples,_Webpages\Firmware_&_EDS\COMSOL-ECS V4.9.0.1\Firmware\cifX

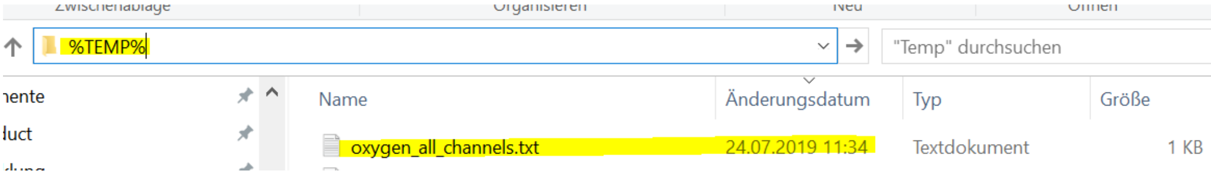


8. Apply the firmware to the device by clicking the button "Apply". A window will pop up, where it will be asked if the device shall be restarted or not. Click "Yes".



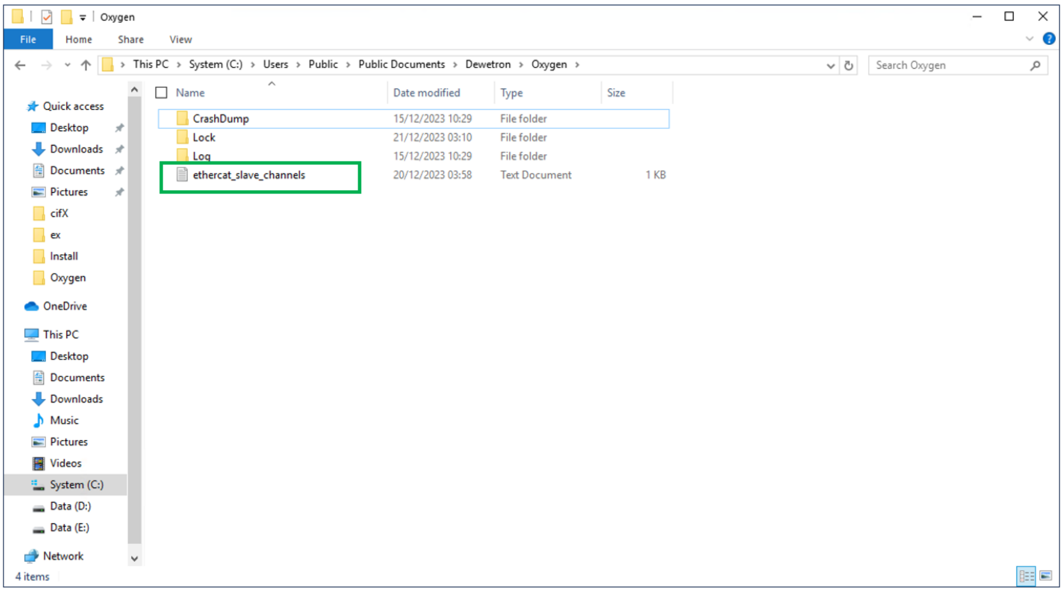
9. Close the setup tool by clicking on "OK".

1. Close Configuration Tool
2. Open OXYGEN
3. Go to %TEMP% in explorer
4. Check if oxygen_all_channels.txt exists with actual date and time

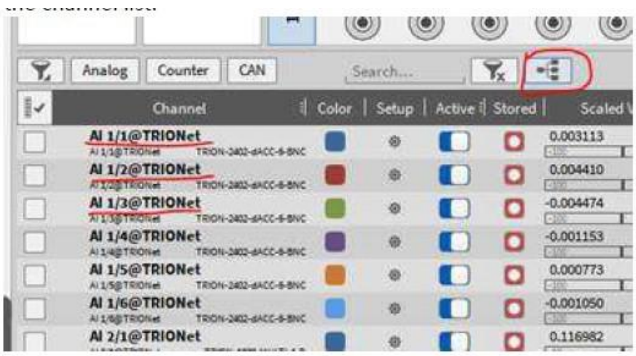


5.1.2 AFTER THE INSTALLATION

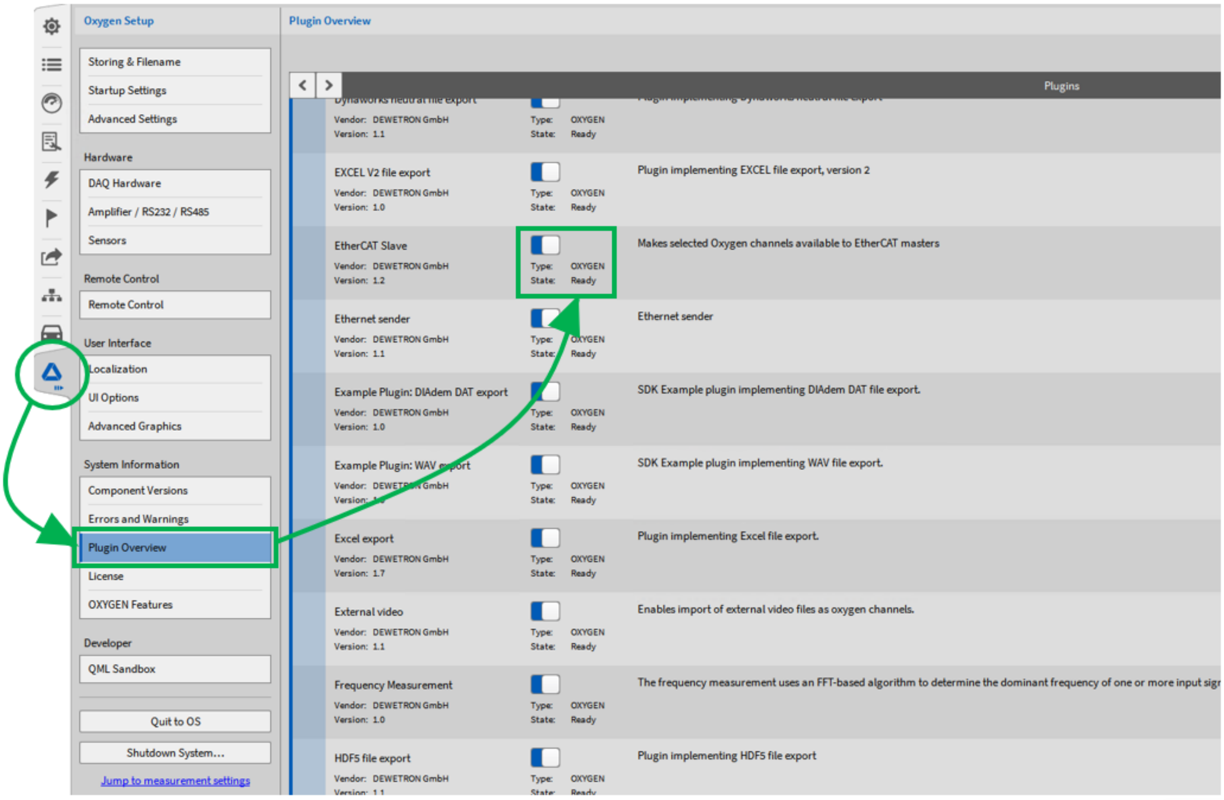
1. After installation, verify the files are in the proper location.



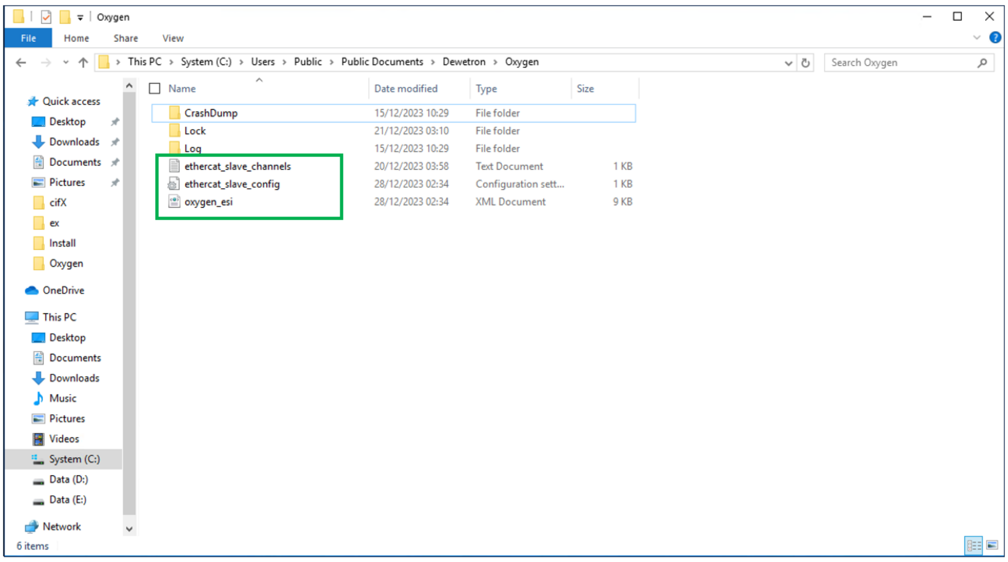
The board will not show up in OXYGEN since it uses third party hard- and software. Therefore, you cannot check if the board is recognized in OXYGEN. Make sure that the customer uses the long channel names for the text file. The long channel names can be seen by clicking on the ungroup button in OXYGEN, then all the long channel names are shown in the channel list. Restart OXYGEN, otherwise, the ESI file will not be generated.



The only possibility in OXYGEN to check if the EtherCAT module has been detected correctly and the driver as well as the firmware have been updated, is to switch to the plugin overview in OXYGEN. If the EtherCAT Slave Plugin has the state "Ready" everything has been installed correctly.



After restarting OXYGEN all files from the section PREFACE in this document must be available now in C:\Users\Public\Documents\Dewetron\Oxygen



6 USING LABVIEW 2017 AND THE NATIONAL INSTRUMENTS MASTER

When using the Dewetron's ESI file generated previously, the master will need to certify all channels that are required to be used and mapped to LabVIEW are available. Mapping each channel is similar to designing any set of test equipment being commanded. The drag and drop LabVIEW icons are easily mapped to the correct input / output. If all the channels needed are not shown, it could be a PDO or SDO issue.

6.1 SETTING UP THE HARDWARE

This tutorial uses a cRIO-9074 CompactRIO Controller as the EtherCAT master. To set up this hardware system, follow these steps:

1. Insert the C Series I/O modules in the appropriate chassis (the CompactRIO master chassis and/or the NI-9144 slave chassis).
2. Connect a standard Category 5 or better Ethernet cable from cRIO-9074 Ethernet Port 2 to NI-9144 Ethernet Port 1. You may daisy chain multiple NI-9144 chassis from the controller in this manner.

Note: Port 1 on the cRIO-9074 is on the bottom. Port 1 on the NI 9144 is on the top.

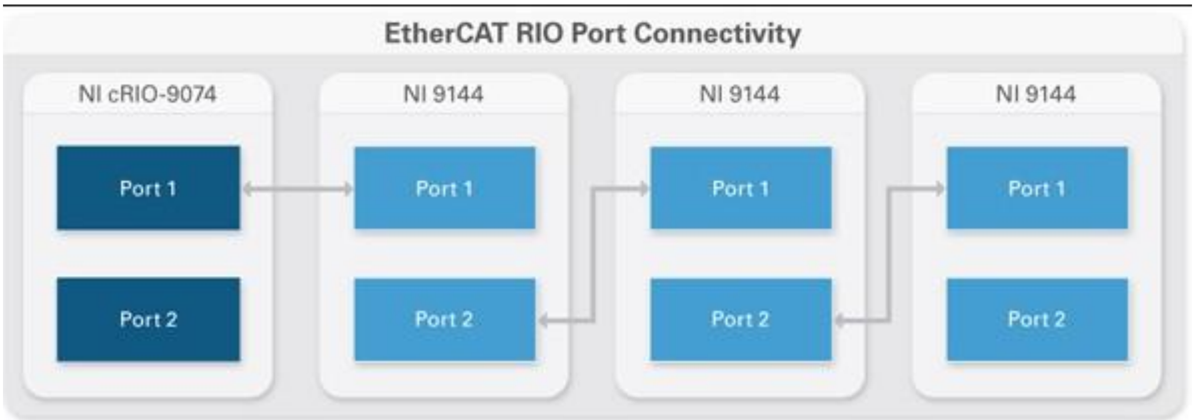


Figure 3-1: Connecting the Ethernet Ports on the CompactRIO Master Controller and NI 9144 Slave Chassis

3. Wire an external power supply to each chassis, which then powers on the hardware.

6.2 CONFIGURING THE MASTER CONTROLLER

1. After you have connected the hardware, install the required software on the host computer.
2. Use an Ethernet cable to connect cRIO-9074 Ethernet Port 1 to the same network as your host PC.

3. Launch the NI Measurement & Automation Explorer (MAX) configuration utility from Start » Programs » National Instruments » Measurement & Automation. Double-click on Remote Systems in the Configuration panel tree on the left to autodiscover the cRIO-9074 on the network and click on the CompactRIO controller.
4. Change the name of the master controller and select Obtain IP address from DHCP server under IP Settings. Click Reboot when you are done. (To use a static IP address, see the related links below.)

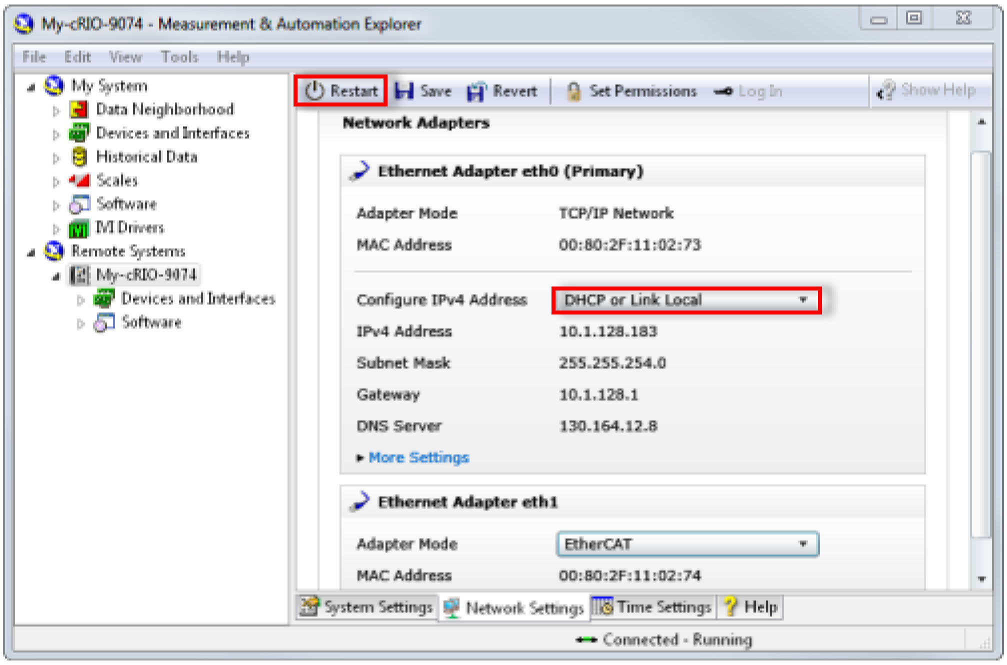


Figure 3-2: Setting the IP Address for the Real-Time Controller

5. To install the software on the real-time master controller, expand the cRIO-9074 under Remote Systems. Right-click Software and select Add/Remove Software.

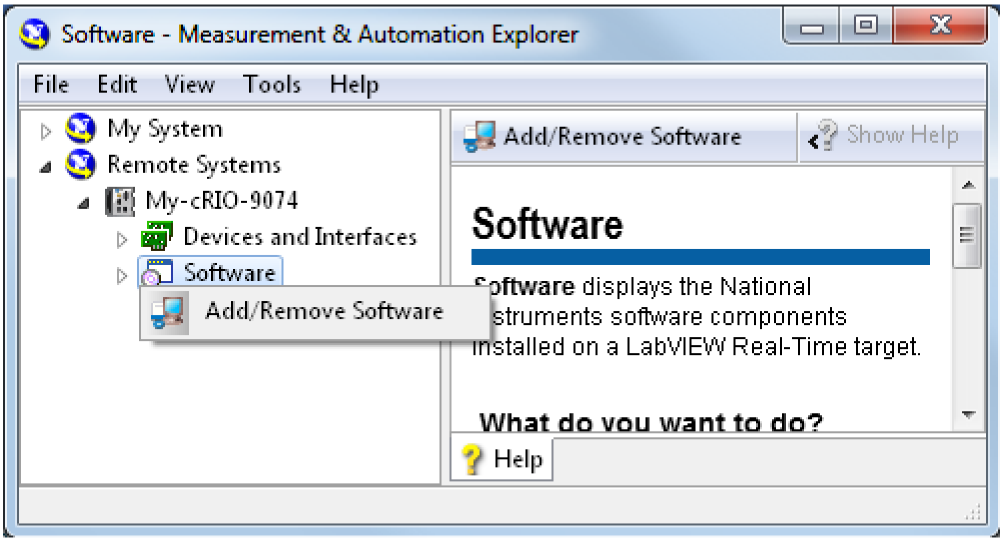


Figure 3-3: Installing Software on the Real-Time Controller

6. Install the recommended software set—NI-RIO driver software with the NI Scan Engine as well as NI-Industrial Communications for EtherCAT driver software—onto the CompactRIO controller.
7. Once the controller has rebooted, find the CompactRIO controller under Remote Systems again. Click on Advanced Ethernet Settings in the lower right-hand corner.
8. Select cRIO-9074 Ethernet Port 2 (the MAC address that is not primary). Then select EtherCAT in the pull-down box under Mode and click OK.

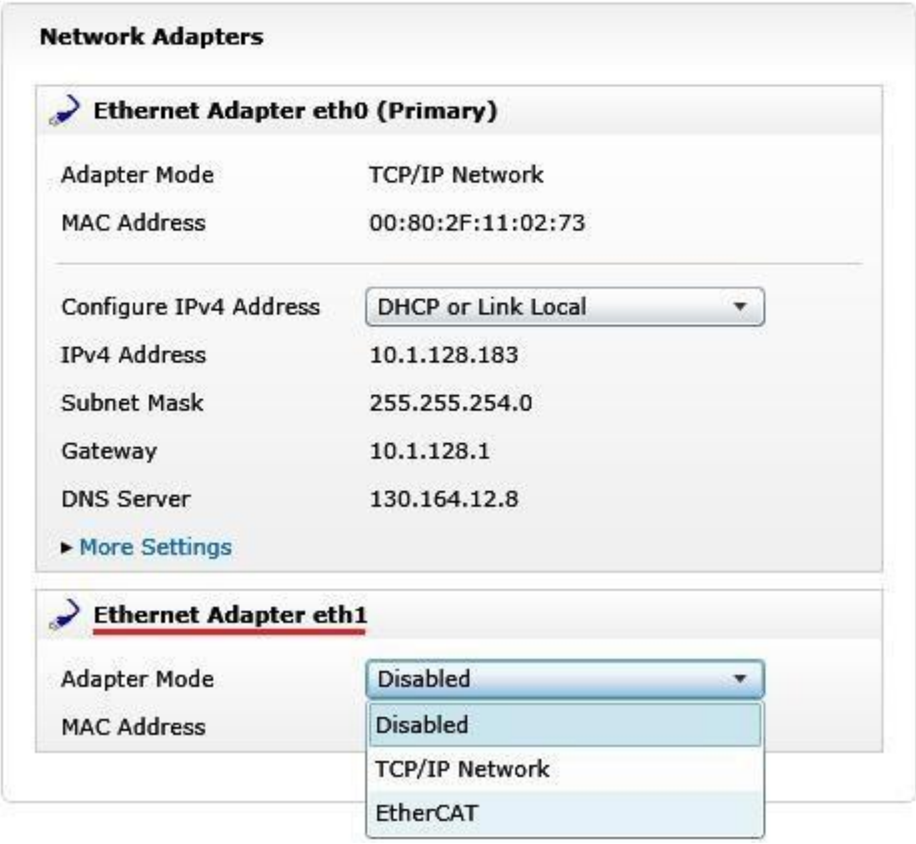


Figure 3-4: Selecting EtherCAT Mode for the Ethernet Port of the Real-Time Controller

Note: When the controller's Ethernet Port 2 is in EtherCAT mode, you cannot use that port on an Ethernet network.

6.3 CONFIGURING THE NETWORK IN LABVIEW

1. Launch LabVIEW from Start » Programs » National Instruments » LabVIEW X.X » LabVIEW. Click on Empty Project.
2. Right-click on Project and select New » Targets and Devices.
3. In the Add Targets and Devices dialog window, select Existing target or device and expand the category Real-Time CompactRIO to autodiscover the EtherCAT master on the host PC's subnet. Select the cRIO-9074 and click OK.
4. In the LabVIEW Project window, right-click on the cRIO-9074 and select New » Targets and Devices.

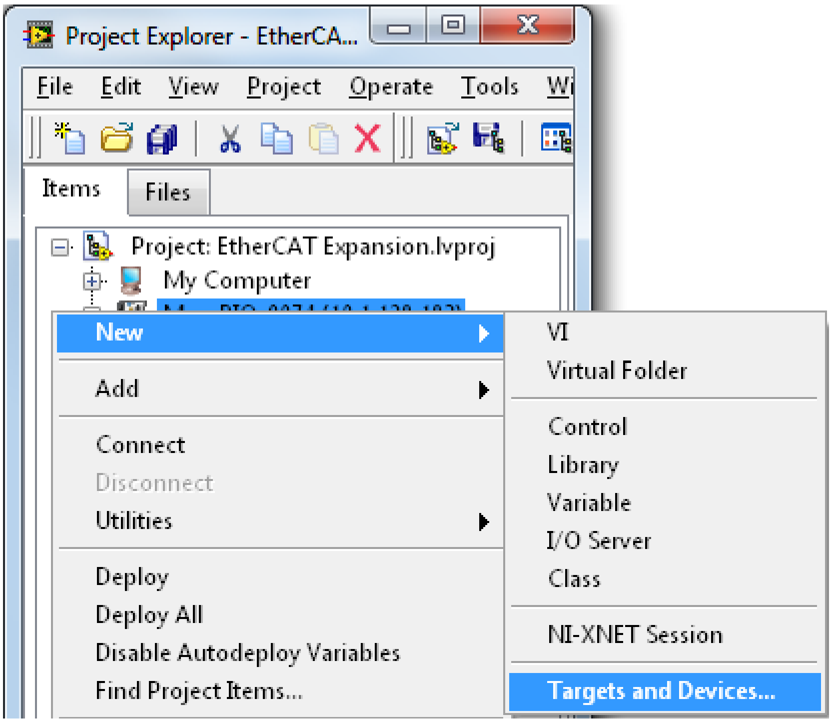


Figure 3-5: Using LabVIEW to Discover Controllers Connected to the Network

5. In the Add Targets and Devices dialog window, select Existing target or device and expand the category EtherCAT Master Device to autodiscover the EtherCAT port on the controller.

Figure 3-6: Selecting the EtherCAT Master Port on the Real-Time Controller

6. Select the available EtherCAT master port and click OK. The Scan Slaves dialog window appears. The first option Scan Interface autodiscovers any slaves connected to the controller. Click OK. The LabVIEW Project now lists each slave device, its I/O modules, and the physical I/O on each module (called I/O variables).
7. Once all the EtherCAT slaves have been autodiscovered, you may create a VI, or logic program, on the CompactRIO master controller. To program the NI 9144 modular slave in LabVIEW Real-Time, click and drag the I/O variables from the LabVIEW Project to the block diagram. With these I/O variables, you can read and write from physical channels in the NI 9144 chassis. When finished, click the Run button on the LabVIEW VI program to deploy the program to the real-time controller.

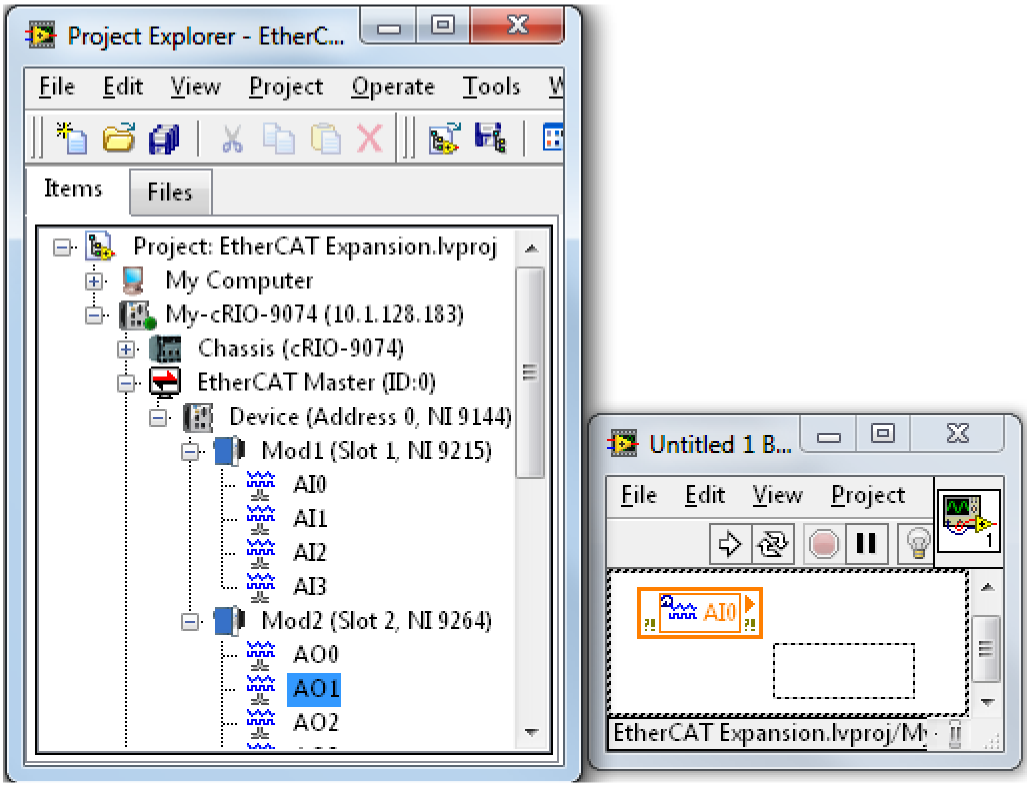


Figure 3-7: Dragging I/O Variables From the LabVIEW Project to the Block Diagram

Note: If you use the I/O variables in a Timed Loop, you can synchronize the loop timing source to the NI Scan Engine. When you run the VI, the program automatically deploys onto the CompactRIO master. For more information about the LabVIEW I/O variable and CompactRIO Scan Mode, see the white paper titled *Understanding CompactRIO Scan Mode*.

6.4 CHANGING ETHERCAT MODES IN LABVIEW

LabVIEW streamlines the different EtherCAT states (that is, Operational, Init, Bootstrap, and so on) by grouping them into two LabVIEW Scan Engine modes: Configuration Mode and Active Mode. In Active Mode, the LabVIEW program gets real data from the network. In Configuration Mode, the network data is not updated and configuration of the EtherCAT network is enabled, allowing you to add new slave devices and modules and download the NI-9144 EtherCAT slave firmware.

To change the LabVIEW Scan Engine mode, right-click the cRIO-9074 master controller in the LabVIEW project explorer and select Utilities » Scan Engine Mode » Switch to Active (or Switch to Configuration). The LabVIEW Scan Engine mode can also be changed programmatically with the Set Scan Engine Mode function, accessed by right-clicking on the LabVIEW block diagram and selecting the Measurement I/O » NI Scan Engine palette.

You can change an individual slave's EtherCAT state by right-clicking on the slave device in the LabVIEW Project window and selecting Online Device State. The slave device's EtherCAT state can be changed almost completely independently from the LabVIEW Scan Engine mode, with only two restrictions. One restriction is that you cannot place a slave device into Operational or Safe-Operational state unless the LabVIEW Scan Engine is in an equivalent state (Active Mode). The other restriction is that you cannot place your device into Bootstrap state unless the LabVIEW Scan Engine is in Configuration Mode.

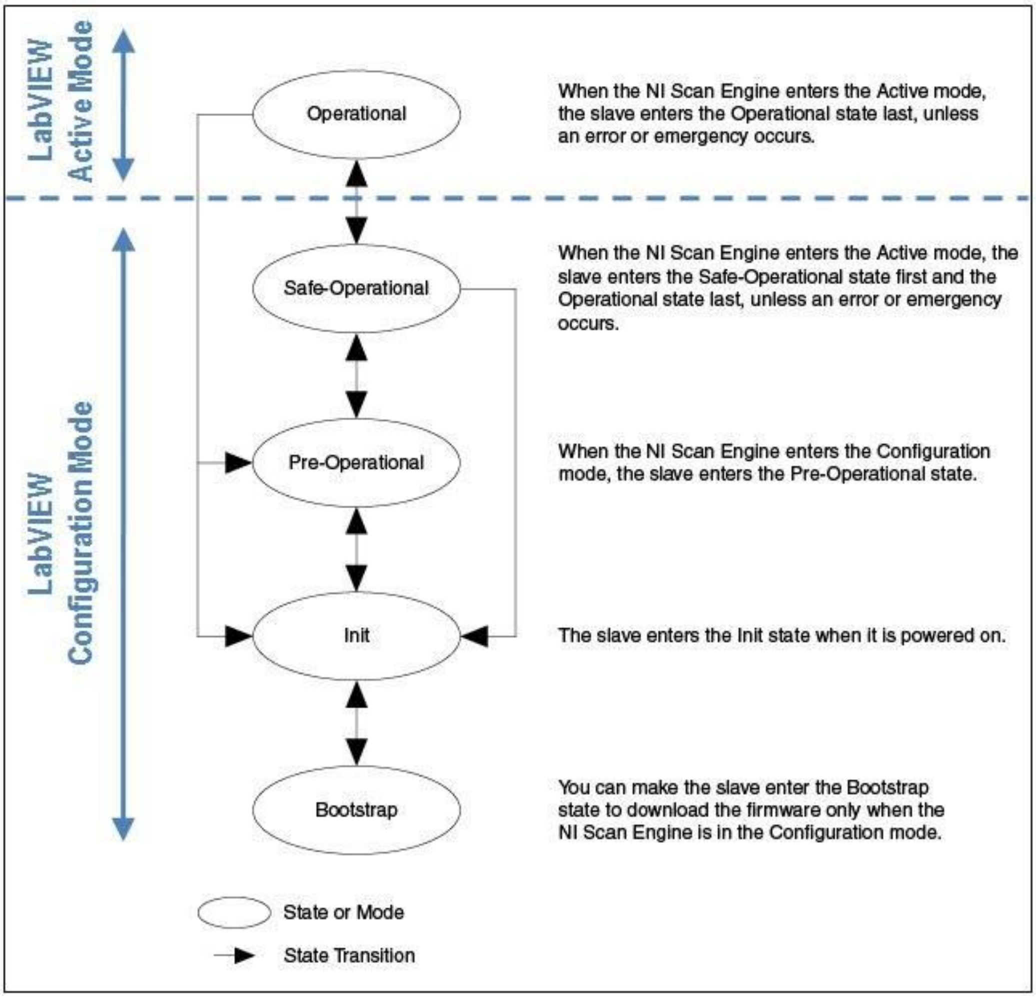


Figure 3-8How LabVIEW Scan Engine Modes Correspond with EtherCAT States

6.5 ADDING THIRD-PARTY ETHERCAT SLAVES TO NI MASTER CONTROLLERS

To have a third-party slave device show up in the LabVIEW Project window, you must import the slave's device profile XML file before configuring the slave with LabVIEW.

1. Locate the EtherCAT XML Device Description File for your third-party slave. Some companies provide these XML files for download on their websites.
2. Right-click the EtherCAT master item and select Utilities » Import Device Profiles to display the Import Device Profile dialog box.
3. Click the file Browse button to select the device profile(s) in either of the following ways.
   - Select a file and click Open to import a device profile.
   - Open a folder and click Current Folder to import all device profiles under the folder.
4. After selecting the device profiles, you can see the file import results. The file is imported successfully if Import Result reads Succeed. Otherwise, the file is not imported due to an invalid format or other reason.
5. (Optional) Repeat step 2 to import other device profiles.
6. Click Close to exit the dialog.
7. You must restart LabVIEW to use the newly imported device profile.

6.6 PROGRAMMING THE NI 9144 FPGA IN LABVIEW FPGA

The program written in LabVIEW Real-Time is deployed to the EtherCAT master, which is responsible for collecting, processing, and controlling the I/O in the EtherCAT network. Additionally, each NI-9144 slave chassis contains an embedded field-programmable gate array (FPGA) that can execute custom timing and signal processing to help you create intelligent distributed devices that are synchronized within 100 ns of each other. To take advantage of the programmable FPGA capabilities, you must use LabVIEW, LabVIEW Real-Time, and LabVIEW FPGA 2009 or later.

1. When you add the EtherCAT Master Device to the LabVIEW Project (see step 5 in the previous section), you can either set up the NI-9144 slave to program in the NI Scan Interface or LabVIEW FPGA Interface. If you select the former, you can still program in LabVIEW FPGA by right-clicking on Device (Address 0, NI-9144) in the LabVIEW Project window and selecting New » FPGA Target.

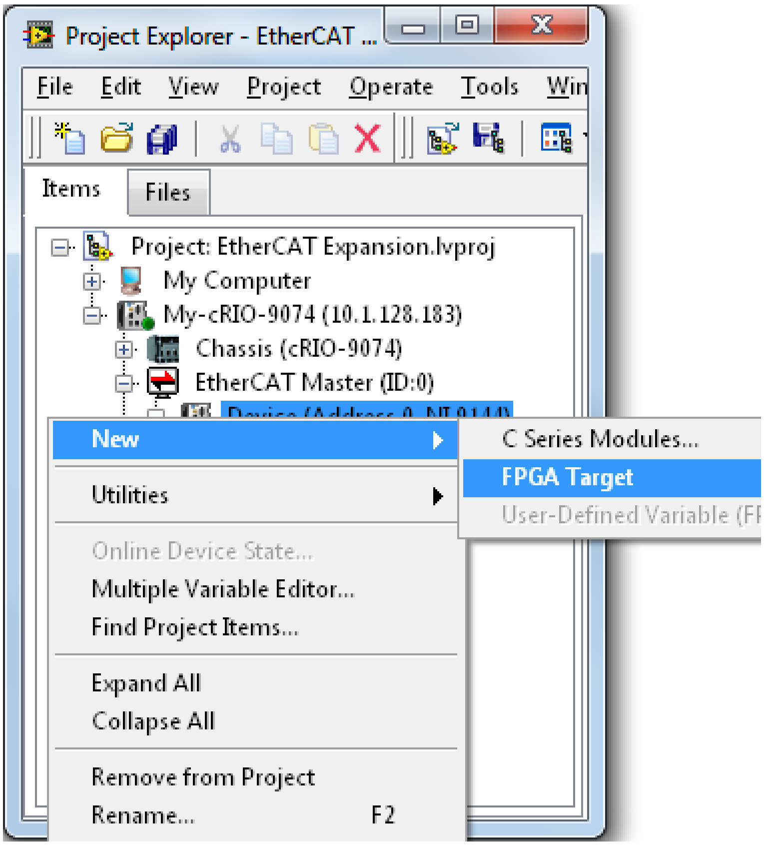


Figure 3-9: Discovering the FPGA Target on the NI 9144

2. In the LabVIEW Project, the NI-9144 slave device features several FPGA signals. For example, Input Virtual Point and Output Virtual Point are digital timing signals used to synchronize FPGA code between different NI-9144 chassis. To program a module in LabVIEW FPGA Mode, drag and drop the module from Device (Address 0, NI-9144) onto the FPGA Target (NI-9144). In the Project Explorer window, you can drag and drop modules between Device and the FPGA Target to switch between Scan Mode and FPGA Mode.

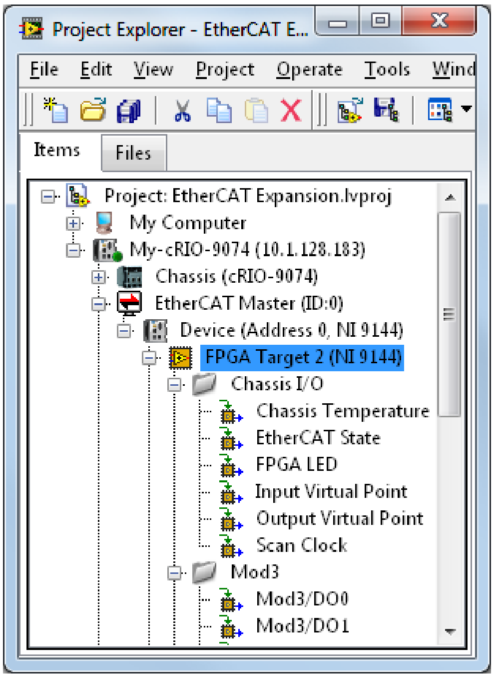


Figure 3-10: Chassis I/O and Module I/O Listed Under the FPGA Target

3. In the LabVIEW Project Explorer window, right-click on FPGA Target (NI-9144) in the LabVIEW Project and select New » VI to create a new FPGA program on the NI 9144. The FPGA I/O channels from modules listed under FPGA Target (NI-9144) can be dragged and dropped into the block diagram of this new FPGA VI.
4. To deploy it to the FPGA Target (NI-9144), the VI must be compiled through creating a new Build Specification » New » Compilation. After the FPGA VI has compiled successfully, it will be listed under Build Specifications in the Project Explorer and can be downloaded to the FPGA Target (NI-9144) by right-clicking on it and selecting Download.

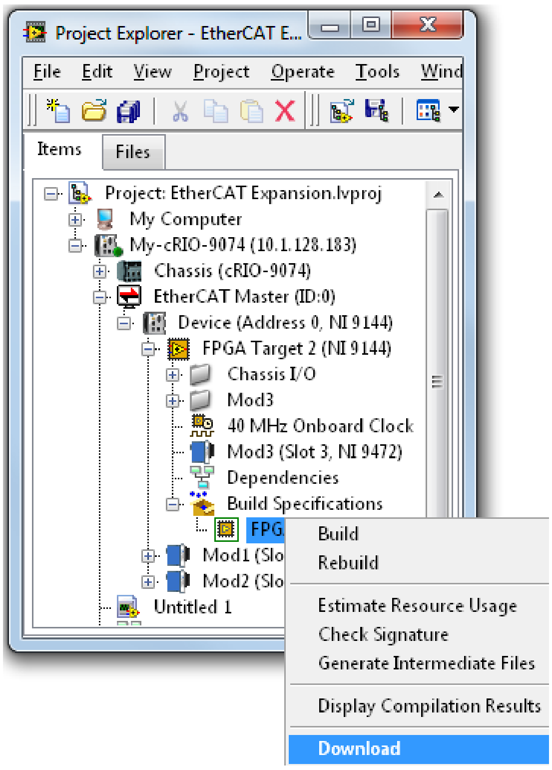


Figure 3-11: FPGA VI Compiled for Downloading to the NI 9144 FPGA

6.7 DIFFERENCES IN THE LABVIEW FPGA PROGRAMMING API

If you are familiar with programming in LabVIEW FPGA for CompactRIO and other NI RIO platforms, there are some differences between programming the master's local FPGA and the slave's FPGA. Since LabVIEW 2009, user-defined I/O variables are used to synchronize FPGA data with the Scan Engine. These user-defined I/O variables are also the only means for transferring data between the controller's real-time VI and the-NI 9144 expansion chassis' FPGA VI.

FPGA Transfer Methods	Local Chassis	Expansion Chassis
User-defined I/O variables	√	√
FPGA Host Interface	√	X
DMA transfer functions	√	X
FPGA front panel debugging	√	X

Table 3-1: FPGA Transfer Methods for Local Versus Expansion Chassis

1. Program the FPGA VI by clicking the NI 9144 FPGA I/O nodes from the LabVIEW Project and dragging them onto the FPGA block diagram. In this example, a proportional integral derivative (PID) function was used to control analog I/O in the NI 9144 chassis.

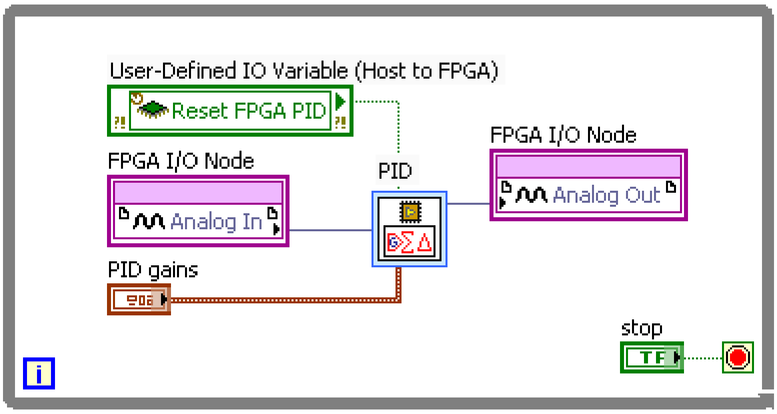


Figure 3-12: Example FPGA Code on the NI 9144

2. Note the Reset FPGA PID Boolean variable. This is a user-defined I/O variable that allows the real-time VI on the NI EtherCAT master to communicate with the FPGA VI on the NI-9144. To create such a variable, right-click on Device (Address 0, NI-9144) in the LabVIEW Project and select New » User-Defined Variable. The variable name, data type, and data direction (from Host to FPGA or vice versa) may be set from the Properties window. In this case, the master controller is sending information to the NI-9144 slave, so the direction is Host to FPGA.

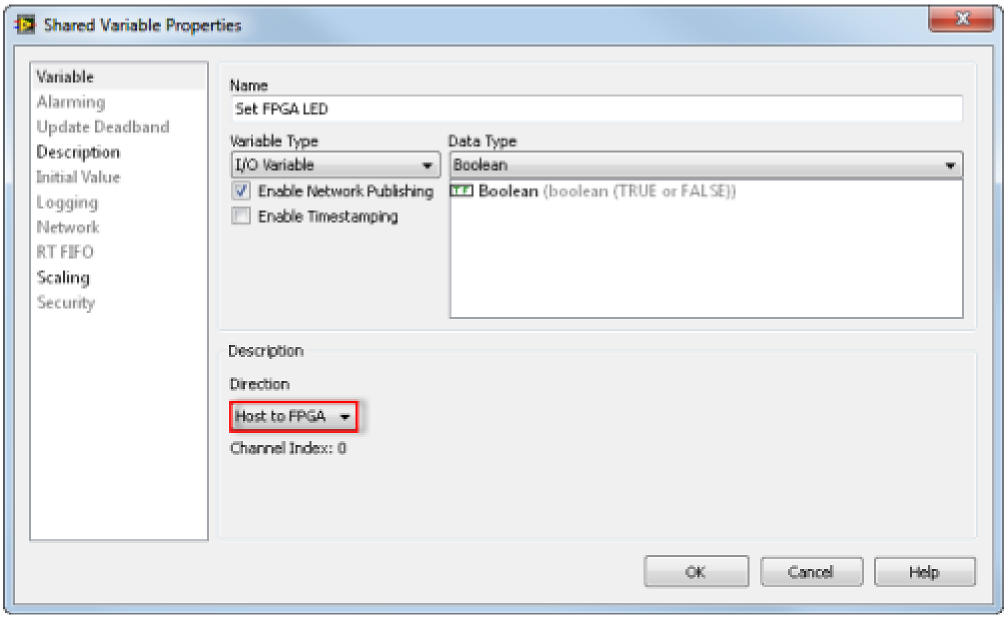


Figure 3-13: Dialogue Window for Creating a User-Defined I/O Variable

You can now program the user-defined I/O variable in the real-time VI used in this example to reset the PID function running in the FPGA VI.

3. To run the FPGA VI, right-click on cRIO-9074 in the LabVIEW Project and select Deploy All to deploy the NI-9144 slave device. Put the LabVIEW Scan Engine into Configuration Mode by right-clicking the CompactRIO controller and selecting Utilities » Scan Engine Mode » Switch to Configuration and click the Run button on the FPGA VI. This starts the compile process and downloads the FPGA bitfile to the NI-9144 chassis. After you have successfully downloaded the bitfile to the NI-9144, the FPGA VI begins executing immediately.

Note: You may need to update the firmware on your NI-9144 slave chassis to run custom FPGA code. If you get an error message stating that your firmware needs to be updated, follow the instructions in the NI-9144 User Guide under the Updating Your Firmware section.

4. When the compile and download process completes, you need to set the LabVIEW Scan Engine back into Active Mode before the FPGA VI will begin executing. To do so, right-click the CompactRIO master controller in the LabVIEW Project window and select Utilities » Scan Engine Mode » Switch to Active.

An important note is that there is a limit to the number of user-defined I/O variables that can be created in FPGA Mode. The NI-9144 slave chassis can hold a total of 512 bytes of input data and 512 bytes of output data for both I/O variables in Scan Mode and user-defined I/O variables in FPGA Mode. For example, if you are using four 32-channel modules in Scan Mode and each channel takes up 32 bits of data, then 256 bytes of input data are being used by Scan Mode I/O variables. With the remaining 256 bytes of input data, you can create 64 input user-defined I/O variables (also of 32-bit length) in FPGA Mode.

7 TROUBLESHOOTING

What to do if the ESI file does not get generated or errors exist Check the following:

- Is the device driver for cifx installed?
- Has the netX configuration been installed?
- Did you exchange the firmware and download to the board in the config tool? (COMSOL-ECS V4.7.0.6.zip)
- Did you change the vendor ID and download to the board in the config tool?
- Did you adapt Vendor ID to 0x00000B0F?
- Setup the ethercat_slave_channels.txt in %PUBLIC%\Documents\Dewetron\Oxygen\
  At least one channel must be listed in the ethercat_slave_channels.txt, otherwise the Ethercat plugin will not be activated in OXYGEN.
- You can find a list with all channels in %TEMP%\oxygen_all_channels.txt
- If all is working fine OXYGEN should automatically create an ESI file in %PUBLIC%\Documents\Dewetron\Oxygen\

Contact customer support if In the National Instruments master llb program, does not show the correct channels or only a portion of them.
An example of this:
An SDO (service data object)? NI EtherCAT masters do not work with SDOs…
Beckhoff Masters can use SDO. The majority of automotive drivetrain and gear test labs use National Instruments in the USA.
Sample code using Beckhoff Masters
The example below is using National Instruments 2017 software.


```
   </RxPdo>
 - <TxPdo Mandatory="1" Fixed="1" Sm="3">
     <Index>#x00001A00</Index>
     <Name>Channels</Name>
   - <Entry>
       <Index>#x3000</Index>
       <SubIndex>0</SubIndex>
       <BitLen>64</BitLen>
       <Name>Current Time</Name>
       <DataType>ULINT</DataType>
     </Entry>
   - <Entry>
       <Index>#x3500</Index>
       <SubIndex>0</SubIndex>
       <BitLen>32</BitLen>
       <Name>Oxygen Flags</Name>
       <DataType>UDINT</DataType>
     </Entry>
   - <Entry>
       <Index>#x3001</Index>
       <SubIndex>0</SubIndex>
       <BitLen>32</BitLen>
       <Name>AI 1/I1</Name>
       <DataType>REAL</DataType>
     </Entry>
   - <Entry>
       <Index>#x3002</Index>
       <SubIndex>0</SubIndex>
       <BitLen>32</BitLen>
       <Name>AI 1/I2</Name>
       <DataType>REAL</DataType>
     </Entry>
   - <Entry>
       <Index>#x3003</Index>
       <SubIndex>0</SubIndex>
       <BitLen>32</BitLen>
       <Name>AI 1/I3</Name>
       <DataType>REAL</DataType>
     </Entry>
   - <Entry>
       <Index>#x3004</Index>
       <SubIndex>0</SubIndex>
       <BitLen>32</BitLen>
       <Name>AI 1/I4</Name>
```