EtherCAT with Single-Axis Motor Control Reference Design

Reference Design

FPGA-RD-02342-1.0

May 2026

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Inclusive Language

This document was created consistent with Lattice Semiconductor's inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice's inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Figures

Tables

Abbreviations in This Document

A list of acronyms and abbreviations used in this document.

Abbreviations	Definition
AHB	Advanced High-Performance Bus
AHBL	Advanced High-Performance Bus Lite
APB	Advanced Peripheral Bus
AXI	Advanced eXtensible Interface
BIOS	Basic Input/Output System
DC	Distributed Clocks
EEPROM	Electrically Erasable Programmable Read-Only Memory
EPV	Erase–Program–Verify
ESC	EtherCAT Slave Controller
ESI	EtherCAT SubDevice Information
EtherCAT	Ethernet for Control Automation Technology
ETG	EtherCAT Technology Group
FOC	Field-Oriented Control
FMMU	Fieldbus Memory Management Unit
GPIO	General Purpose Input/Output
GHRD	Golden Hardware Reference Design
IP	Intellectual Property
MII	Media Independent Interface
PDO	Process Data Object
PDI	Process Data Interface
PHY	Physical Layer
PLC	Programmable Logic Controller
PLL	Phase-Locked Loop
PREOP	Pre-Operational
PWM	Pulse Width Modulation
RPM	Revolutions Per Minute
RxPDO	Receive Process Data Object
SAFEOP	Safe-Operational
SDO	Service Data Object
SM	Sync Manager
SSC	EtherCAT SubDevice Stack Code
TCM	Tightly Coupled Memory
TxPDO	Transmit Process Data Object
UART	Universal Asynchronous Receiver/Transmitter
VT-d	Virtualization Technology for Directed I/O
VT-x	Virtualization Technology
XAE	eXtended Automation Engineering
XAR	TwinCAT Runtime

1. Introduction

This EtherCAT with Motor Control reference design comprises a complete RISC-V embedded system integrated with the Beckhoff EtherCAT SubDevice IP and the Lattice FOC Motor Control IP. A RISC-V RX soft processor runs the Beckhoff EtherCAT Slave Controller (ESC) software stack to provide the communication protocol interface to the EtherCAT SubDevice IP. The design is operated using Beckhoff TwinCAT automation software installed on a host PC, which acts as the EtherCAT MainDevice to control the motor.

1.1. Quick Facts

Download the reference design files from the Lattice reference design web page.

Table 1.1. Summary of the Reference Design

General	Target development kit	Secure Connected Motion Control Platform
	Target device	LFCPNX-100-9LFG672I
Software Requirements	Software tool and version	• Lattice Radiant™ software 2025.2.1 • Lattice Propel™ Builder 2025.2.1 • TwinCAT v3.1 Build 4024 • EtherCAT Slave Stack Code (SSC) ET9300 [1]
	IP version	• RISC-V RX IP v2.8.0 • TCM IP v1.5.3 • System Memory IP v2.5.1 • PLL IP v1.9.1 • EtherCAT IP v4.0.0 [1] • Motor Control IP v3.1 • Unified Interconnect v1.0.0 • GPIO IP v1.8.0 • Timer-Counter IP v1.4.0 • Octal SPI Controller IP v1.4.0 • UART IP v1.5.0 • AHBL Interconnect IP v1.5.0 • APB Interconnect IP v1.4.0 • AHBL to APB Bridge v1.2.0 • AXI to AHBL Bridge v1.5.0
Hardware Requirements	Board	• RS485 Encoder Transceiver Board • Trenz Electronic TEP0002 motor driver board
	Cable	• Heidenhain 8-pin M12 encoder cable (1133832-01) • 24-V power supply and adapter • RJ-45 cable • USB cable (USB-A to Mini-B)
	Host PC	• Windows® 10 PC with supported network controller[2]
	Miscellaneous	• Anaheim Motor (Dual Shaft) BLY171D-24V-4000 • Heidenhain ROQ 437 Absolute Rotary Encoder (1322273-05)

Notes:

1. Third-party vendor IP and software stack code. You must acquire them from Beckhoff to proceed with software and hardware compilation and add the Beckhoff IP license into the Radiant license directory to ensure successful compilation in the Radiant software.
2. For more information on supported network controllers, refer to the *TwinCAT 3 Supported Network Controllers* web page on the Beckhoff Information System website (TwinCAT 3 -> Product Overview -> System requirements).

1.1.1. Quick Start Guide

The steps to set up the hardware and run the motor control designs are detailed in the Running the Reference Design section.

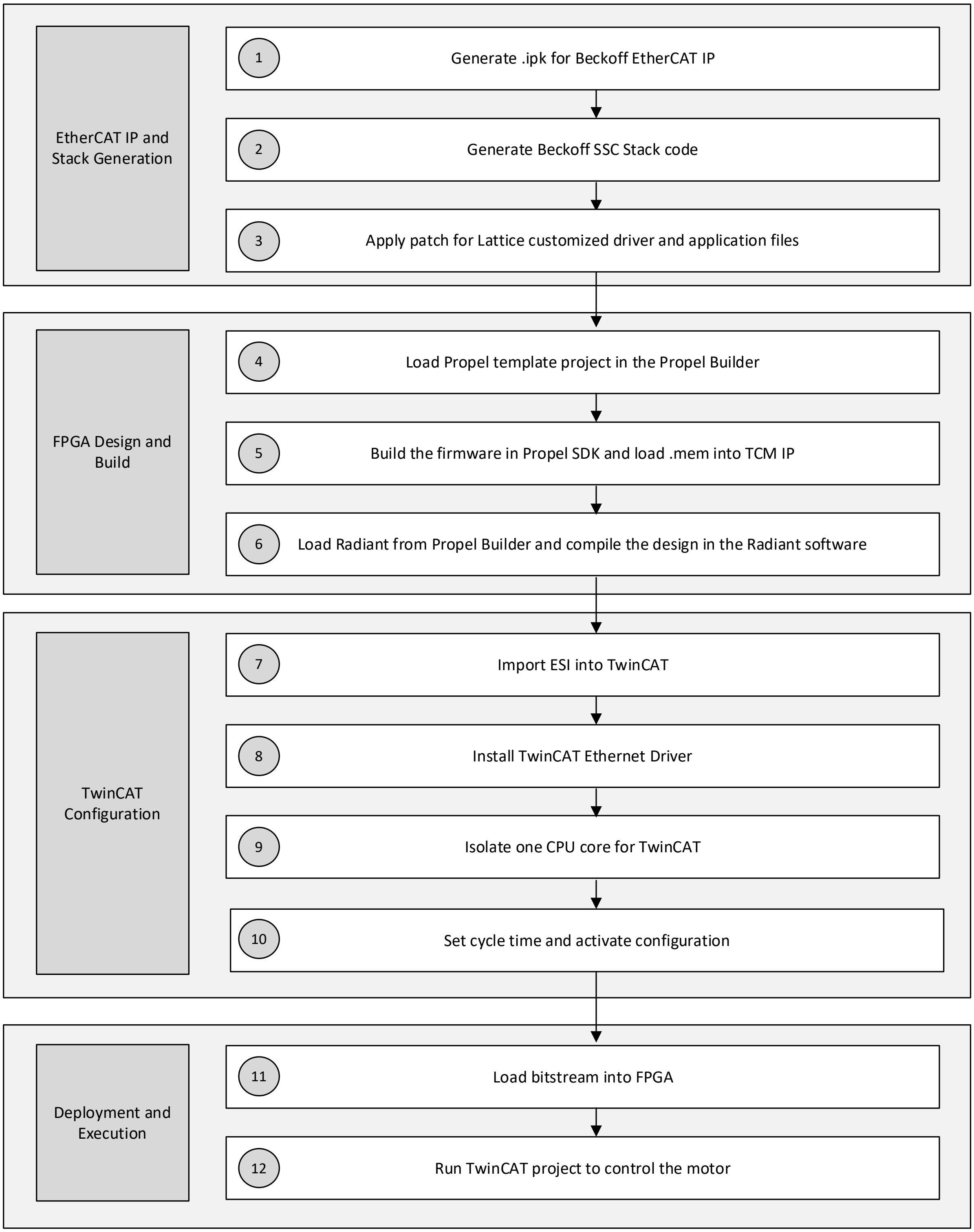

Figure 1.1. Simplified Reference Design Setup Flow

1.2. Features

Key features of the design include:

- Motor Control with EtherCAT as the real-time Ethernet network.
- Single-node SubDevice implementation with EtherCAT SM Synchronous Mode.
- Position Control of Motor.
- Speed Control of Motor.
- 25-bit position feedback from encoder and 12-bit phase-current sensing.
- Various features from the original Motor Control Reference Design. For more information, refer to the FOC Motor Control Reference Design User Guide (FPGA-RD-02325).

2. Directory Structure and Files

In your downloaded reference design **.zip** folder, you will find two folders—*Latticesemi_LFCPNX_EtherCAT_1-Axis_Motor_control_<version>* and *supp*. The first folder is a Propel Builder Template folder that you must copy into your *PropelTemplateLocal* folder. This folder usually resides in your user directory after you have installed the Propel software. For example, *C:\Users\<your username>\PropelTemplateLocal*. The supp folder is a supplementary folder that contains other files required to set up the project.

```
Latticesemi_LFCPNX_EtherCAT_1-Axis_Motor_Control_25.2/
├── application
│   ├── board.c
│   ├── board.h
│   ├── bsp
│   ├── cpnx.cfg
│   ├── cpu.svd
│   ├── cpu.yaml
│   ├── debuglib.c
│   ├── debuglib.h
│   ├── ethercat
│   ├── FreeRTOSConfig.h
│   ├── FreeRTOS-Kernel
│   ├── hooks.c
│   ├── linker.ld
│   ├── lscc_ecat_ssc.patch
│   ├── main.c
│   ├── mmio.h
│   ├── motor_helper.h
│   ├── osASM.S
│   ├── os.c
│   ├── osmacro.h
│   ├── sys_env.xml
│   ├── utils.c
│   ├── utils.h
│   └── vector.S
├── LFCPNX_EtherCAT_1-Axis_Motor_Control.layout
├── LFCPNX_EtherCAT_1-Axis_Motor_Control.psc
├── LFCPNX_EtherCAT_1-Axis_Motor_Control.tcl
├── lib
│   ├── beckhoff.com
│   ├── encoder_subsys
│   ├── gluelogics
│   ├── latticesemi.com
│   └── motor_ctrl_m_code
├── radiant_init.tcl
└── Secure Connected Motion Control Platform(2)
    └── LFCPNX_EtherCAT_1-Axis_Motor_Control.pdc
```

Figure 2.1. Propel Builder Template Folder

Table 2.1. Propel Builder Template Folder File List

Folder / File	Description
application	Software application code to run the design.
LFCPNX_EtherCAT_1-Axis_Motor_Control.layout	Propel Builder mandatory templates files to construct into a Propel Builder project.
LFCPNX_EtherCAT_1-Axis_Motor_Control.psc	Documents the configuration parameters of the IP in IP-XACT 2014 format.
LFCPNX_EtherCAT_1-Axis_Motor_Control.tcl	This file provides an example RTL top file that instantiates the module.
lib	Contains all the IP library files and necessary glue logic RTL files.
radiant_init.tcl	A Tcl script that is executed when the Radiant software starts up.
Secure_Connected Motion Control Platform(2)	Contains post-synthesis constraint file.

```
supp
├── EtherCAT_CertusProNX_MCB_MII_AXI.ecpxml
├── EtherCATStack
│   ├── CPNX_MCB_ECAT_FOC_MotorControl.xlsx
│   ├── CPNX_MCB_ECAT_FOC_MotorControl.xml
│   ├── files
│   └── Lattice_CertusPro-NX_MCB_FOC_SSC_Config.xml
└── TwinCATProject_ADIN_ECAT_MotorCtrl.zip
```

Figure 2.2. Supplementary Folder

Table 2.2. Supplementary Folder File List

Folder / File	Description
EtherCAT_CertusProNX_MCB_MII_AXI.ecpxml	Contains the EtherCAT IP configurations to be used in the Beckhoff IPCore_Config.exe to generate the .ipk file.
EtherCATStack/CPNX_MCB_ECAT_FOC_MotorControl.xlsx	Application definition file for the EtherCAT SubDevice Stack Code (SSC) tool.
EtherCATStack/files	Contains all the driver and configuration files to interface with the EtherCAT SubDevice Stack Code (SSC) tool.
EtherCATStack/CPNX_MCB_ECAT_FOC_MotorControl.xml	Reference EtherCAT SubDevice Information (ESI) file generated by EtherCAT SubDevice Stack Code tool.
EtherCATStack/Lattice_CertusPro-NX_MCB_FOC_SSC_Config.xml	Configuration file to be imported into the EtherCAT SubDevice Stack Code (SSC) tool.
TwinCATProject_ADIN_ECAT_MotorCtrl.zip	Zipped file containing TwinCAT Project to interface with the reference design.

The following figure shows the generated project files from the template. For more information on the generation steps, refer to the Generate Design section.

```
ethercat_motor_ctrl
├── bht_ini_53.bin
├── bht_ini.bin
├── bht_ini_p.bin
├── ethercat_motor_ctrl
│   ├── application
│   ├── Eda_Ext_TclFiles
│   ├── ethercat_motor_ctrl.layout
│   ├── ethercat_motor_ctrl.pdc
│   ├── ethercat_motor_ctrl.sbx
│   ├── ethercat_motor_ctrl_tmpl.v
│   ├── ethercat_motor_ctrl_tmpl.vhd
│   ├── ethercat_motor_ctrl.v
│   └── lib
├── ethercat_motor_ctrl1.sty
├── ethercat_motor_ctrl.rdf
├── ethercat_motor_ctrl_tcl.html
├── ethercat_motor_ctrl_tcr.dir
│   └── pn260408234131.tcr
├── flist_design.f
├── impl_1
│   ├── ethercat_motor_ctrl_impl_1_ip_summary.xml
│   ├── ethercat_motor_ctrl_impl_1_summary.html
│   ├── hdla_gen_hierarchy.html
│   └── message_log.xml
├── radiant_setup_template.tcl
├── reginit.bin
├── reportview.xml
└── sge
    ├── application
    ├── bsp
    ├── cpu0.yaml
    └── sys_env.xml
```

Figure 2.3. Generated Design Folder

Table 2.3. Generated Design Folder File List

Folder / File	Description
<project_name folder> For example: *<ethercat_motor_ctrl>*	• Contains *application* and *lib* folders, which include all the software and hardware library files. • *<project_name>*.**pdc** includes the timing constraints and pin location of this design. • *<project_name>*.**sbx** is the Propel Builder project file that generates this design.
*<project_name>*1.sty	Include all the strategy parameters that are used in this design.
<project_name>.rdf	Radiant project file that includes all the information about this project.
impl_1	Contains all the log and report files during compilation, as well as the bitstream file to be loaded to the FPGA after compilation is completed.
sge	Contains all the software library files that are required for Propel IDK compilation.

3. Functional Description

This reference design is built based on the GHRD 3.0 design for CertusPro™-NX devices and then integrated the EtherCAT SubDevice IP from Beckhoff, as well as the Motor Control IP and Motor Encoder subsystem from the FOC Motor Control Reference Design into a single design.

3.1. Design Components

The reference design includes the following main components.

Table 3.1. Reference Design Components

Components	Descriptions
RISC-V RX soft processor TCM (Tightly Coupled Memory)	Runs the motor control application code with the Beckhoff software stack / U-PHY software stack from RT-Labs serving as the abstraction layer to communicate with the EtherCAT IP.
System Memory	An extra embedded memory component that can be utilized if the application code size required is larger than the TCM size. This memory component currently is not used by the design.
Phase-locked loop (PLL)	Generate the required clock frequencies for each of the components. There are a total of two PLLs used in the design: • System PLL: Generate the clocks for all the memory-mapped interfaces (AXI: 100 MHz, AHB and APB: 75 MHz) as well as the 25 MHz clock required for the EtherCAT SubDevice Controller IP. • Motor PLL: Generate the 200 MHz PWM clock, 20 MHz motor clock, and the 50 MHz clock for encoder blocks.
EtherCAT IP core for Lattice FPGA IP	An IP developed by Beckhoff that enables the EtherCAT communication on Lattice FPGAs. This reference design does not include the design files of the IP. To acquire the design files, you must contact Beckhoff. For more information, refer to the Beckhoff Information System website.
Motor Control IP Encoder Subsystem	IP components that are required for FOC Motor Control. For more details about the components, refer to the FOC Motor Control Reference Design User Guide (FPGA-RD-02325).
General Purpose Input/Output (GPIO) IP	Accepts various input status. For example, pdi_axi_irq_main, dc_sync, prom_loaded, device_id, and the output interrupt pin are connected to the RISC-V Rx core as one of the interrupt input. The current application code configures the assertion of pdi_axi_irq_main or dc_sync[1:0] inputs as the trigger point to assert the interrupt output pin, which will then trigger the interrupt routine in application code indicating that an event occurred on the EtherCAT IP core.
Timer Counter	To assert an interrupt signal to RISC V RX when timer counter reaches a pre-defined value which is configurable in application code. This can be used to support EtherCAT stack code for timing related operations and watchdog control.
UART Controller	To print diagnostic messages from the RISC V RX soft processor to a PC terminal for debugging and monitoring.
Octal SPI	An extra component that can be used to interface with Flash component externally or for storing virtual EEPROM contents. At power-on, the design performs an Erase–Program–Verify (EPV) self-test on a reserved flash memory region. The test erases one sector, programs a known data pattern, then reads back and verifies correctness to detect flash or bus reliability issues
Dual Boot Config IP	An extra component that can be used to load the secondary bootloader in the case when primary bootloader fails. This design is currently not utilizing this IP.
Unified Interconnect Advanced High-Performance Bus (AHB)-Lite Interconnect	Interconnect or bridge components that are required to connect all the memory-mapped interfaces to the RISC-V RX soft processor.

Components	Descriptions
Advanced Peripheral Bus (APB) Interconnect	
Advanced eXtensible Interface (AXI) to AHB-Lite Bridge	
AHB-Lite to APB Bridge	

Because the main component in this reference design is about the EtherCAT IP, for more information on other peripheral and motor control components, refer to the Golden System Reference Design and Demo User Guide v3.0 for CertusPro-NX Devices (FPGA- RD-02322) and the FOC Motor Control Reference Design User Guide (FPGA-RD-02325).

3.1.1. EtherCAT IP Core for Lattice FPGA

This IP is not included in this reference design due to licensing terms. To acquire this IP design files to proceed with compilation, you must contact Beckhoff and register yourself as an EtherCAT Technology Group (ETG) member to have a valid vendor ID. After getting the IP, you should be able to open **IPCore_Config.exe** file located in ipcore_config folder for generating the **.ipk** file to be imported to the Radiant software.

To apply the IP configuration used in this reference design, first copy the **EtherCAT_CertusProNX_MCB_MII_AXI.ecpxml** file located in the supp folder of the reference design ZIP file into your PropelIPLocal directory, which is typically located under your PC user directory (for example, *C:\Users\<your username>\PropelIPLocal*). Next, open **IPCore_Config.exe** from the Beckhoff IP directory and browse to this **.ecpxml** file. Click Continue to proceed to the IP configuration pages.

On the Identification tab, as shown below, you can change the Product ID to your own after acquiring the IP from Beckhoff, as well as modify the product description text if desired.

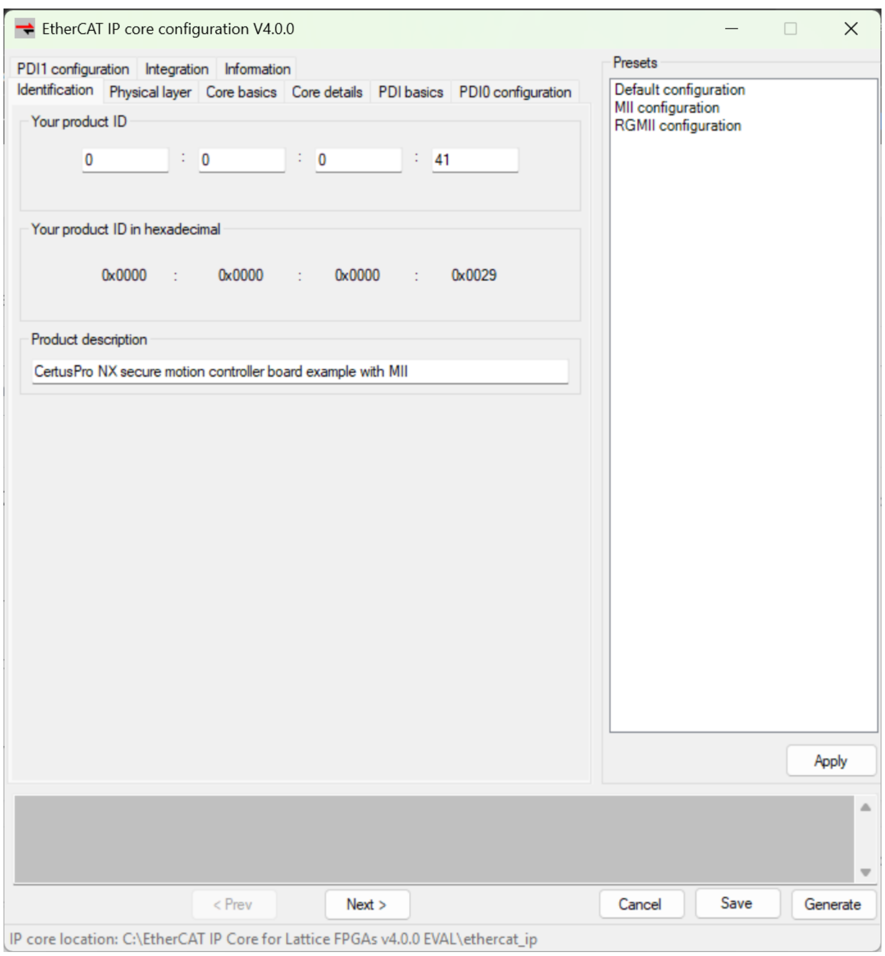

Figure 3.1. EtherCAT IP Configuration Page

The EtherCAT IP is configured for an MII interface and communicates with a host PC via the onboard ADIN1200 MII PHY on Port 0. For more information on ADIN1200, refer to the ADIN1200 web page on the Analog Devices website.
The selected EtherCAT Process Data Interface (PDI) is the AXI4 interface, which can be directly connected to an AXI interconnect to communicate with the rest of the embedded system components, including the RISC-V RX core, thereby forming a complete system. In the PDI0 configuration tab, the on-chip bus clock (CLK) is set to 10,000 ps, which corresponds to a frequency of 100 MHz. This configuration ensures that the AXI interface operates in the same clock domain as the RISC-V RX core.

The table below summarizes the EtherCAT IP core configuration in the reference design.

Table 3.2. EtherCAT IP Core Configuration

Parameter Name	Value
Physical Layer	
Number of communication ports	2
Selected communication type port	MII
Optical link (FX)	Disabled
MI link detection and configuration	Enabled
PHY management interface	Enabled
Separate PHY management interfaces for each port	Disabled
User initialization of PHY registers	Disabled
Export PHY address as signals	Enabled
Independent PHY addresses	Enabled
Enhanced link detection	Disabled
Core Basics	
Number of FMMU	4
Number of SyncManager	4
SyncManager deactivation delay for PDI	Disabled
SyncManager sequential mode for ECAT	Enabled
Process Data RAM	4k byte
Private PDI RAM	0k byte
Distributed clocks enabled	Enabled
DC SyncSignals	Sync0/1
DC LatchSignals	Sync0/1
Distributed clocks width	32 bits
Cyclic pulse length	100
SyncManager event times	Enabled
DC speed counter direct control	Disabled
Map SyncSignals[*] to AL event reg	Disabled
Core Details	
Read/write offset	Disabled
Write protection	Disabled
Extended RX error counter and RX error code	Enabled
EEPROM emulation by PDI	Enabled
RESET slave by ECAT/PDI	Disabled
RUN_LED	Enabled
Extended RUN/ERR LED features	Enabled
LED test	Enabled
PDI Basics	
Device emulation	Disabled
AL status code register	Enabled
Extended watchdog	Enabled
Watchdog counter	Enabled
AL event mask register	Enabled
PDI SyncManager/interrupt acknowledge by write access	Disabled
PDI user mode	Disabled
Enable PDI1	Disabled
Number of GPIOs	None
Internal PDI CLK	25 MHz

Parameter Name	Value
Internal PDI data bus width	32 bit
PDI0 Configuration	
Selected PDI	AXI4 Slave
On-chip bus clock is asynchronous to CLK25	Enabled
On-chip bus CLK	10000 ps
External data bus width	32 bit
ID width	6
Integration	
Disable RAM initialization	Disabled

For more detailed information and explanations of this IP core, refer to the following sections of the Beckhoff EtherCAT slave controller (ESC) documentation that can be found on the Beckhoff website:

- EtherCAT Technology (Section I)
- EtherCAT Registers (Section II)
- EtherCAT IP Core for Lattice FPGAs (Section III)

3.1.1.1. Interfacing with EtherCAT SubDevice Stack Code (SSC) ET9300

The SSC is a reference C implementation for an EtherCAT SubDevice provided by Beckhoff that runs on a RISC-V CPU. The SSC handles the full complexity of the EtherCAT protocol and abstracts it from the application layer, preventing the application code from becoming cluttered. You only need to integrate your application-specific logic by implementing the defined callback functions in the code.

There are four main layers in this reference design software.

Application Layer (User Implement) APPL_Application() - User main application APPL_InputMapping() - Application Data to Input of EtherCAT Tx APPL_OutputMapping() - Output Data from EtherCAT Rx to application APPL_StartInputHandler() - Configure Hardware APPL_StartOutputHandler() - Enable Outputs APPL_StopOUtputHandler() - Disable Outputs … (a few more)
SSC Layer (Beckhoff) Handles EtherCAT State Machine, PDO cycling, mailbox, CoE/SDO, object dictionary framework
Hardware Abstraction (User Implement) HW_Init() - Initialize ESC hardware HW_EscRead() / HW_EscWrite() - read/write ESC registers HW_GetALEventRegister() - read AL event register EnableESCInterrupt() / DisableESCInterrupt()
Platform Drivers (Lattice specific hardware drivers) ethercat_irq.c - ESC interrupt dispatch ethercat_timer.c - system timer ethercat_platform_config.h - addresses, GPIO pin / timer configuration

Figure 3.2. Main Layers of the Reference Design

As shown in the layered diagram above, the SSC invokes the application code through a set of *APPL_** callback functions at the application layer. These callbacks are customizable and can be implemented according to specific use-case requirements.
As you can see from the layered diagram above, SSC will call the application code through a set of "*APPL_**" callback functions at the Application layer, which the code are customizable according to the use case scenarios.

The diagram below illustrates the EtherCAT data flow in a simplified form. For a more detailed explanation of the SSC, refer to the *Application Note ET9300 (EtherCAT Slave Stack Code)* on the *ET9300 EtherCAT Slave Stack Code* page of the Beckhoff website, available under the Documentation and Download tab.

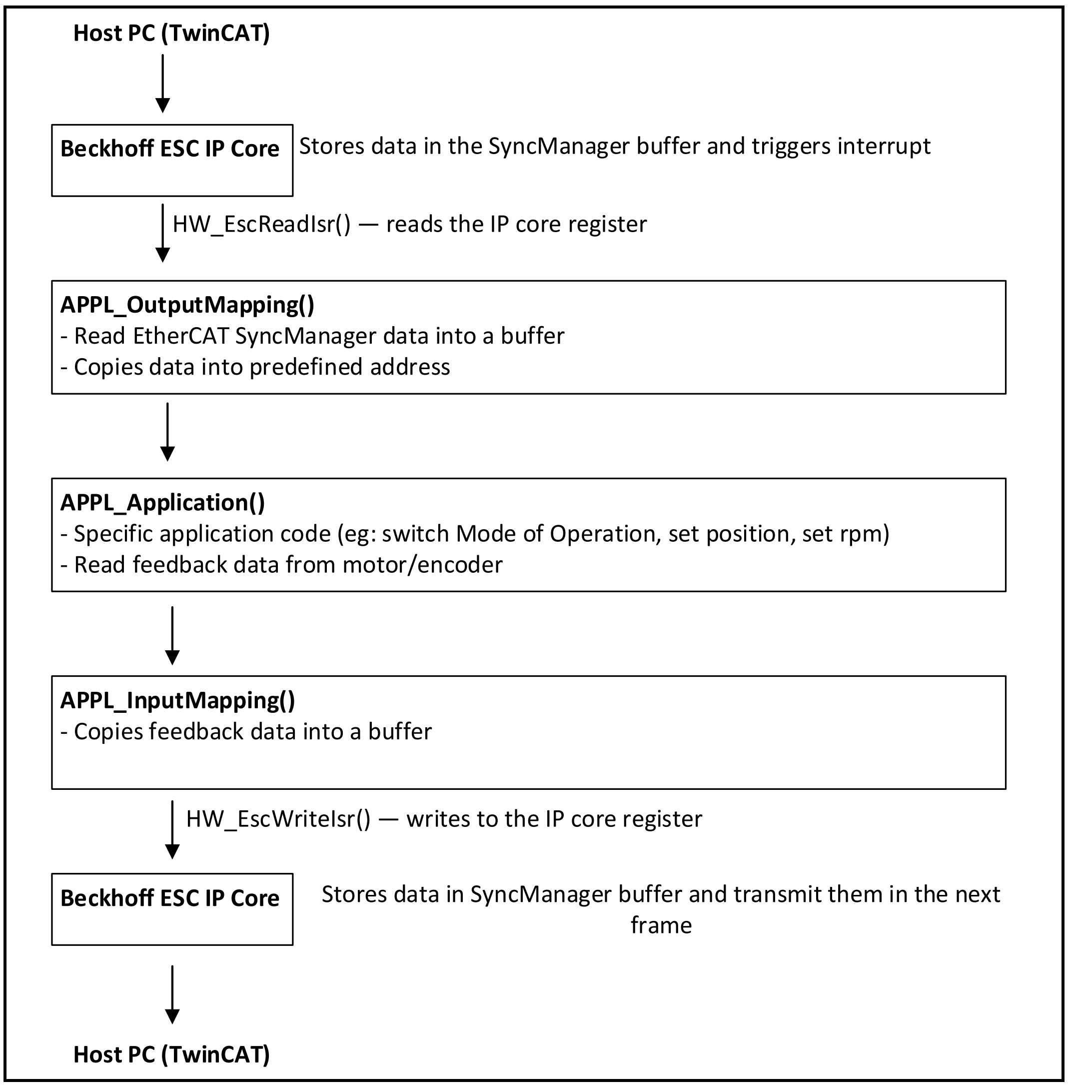

Figure 3.3. EtherCAT Data Flow

The following steps describe the EtherCAT data flow:

1. At every cyclic cycle, the EtherCAT MainDevice sends some command data to the SubDevice receive Process Data Object (RxPDO) buffer.
2. This action triggers an interrupt to the RISC-V core. Then, the SSC reads the data from the EtherCAT IP core and calls APPL_OutputMapping() to copy the data into the application layer.
3. APPL_Application() processes the variables (for example, writing to motor control IP registers).

4. APPL_InputMapping() copies feedback data from the motor control IP (such as status, position, and other parameters) and puts them into the transmit Process Data Object (TxPDO) buffer.
5. The SSC writes the TxPDO buffer data back to the EtherCAT IP core so that the data can be forwarded onto the network in the next EtherCAT frame.

3.2. Clocking Scheme

The following figure shows the clocking scheme of this reference design. The system input clock is a 25 MHz clock sourced from the Ethernet PHY. This clock is routed to two main PLLs—the System PLL and the Motor PLL—which generate the required clocks for each functional component in the system. Additional details are provided in the table below.

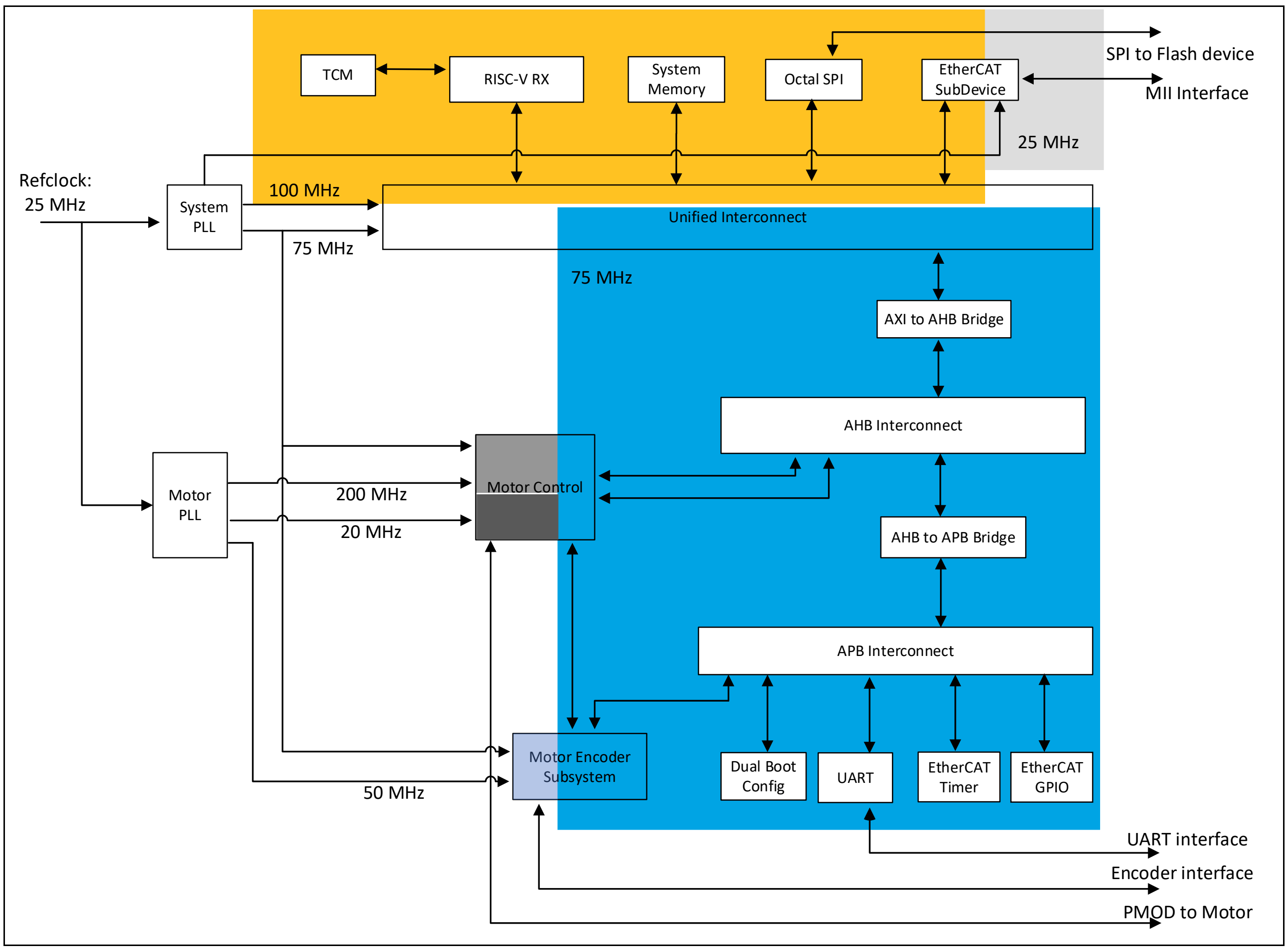

Figure 3.4. Clocking Scheme

Table 3.3. PLL Clock Signals

Signal Name	Clock Frequency (MHz)	Description
syspll_refclk_i	25	Input reference clock from the onboard Ethernet MII PHY chip (ADIN1200).
system_pll/clkop_o	75	Operating clock frequency on the AHB and APB memory-mapped components. This clock is set per the Fmax achievable by the Motor Control IP.
system_pll/clkos_o	100	Operating clock frequency on the AXI memory-mapped components, as well as the RISC-V Rx core and the TCM IP. This clock is set per the Fmax achievable by the RISC-V Rx IP.

Signal Name	Clock Frequency (MHz)	Description
system_pll/clkos2_o	25	Input clock required by EtherCAT SubDevice IP for its MII data processing. This frequency is fixed and not modifiable.
motor_pll/clkop_o	200	For pwm_clk_i to achieve 5-ns PWM resolution. This frequency is fixed and not modifiable.
motor_pll/clkos_o	20	For motor_clk_i clock domain required by Motor Control IP. This frequency is fixed and not modifiable.
motor_pll/clkos2_o	50	For clk_50_i clock domain required by encoder block. This frequency is fixed and not modifiable.

4. Running the Reference Design

4.1. Hardware Setup

The following lists the three main hardware that are required to run the reference design:

- 1—Secure Connected Motion Control Platform rev B with three PMOD connectors. For more information, refer to the Secure Connected Motion Control Platform web page.
- 2—Lattice Motor Control Solution Kit. For more information, refer to the FOC Motor Control Reference Design User Guide (FPGA-RD-02325).
- 3—Windows® 10 host PC with supported network controller listed by Beckhoff for measuring real-time performance.
  The TwinCAT Automation software v3.1 build 4024 must be installed to run the hardware.
  For more information on supported network controllers, refer to the *TwinCAT 3 Supported Network Controllers* web page on the Beckhoff website (TwinCAT 3 -> Product Overview -> System requirements).

Note: An industrial PC is recommended to get a deterministic cycle time.

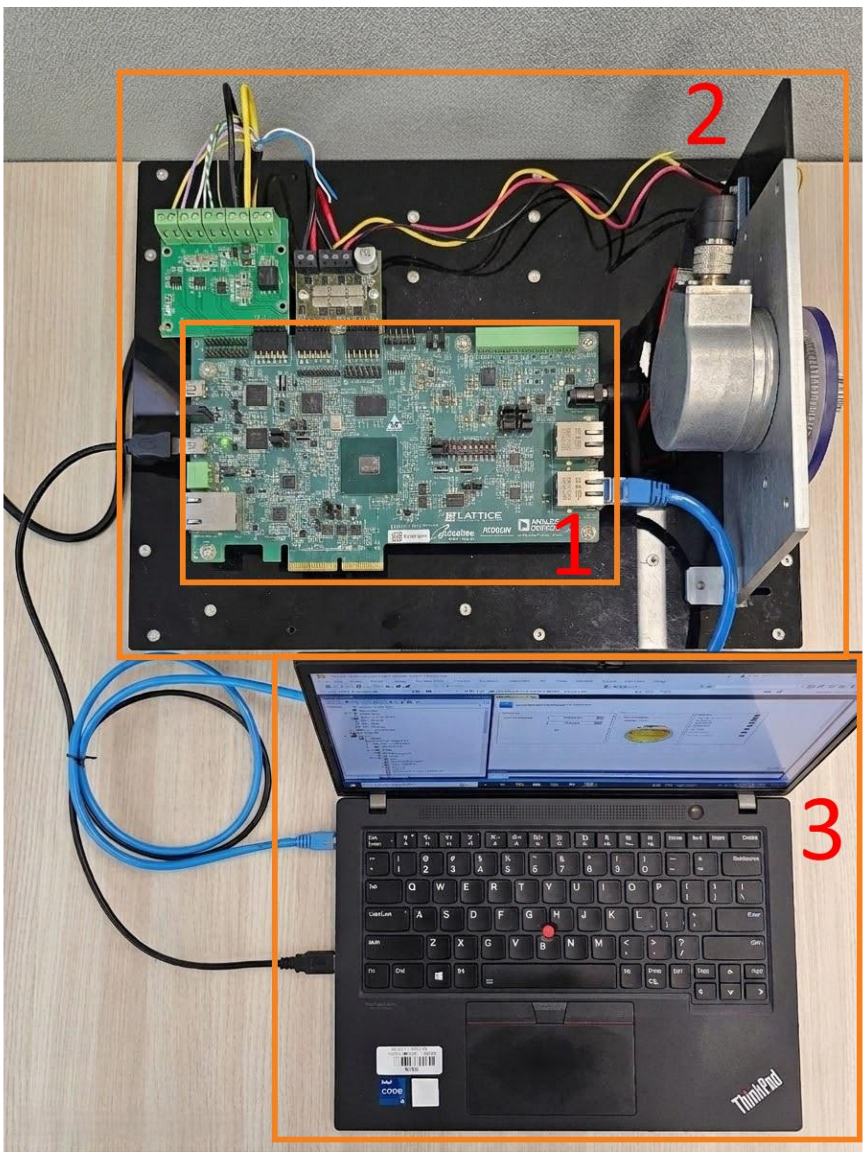

Figure 4.1. Hardware Setup

The following figure shows the connection between the Motion Control Platform board with the other hardware components:

- 1—Encoder board, which is connected to the onboard PMOD2 connector.
- 2—Trenz driver board, which is connected to the onboard PMOD0 and PMOD1 connector.
- 3—12V DC power input.
- 4—RJ-45 cable connection between the Motion Control Platform board and the host PC.

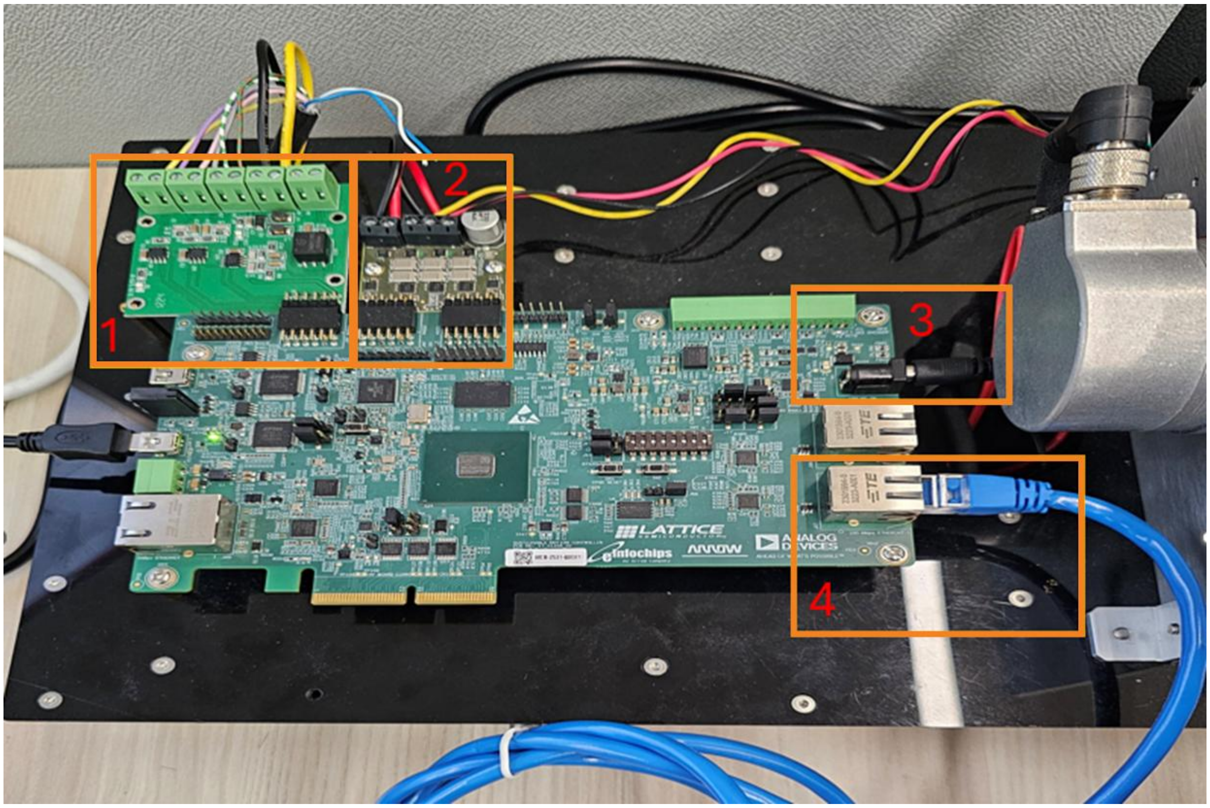

Figure 4.2. Motion Control Platform Board with Other Hardware Components

4.2. Compiling the SoC Design and Application Code with Radiant and Propel Software

Copy the **Latticesemi_LFCPNX_EtherCAT_1-Axis_Motor_control_<version></version>** folder from the downloaded .zip file into your **PropelTemplateLocal** folder, which is typically located in your user directory—*C:\Users\<your username>\PropelTemplateLocal*), to proceed with the following steps.

4.2.1. Generating EtherCAT Stack Code

To build the reference design, you must obtain the EtherCAT SubDevice Stack Code (SSC) tool to generate the required stack code files.

To generate the EtherCAT stack code, follow these steps:

1. Download and install the EtherCAT SubDevice Stack Code (SSC) ET9300 version 5.13 from the *EtherCAT Slave Stack Code (SSC) ET9300* web page on the EtherCAT Technology Group (ETG) website.
2. Launch the SSC tool to generate the stack code files.

For more information about the SSC tool, refer to the *Application Note ET9300 (EtherCAT Slave Stack Code)* on the *ET9300 EtherCAT Slave Stack Code* page of the Beckhoff website.

4.2.1.1. Import Configuration Files

1. From the menu bar, click **Tools > Options**.

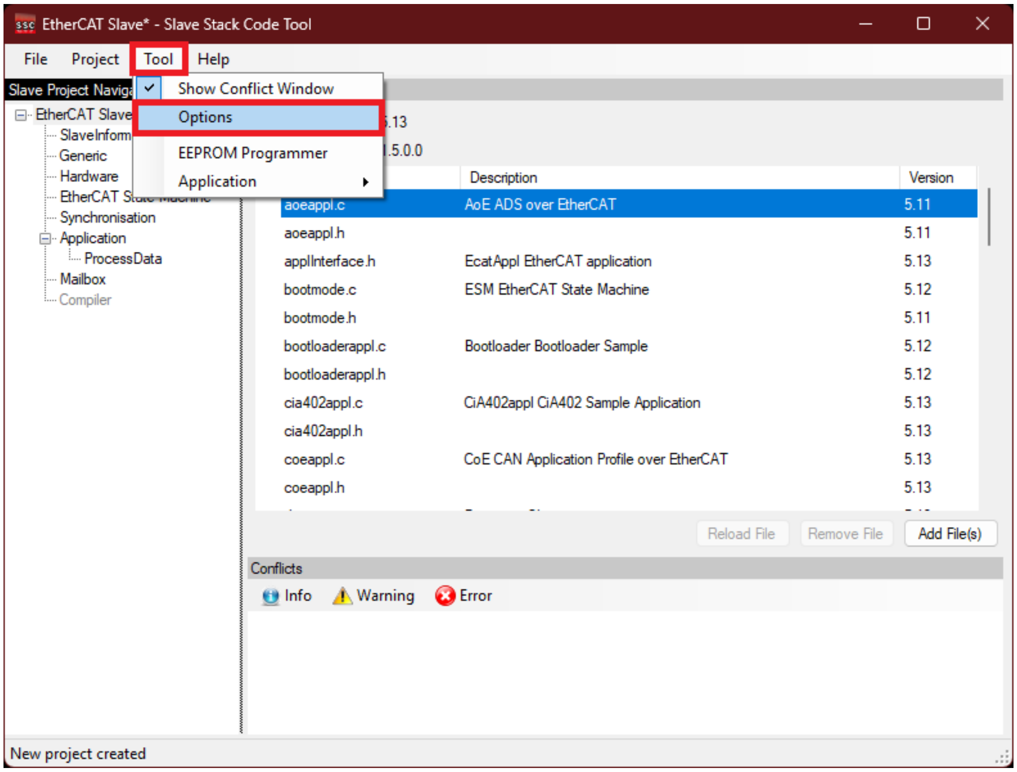

Figure 4.3. SSC Tool Options

2. The SubDevice Stack Code Tool Options window appears. Click the Configurations tab followed by the + icon.

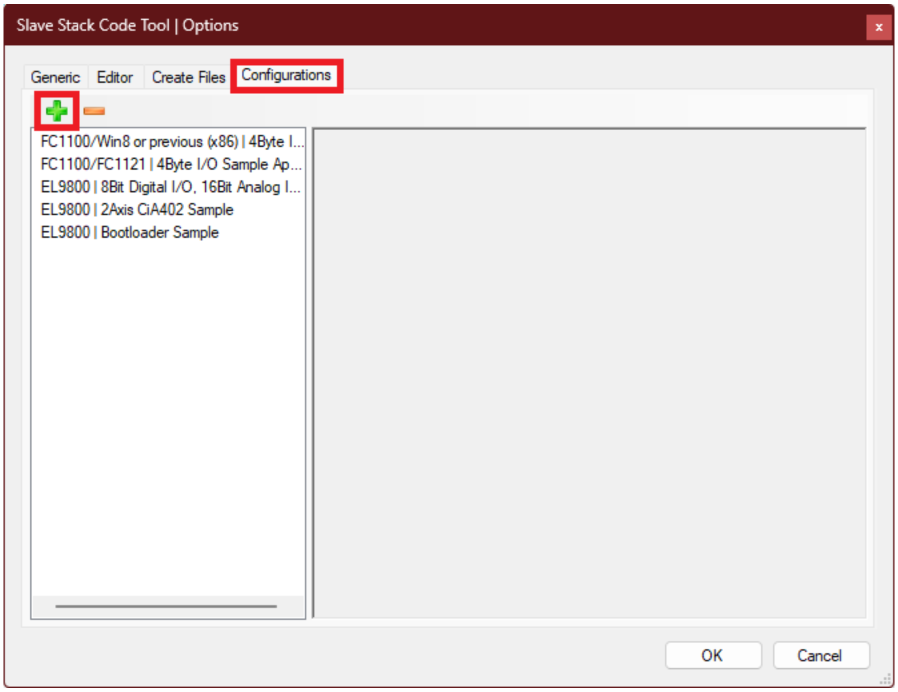

Figure 4.4. SSC Tool Configurations Import

3. Navigate to the EtherCAT Stack folder in the reference design and select the **Lattice_CertusPro NX_MCB_FOC_SSC_Config.xml** file. Click **Open** to import the configurations.
4. The configuration list is updated with the imported configuration as shown in the figure below.

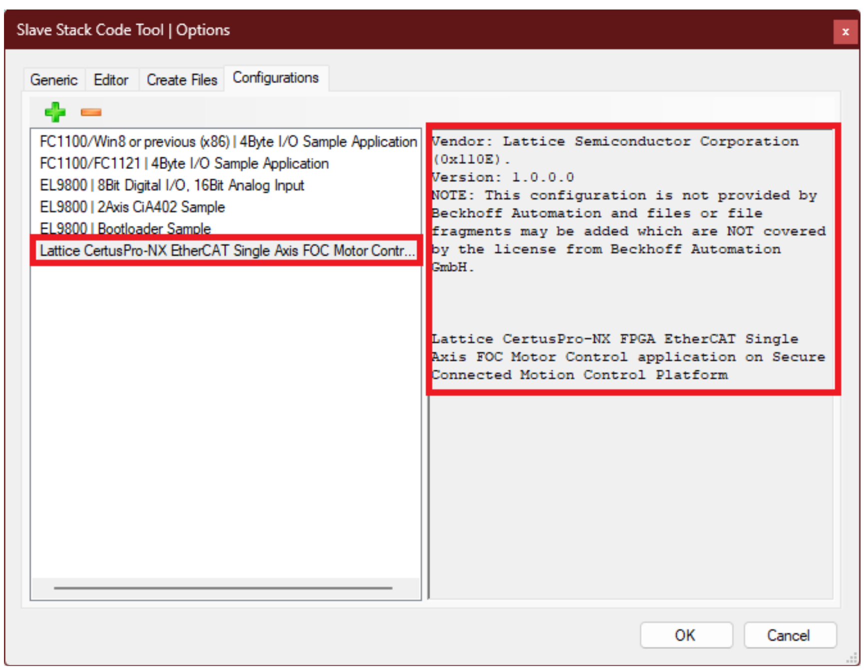

Figure 4.5. New Configuration Added to the List

5. Click **OK** to go back to the SSC tool.

4.2.1.2. Create new SSC Project

1. From the menu bar, click **File > New** to launch the Slave Stack Code Tool New Project window.

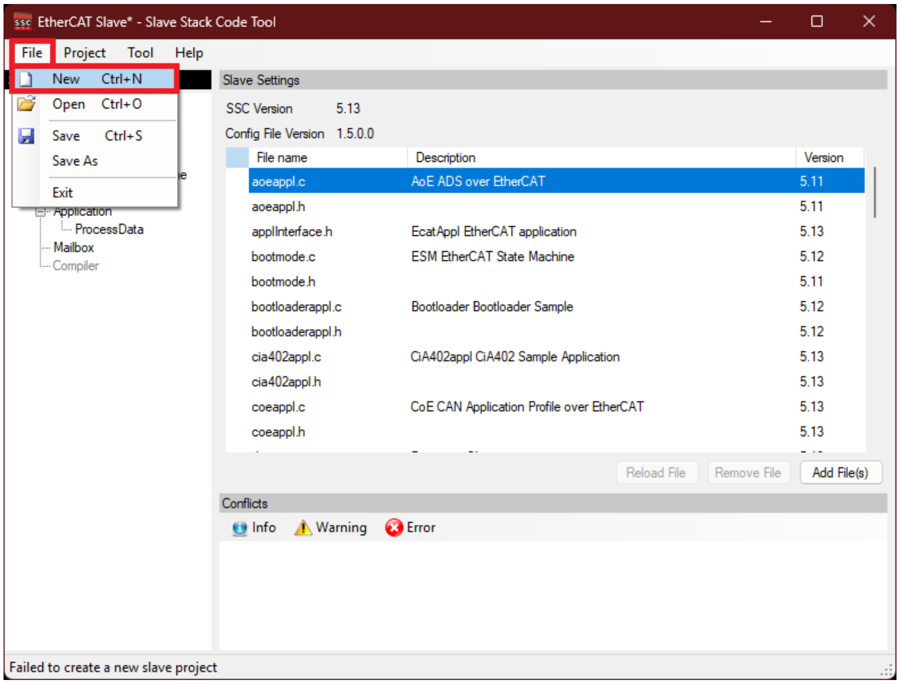

Figure 4.6. Create New Project in SSC Tool

2. Turn on the **Custom** option and select **Lattice CertusPro-NX EtherCAT Single Axis FOC Motor Control** from the drop-down list. Click **OK**.

Figure 4.7. Project Configuration Selection Window

3. Click **Yes** when prompted to proceed.

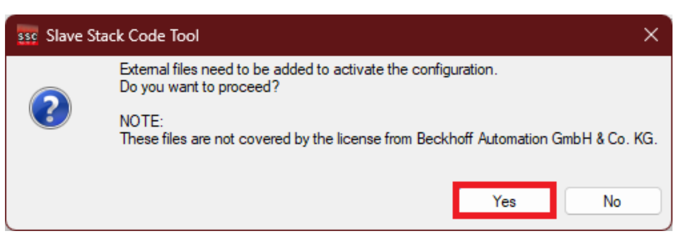

Figure 4.8. Custom Project Configuration Prompt

4.2.1.3. Import Application Definition File

1. From the **Tool** menu, select **Application > Import**.

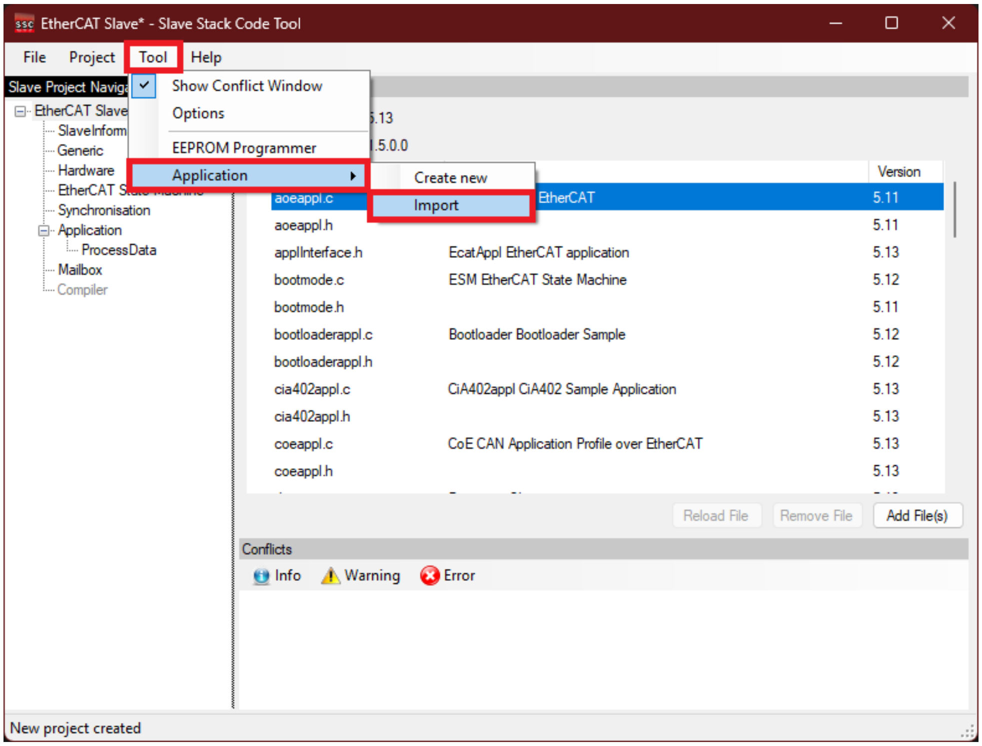

Figure 4.9. Import Application Definition File

2. Navigate to the EtherCAT Stack folder in the reference design and select the **CPNX_MCB_ECAT_FOC_MotorControl.xlsx**. Click **Open** to import.
3. Keep the application name as it is and click **OK** to proceed.

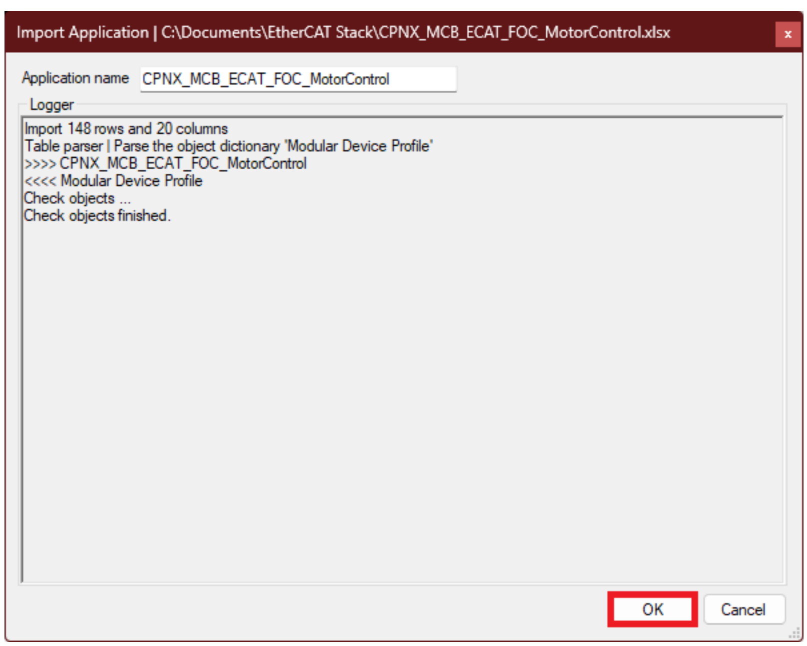

Figure 4.10. Result of Importing the Application Definition File

4. (Optional) In the previous step, the DEVICE_NAME is overwritten by the Application name. Modify DEVICE_NAME in the SubDevice Information page back to CertusPro-NX Single Axis FOC Motor Control.

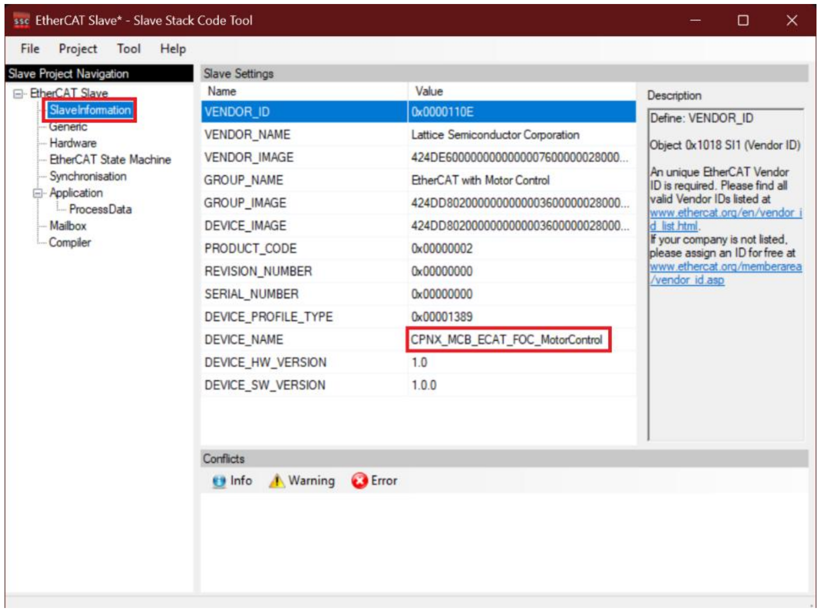

Figure 4.11. SubDevice Information Page

5. Replace the existing values with your own **VENDOR_ID**, **VENDOR_NAME**, and **PRODUCT_CODE**.
6. Click **File** -> **Save As** to save the project.

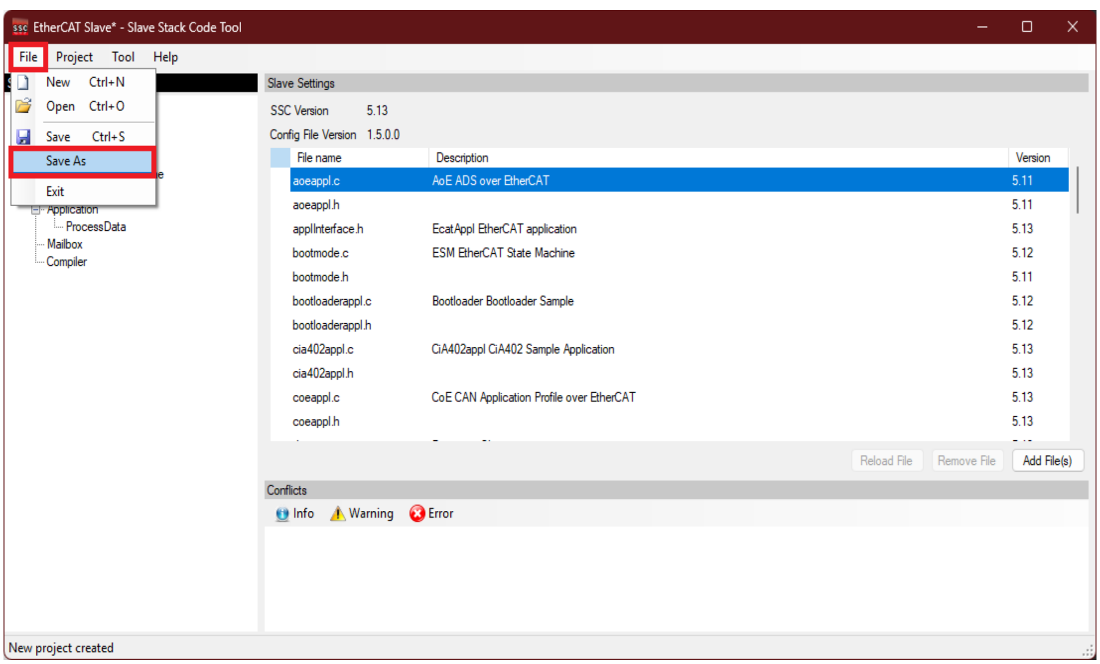

Figure 4.12. Saving the SSC Tool Project

4.2.1.4. Generate Stack Code and ESI File

1. From the menu bar, click **Project > Create new SubDevice Files**.

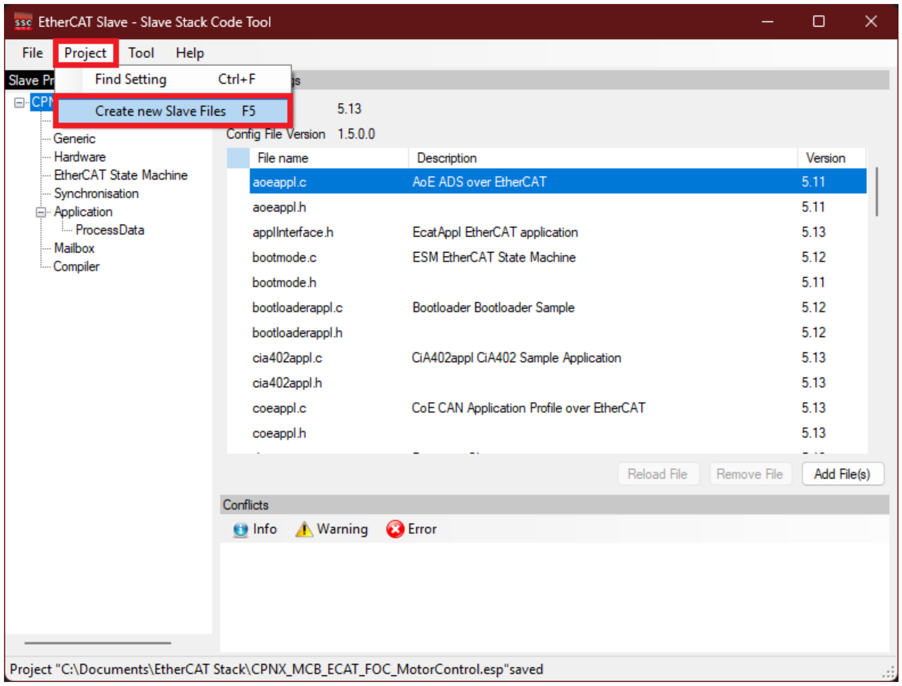

Figure 4.13. Create New SubDevice Files

2. Change the output file path for **Source Folder** and **ESI File** if required. By default, these paths are generated in the same directory as the SSC project. Click **Start** to generate the source code and ESI file.

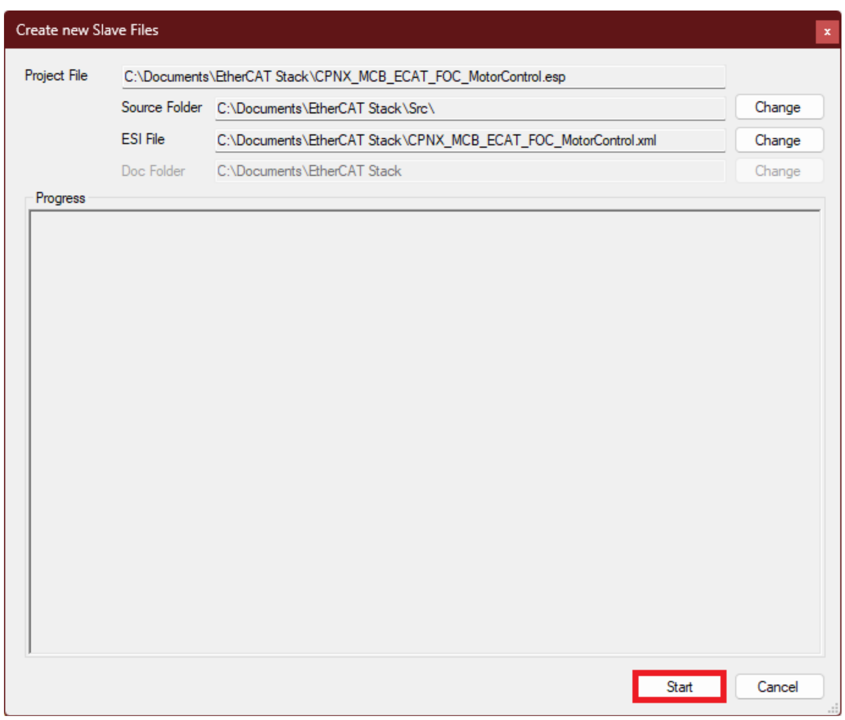

Figure 4.14. SSC Tool SubDevice Files Creation Window

3. Copy the entire Src folder generated and paste it into the application folder within the reference design directory, which is located in the PropelTemplateLocal folder. Rename the Src folder to the ethercat folder as shown in the figure below.

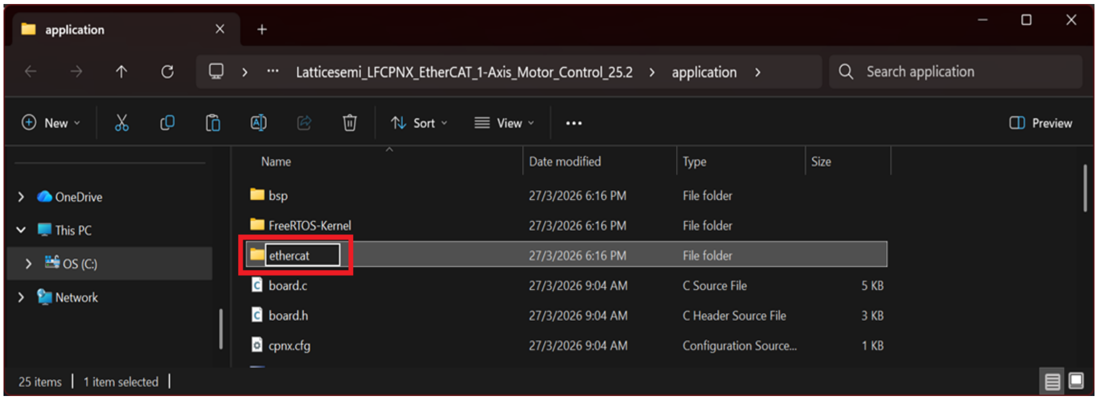

Figure 4.15. Adding SSC SubDevice Files to the Propel Template Folder

4.2.1.5. Applying Patch to Generated Stack Code

1. Go to Git Bash if you are using Windows, or go directly to a Terminal if you are using Linux/Unix operating system.
2. Use the *cd* command to navigate to the *ethercat* folder, then run the following command to convert the line endings of all previously copied files to LF (Linux/Unix line-ending format).

   For example, *find . -name "*.[ch]" | xargs dos2unix*.
3. Apply the patch by running the following command:
   patch -p1 < ../lscc_ecat_ssc.patch.

4.2.2. Generate Design

1. Refer to the EtherCAT IP Core for Lattice FPGA section to generate the **.ipk** file prior to the steps below.
2. Launch the Propel Builder software.
3. Go to **File -> New SoC Design**.
4. Input your project name and the saved location for the design.
5. At the **Select Template** page, select **LFCPNX_EtherCAT_1-Axis-Motor_Control** and click **Next**.
6. On the **Select Device** page, click **Next** again. Only one option is available – **Secure Connected Motion Control Platform**.
7. On the next page, click **Finish** to bring up the layout of this reference design.
8. When the design is initially loaded, the connections may not be well organized. Right-click on any empty area and select **Relayout** to tidy up the connections.
9. Save the design, then click on the **Run Propel** button at the top.

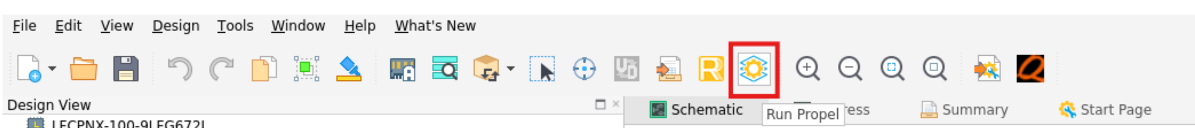

Figure 4.16. Run the Propel Button on the Propel Builder Toolbar

10. Wait for the Propel software to launch, and for the C/C++ Project window to appear. Enter the desired project name, and click **Next.**

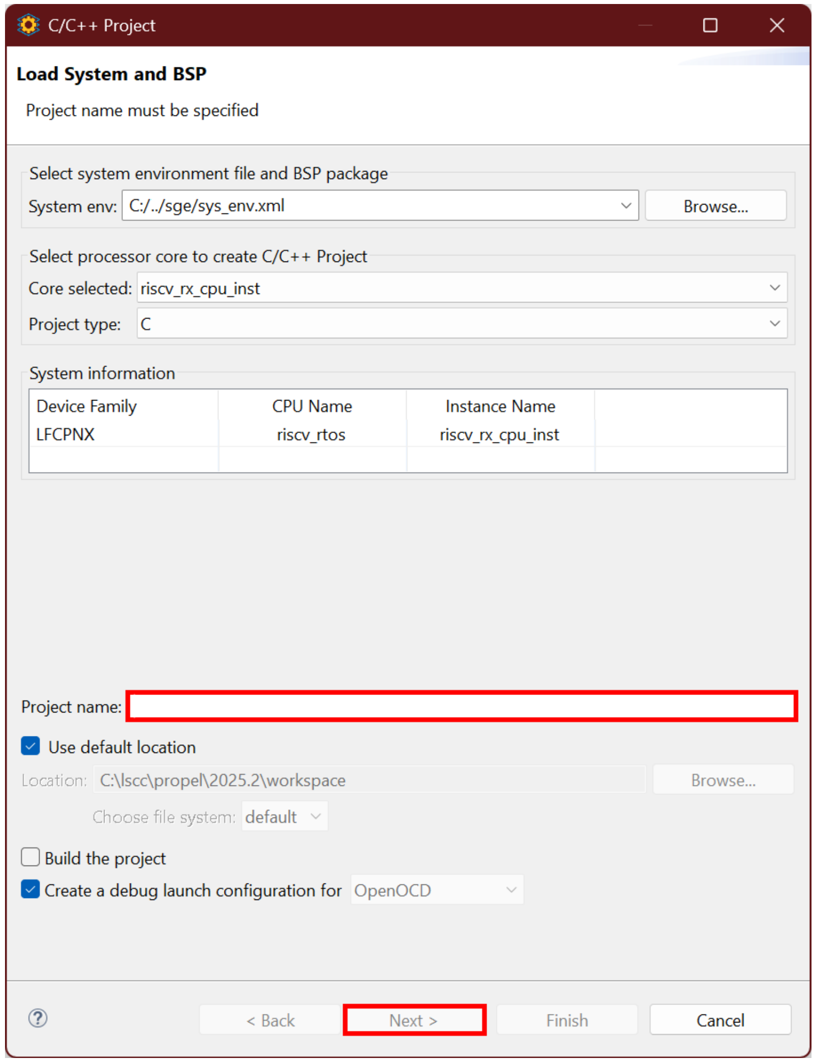

Figure 4.17. New C/C++ Project Creation Window in Propel

11. In the **C/C++ Compiler** tab, set the **Optimization Level** to **Optimize more (-O2)** and click **Finish**.

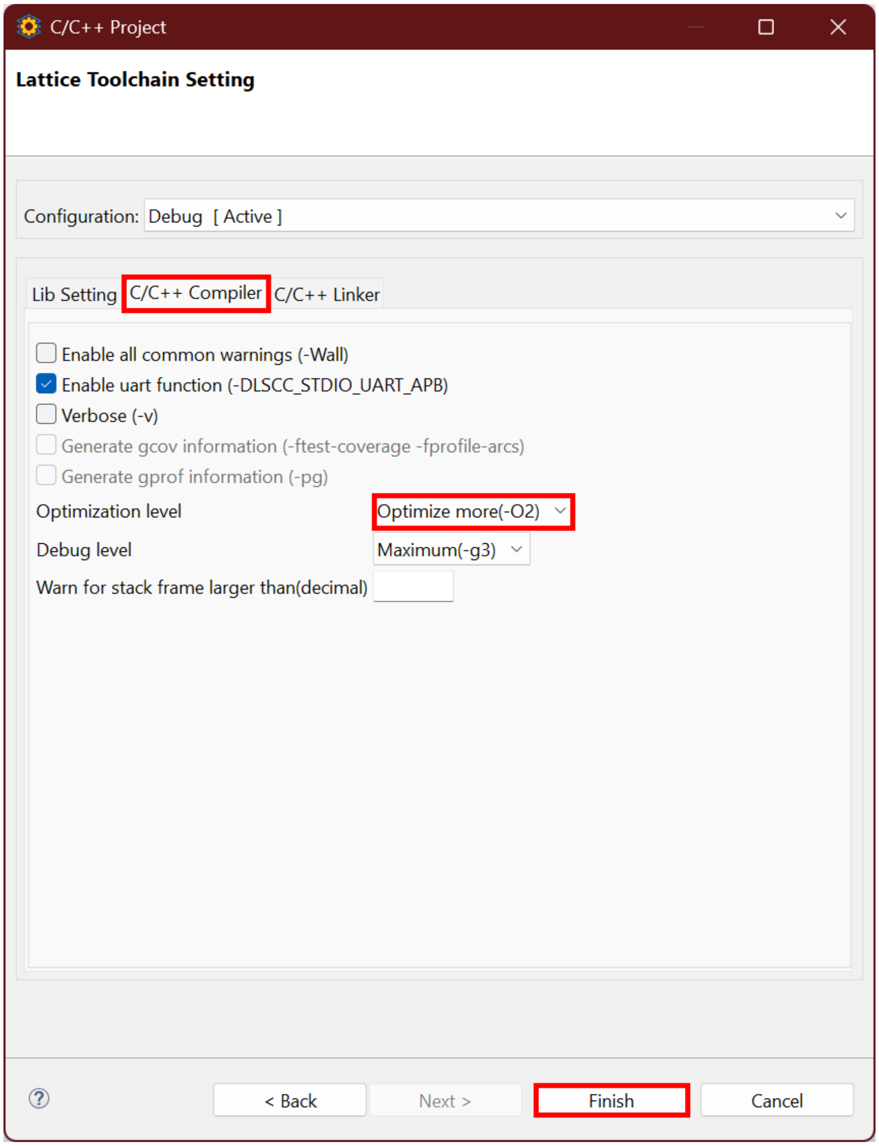

Figure 4.18. New Project C/C++ Compiler Settings in Propel

12. Select the project and click on the **Build** button. Wait for the compilation to complete and the **.mem** file will be generated in the **Debug** folder.

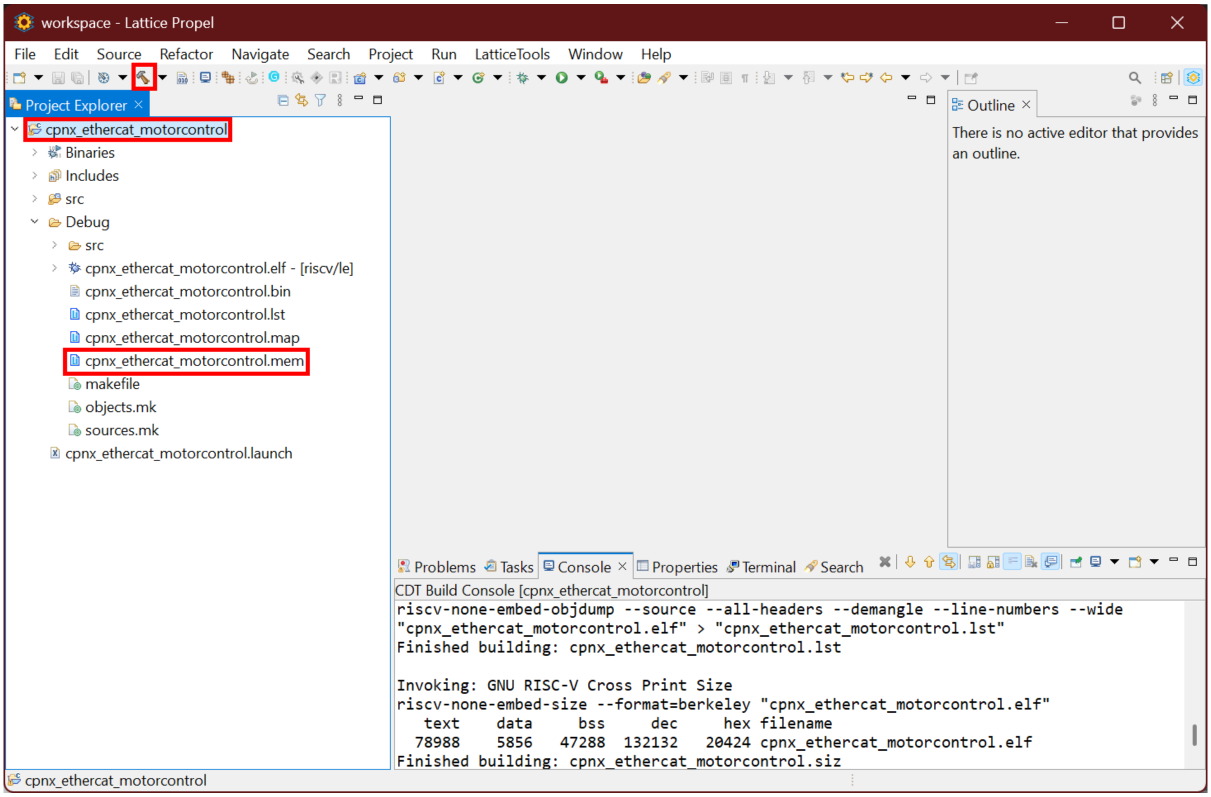

Figure 4.19. Propel Project Build Button and Project Memory Initialization File

13. After the **.mem** file is generated, return to the Propel Builder, right-click on the TCM component IP, and select **Reconfig**.
14. Change the **Memory Initialization** property value to **Memory File,** as shown in the screenshot below. Then, browse to the newly generated **.mem** file and select it in the **Memory File** property. Finally, click the **Generate** button located at the bottom right.

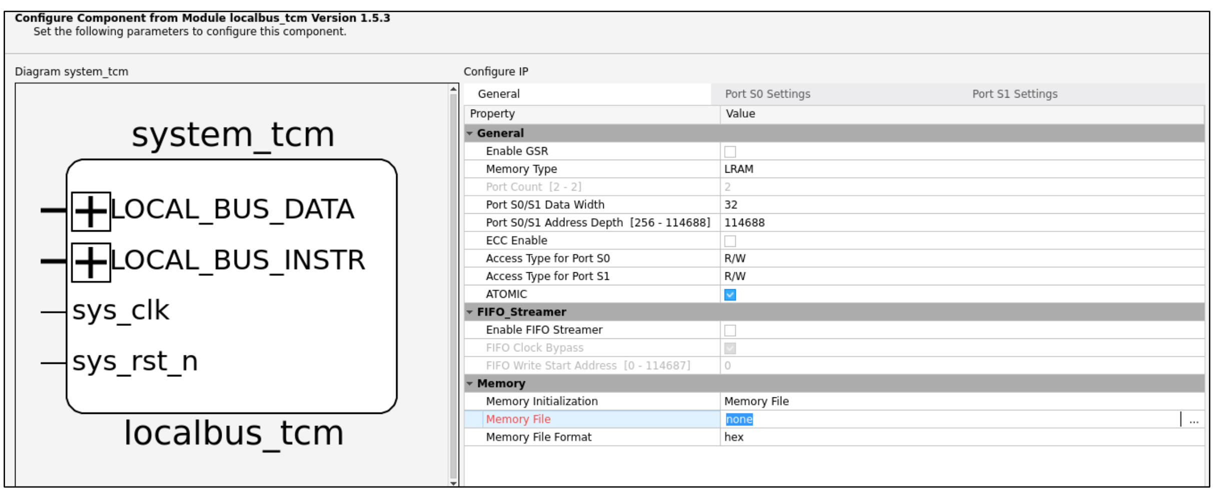

Figure 4.20. Memory Initialization File Selection in Module/IP Wizard

15. Save the design and click on the **Run Radiant** button located on the top menu bar.

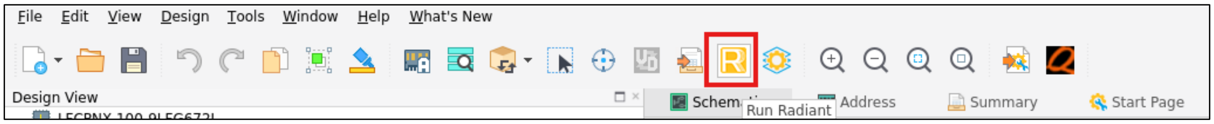

Figure 4.21. Run Radiant Button in the Propel Builder Toolbar

16. After the design is loaded successfully in the Radiant software, click on the **Play** icon to start the compilation process.
17. A **.bit** file will be generated in the **impl_1** folder after the compilation is completed.

4.3. Setting Up the Host PC (Windows 10 PC)

4.3.1. Ethernet Adapter Driver Installation

1. Launch the TwinCAT XAE Shell from the Windows Start menu
2. Click **TwinCAT** in the menu bar and select **Show Realtime Compatible Devices…**.

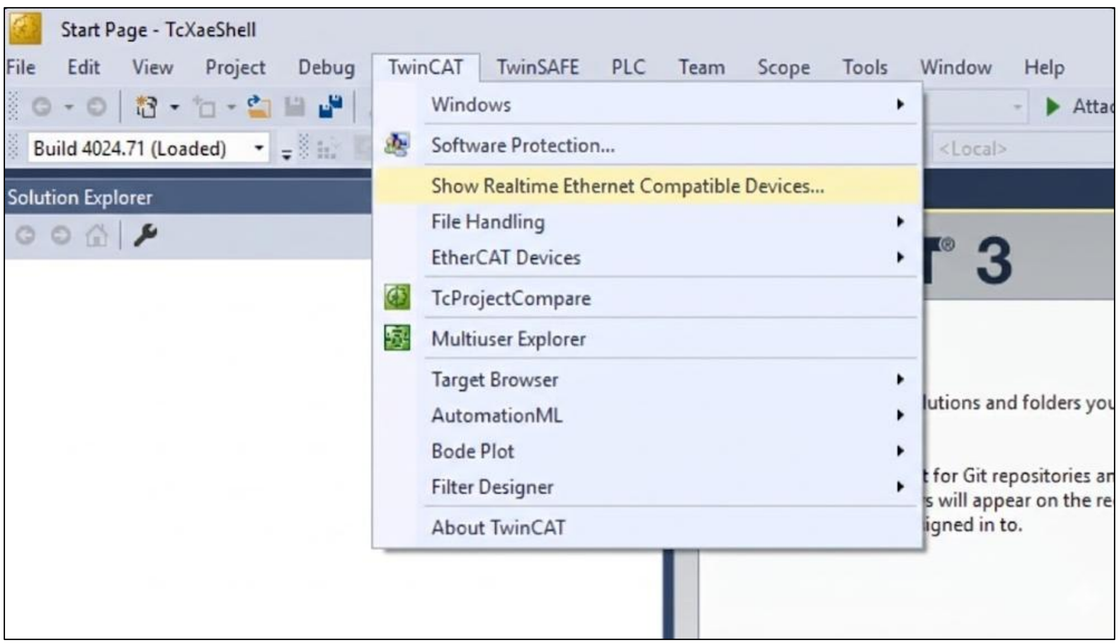

Figure 4.22. TwinCAT: Show Real-Time Ethernet-Compatible Devices

3. A new window appears as shown in the figure below.

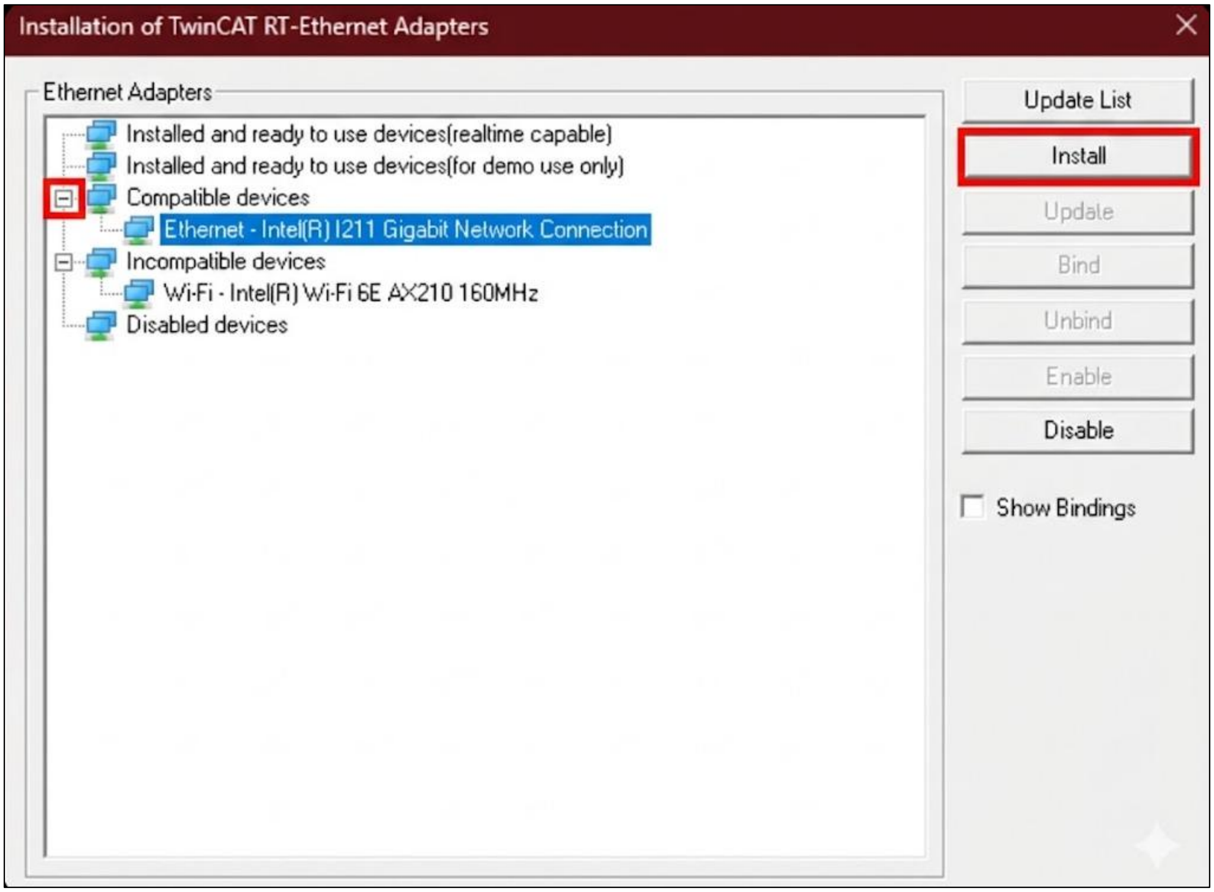

Figure 4.23. Network Adapter List in the System and Driver Installation

4. Select the network adapter for EtherCAT communication. It is recommended to choose a device listed under **Compatible devices**. If your PC is equipped with a supported network controller, as described in the Hardware Setup section, the adapter should appear in the compatible device list.
5. Click the **Install** button, then restart your PC after the installation process is complete.

4.3.2. Disable Hyper-V in Windows

Hyper-V is a Windows hardware virtualization technology that allows you to create and run virtual machines on Windows. However, this tool fundamentally conflicts with TwinCAT at the runtime level because both systems attempt to control and manage the computer hardware resources.

TwinCAT requires deterministic and real-time performance and does not allow interruptions during cyclic task execution. However, enabling Hyper-V inherently disrupts the operating system control flow.

To disable Hyper-V, follow these steps:

1. Launch the **Control Panel** and click the **Programs** category. Under **Programs and Features**, select **Turn Windows features on or off**. Alternatively, you can search for **Turn Windows features on or off** using the Windows search bar.

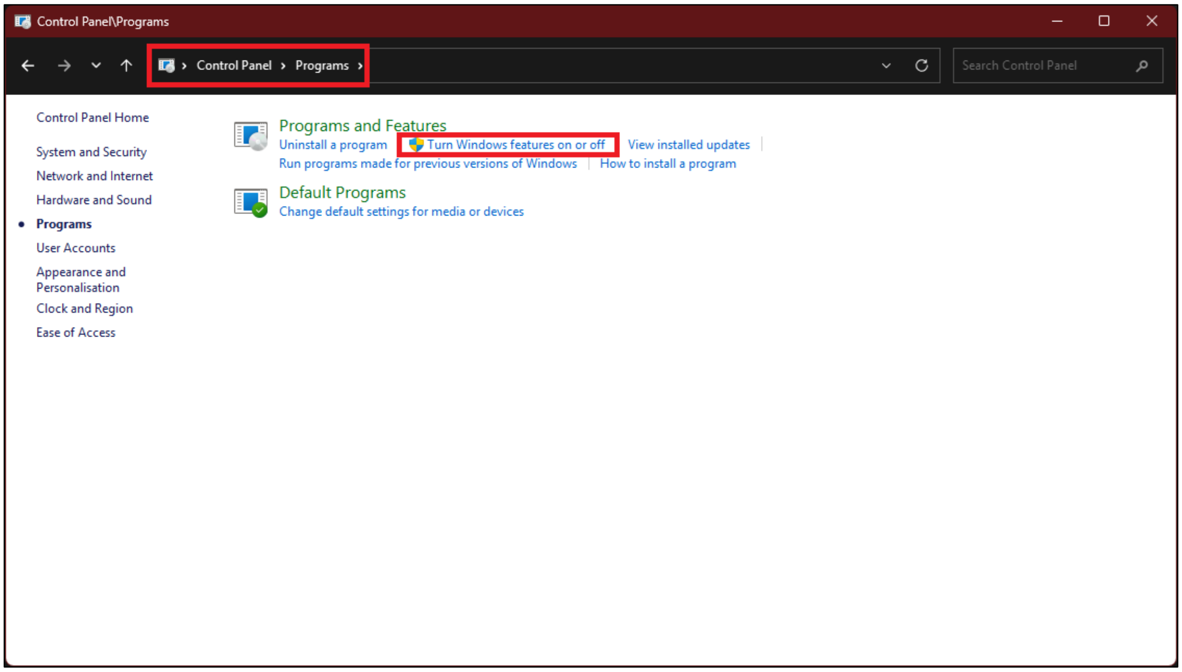

Figure 4.24. Turn Windows Features On or Off through the Control Panel

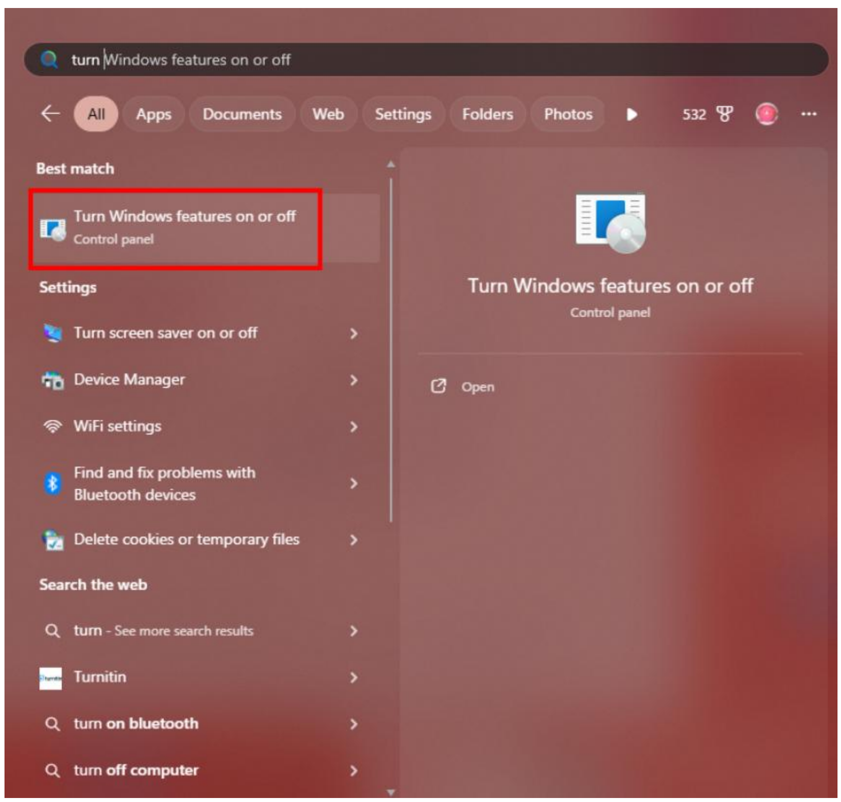

Figure 4.25. Turn Windows Features On or Off via Control Panel Using Windows Search

2. Make sure all the check boxes highlighted below are unchecked.

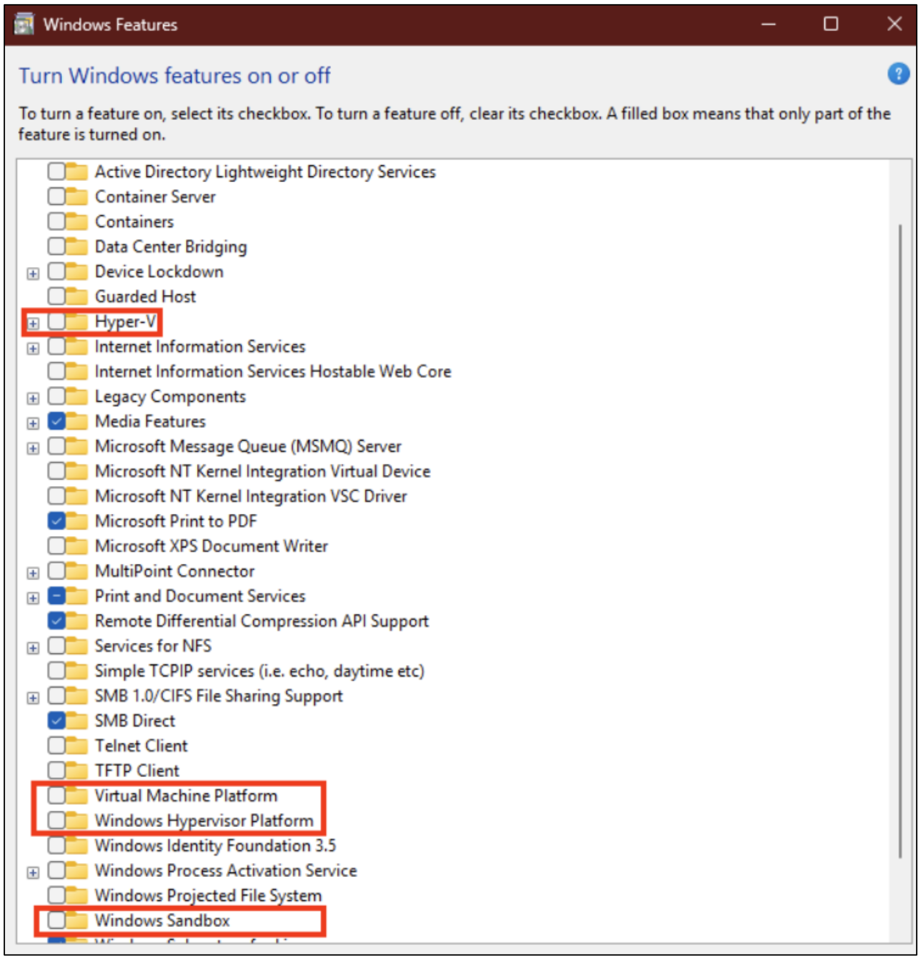

Figure 4.26. Windows Feature List

3. Click **OK** and restart your PC.

4.3.3. TwinCAT win8settick.bat Batch Script

The batch script, which is typically located in the *C:\TwinCAT\3.1\System* folder directory, is a specialized configuration utility provided by Beckhoff to prepare the Windows environment for TwinCAT 3 Runtime (XAR). The script modifies the Windows boot configuration to disable dynamic ticks and synthetic timers, which is required by TwinCAT to achieve deterministic communication.

To execute the script,

1. Locate the **win8settick.bat** file in your TwinCAT installation directory, which is typically located under *C:\TwinCAT\3.1\System*.
2. Right-click the **.bat** file and select **Run as administrator** to execute the script.
3. Restart your PC after the process is complete.

4.3.4. Importing ESI File into TwinCAT

An EtherCAT SubDevice Information (ESI) file is an XML file that describes an EtherCAT SubDevice to the EtherCAT MainDevice. It contains device data such as identification information, communication settings, and Process Data Object (PDO) mappings, enabling the EtherCAT MainDevice to correctly configure and communicate with the SubDevice.

To import an ESI file, follow these steps:

1. Ensure that the TwinCAT XAE Shell program is closed.
2. Copy the ESI file generated from the Generate Stack Code and ESI File section.
3. Paste the file into the *C:\TwinCAT\3.1\Config\IO\EtherCAT* folder.

Whenever modifications are made to the ESI file, this procedure must be repeated.

4.3.5. Virtualization Technology in BIOS

TwinCAT requires hardware virtualization technology to be enabled on the PC to support real-time execution. Without it, the TwinCAT runtime cannot initialize its virtualized real-time kernel.

To enable hardware virtualization technology, follow these steps:

1. Enter the BIOS setup on your PC.
2. For Intel-based platforms:
   a. Navigate to **Advanced > Security > Security Features**, locate the Intel Virtualization Technology (VT-x) and enable it.
   b. If Intel Virtualization Technology for Directed I/O (VT-d) is available, enable it as well then restart your PC.
3. For AMD-based platforms:

   Navigate to **Advanced > CPU Configuration**, locate **AMD Virtualization (AMD-V)** or **Secure Virtual Machine (SVM)** and enable it.

4.3.6. TwinCAT Project Settings

1. Launch the TwinCAT XAE Shell from the Windows Start menu.

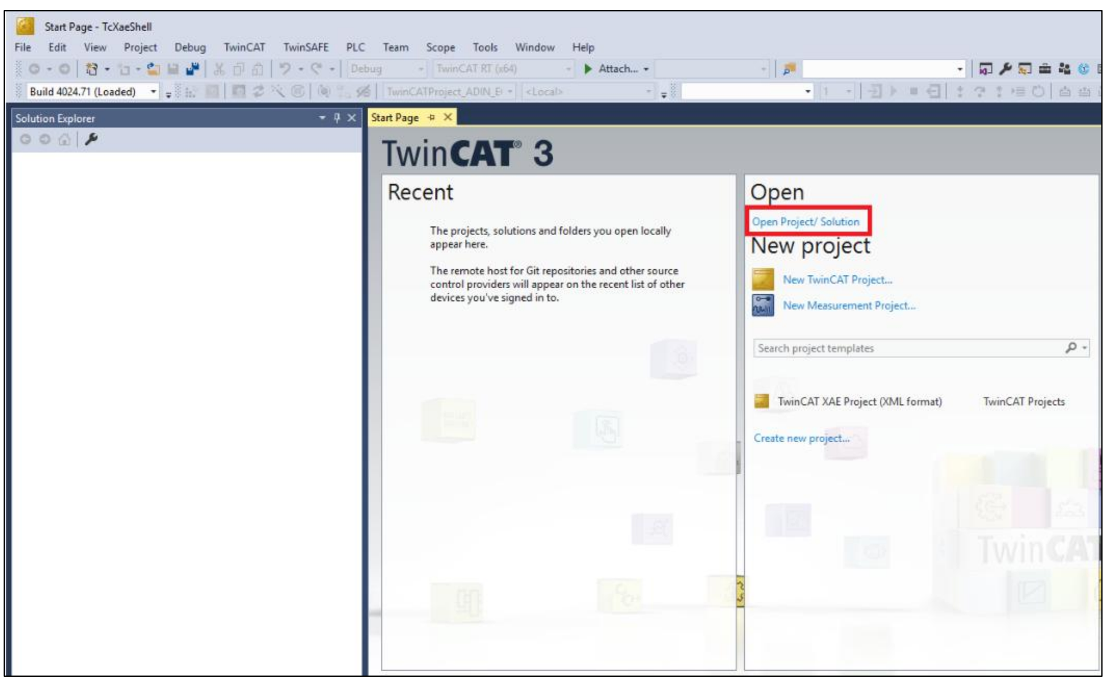

Figure 4.27. TwinCAT Main Page to Open Project/Solution

2. Click on the **TwinCATProject_ADIN _ECAT_MotorCtrl.sln** from the downloaded **.zip** file to open the TwinCAT project.

4.3.6.1. Trial License Generation

If you have previously generated a TwinCAT license or are still within the 7-day Trial License period, you may skip the following steps:

1. In the **Solution Explorer** panel on the left, expand the **SYSTEM** tree view and click **License**.
2. Click the **7 Days Trial License** button and enter the **Security Code** as prompted on the right.
3. A pop-up window appears indicating that license generation is complete. Click **OK** to continue.

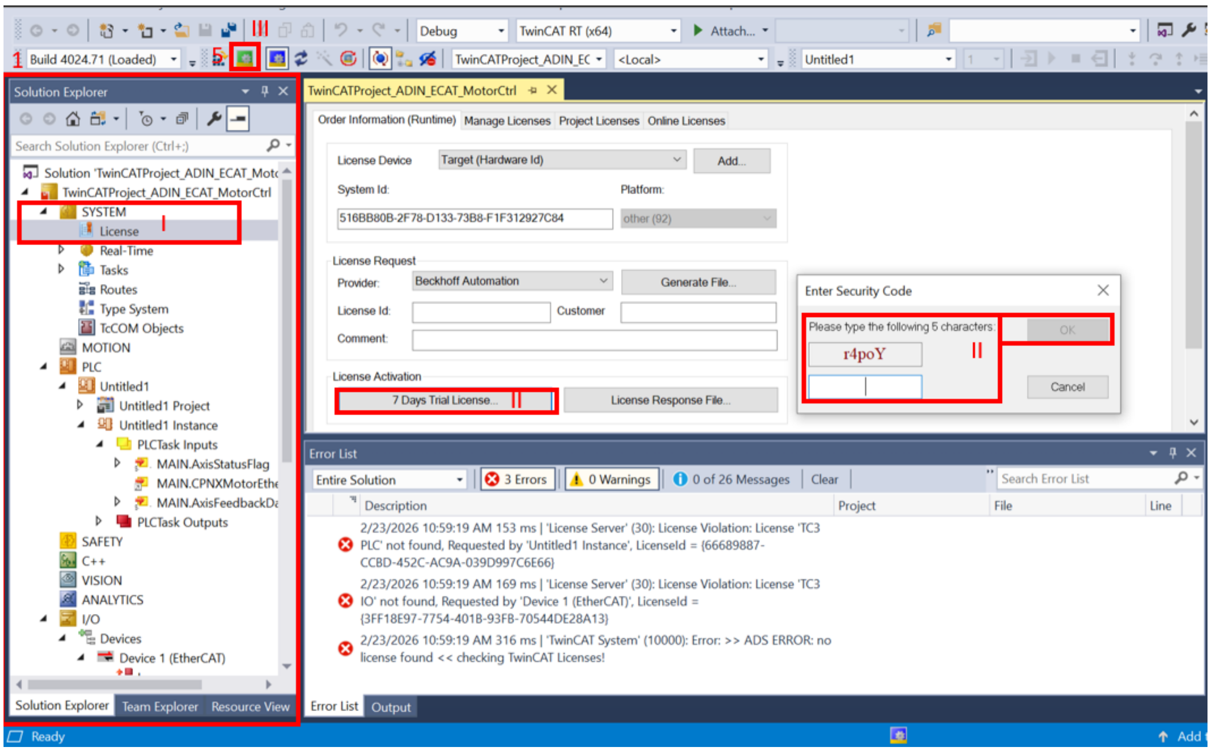

Figure 4.28. TwinCAT Trial License Generation

4.3.6.2. Ethernet Network Adapter Binding

1. This process is required to select the network adapter used by TwinCAT. In the **Solution Explorer** panel, expand the **I/O** tree, select **Devices,** and then click **Device 1 (EtherCAT)**.
2. Click on the **Adapter** tab on the right panel, and then click **Search**. A list of devices will appear in a new window as shown below.

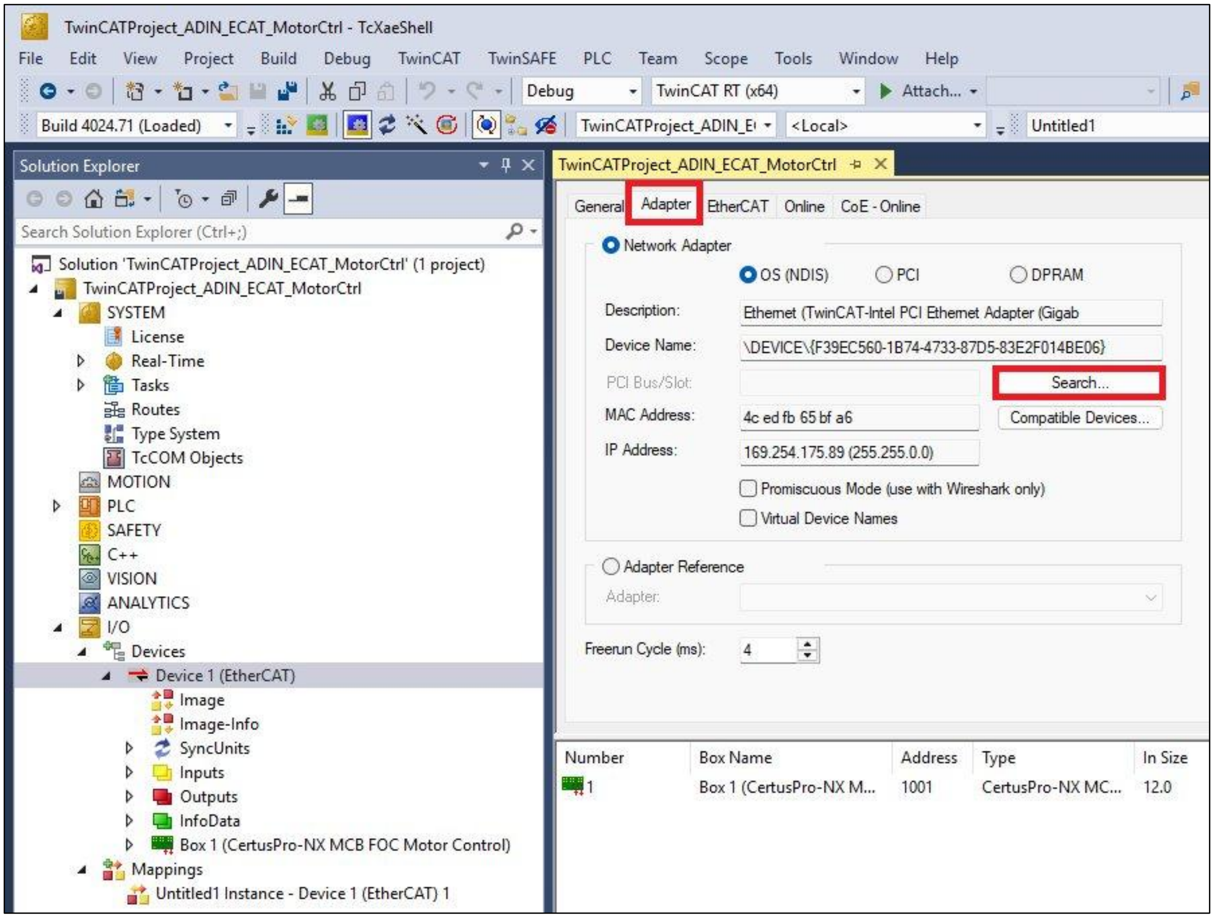

Figure 4.29. TwinCAT Project Network Adapter Information Page

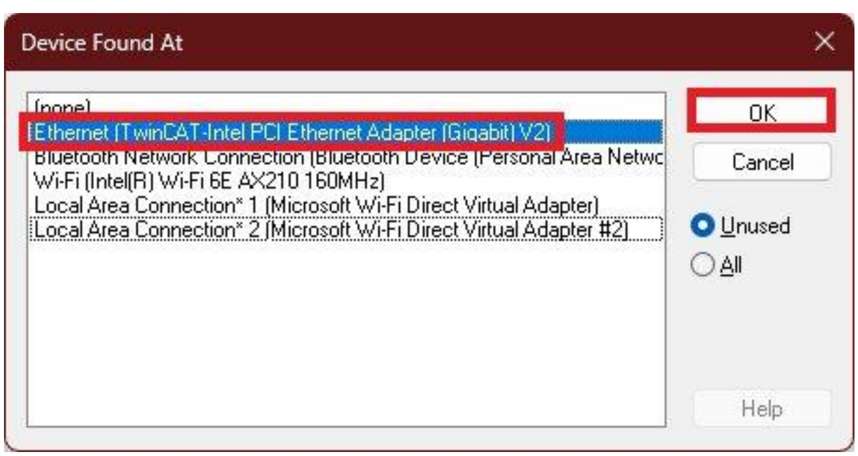

Figure 4.30. Network Adapter Selection List for EtherCAT Communications

3. Select the network adapter that was installed earlier based on the installation steps in the Ethernet Adapter Driver Installation section, and click **OK**.
4. Save the project by holding **Ctrl + Shift + S** buttons on your keyboard.

4.3.6.3. Isolated Core

To achieve deterministic performance, it is important to dedicate one full physical CPU core exclusively to TwinCAT. In a typical configuration, Windows and TwinCAT share the available CPU cores, and interruptions from the Windows scheduler can disrupt TwinCAT's real-time tasks. By enabling core isolation, TwinCAT gains exclusive ownership of a CPU core, ensuring that its real-time tasks are isolated from Windows processes. If processor multithreading is enabled, all logical cores associated with the same physical core should be isolated.

1. Expand the **SYSTEM** tree view on the left, and click **Real-Time**.

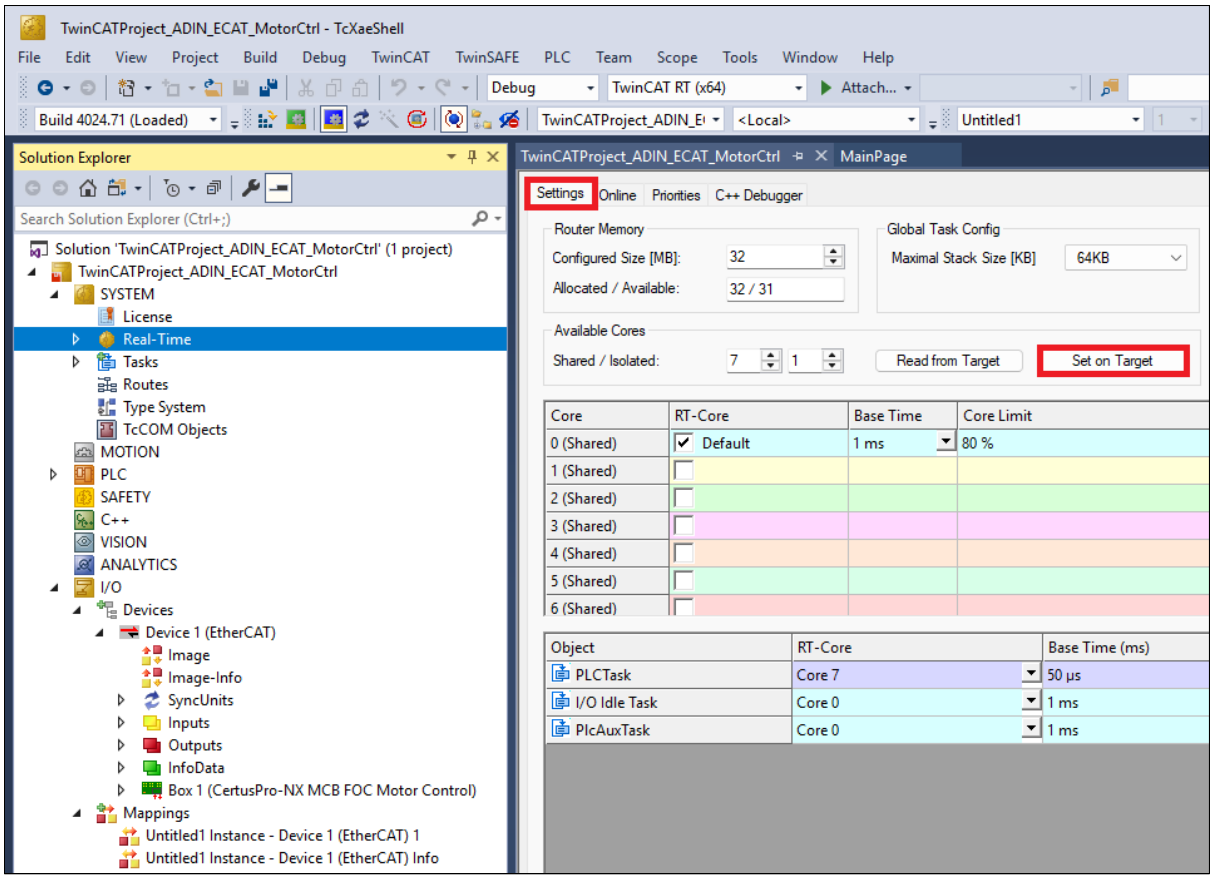

Figure 4.31. TwinCAT Project Real-Time Settings Page

2. On the **Settings** tab, click the **Set on Target** button. A new window will appear allowing you to change the number of shared cores. Select the desired core count using the up and down arrows. Reduce the number of shared cores so that one full physical CPU core remains isolated for TwinCAT.

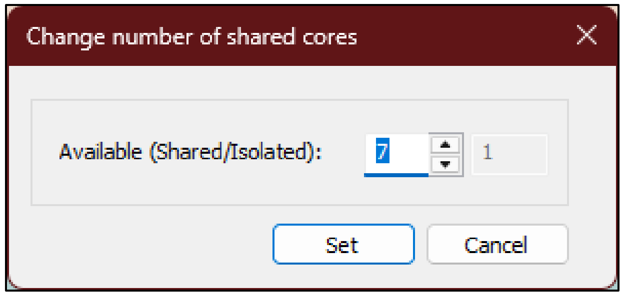

Figure 4.32. Configure the Number of Shared and Isolated Cores

3. Click the **Set** button, then click **OK** in the next prompt message.
4. When the reboot prompt appears, select **No**.
5. Press **Ctrl + Shift + S** to save the project, then exit the TwinCAT program.
6. Restart your PC.

4.3.6.4. Core Assignment

1. After restarting the PC, launch **TwinCATProject_ADIN _ECAT_MotorCtrl.sln** again.
2. In the **Solution Explorer**, expand the **SYSTEM** tree and click **Real-Time**.
3. On the **Settings** tab, locate the core number labeled with the (Isolated) keyword and enable the **RT-Core** checkbox next to it.
4. Set the **Base Time** of the isolated core to **50 µs**.

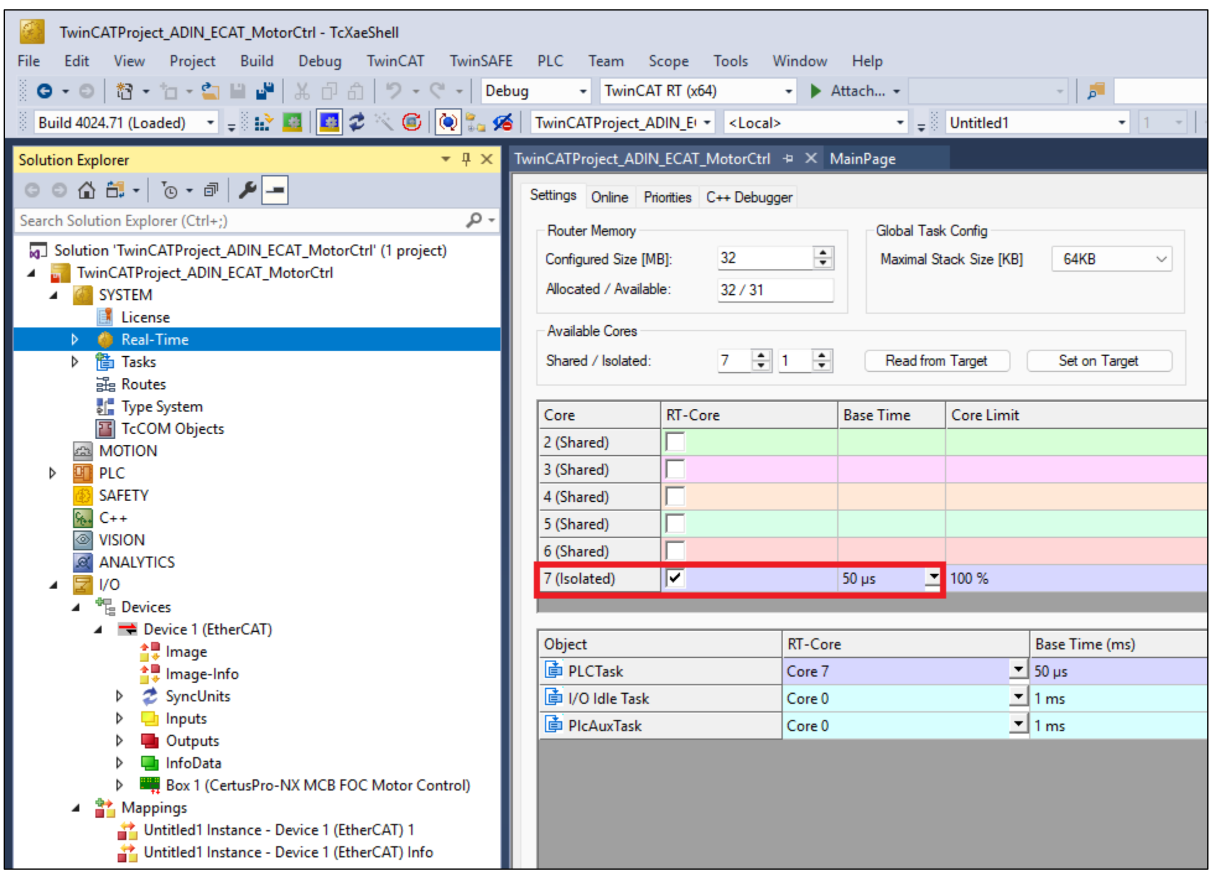

Figure 4.33. Selecting an Isolated CPU Core for a TwinCAT Project

5. In the **PLCTask** under the **Object** column, as shown in screenshot below, select the RT-Core from the drop-down list to match the isolated core number.

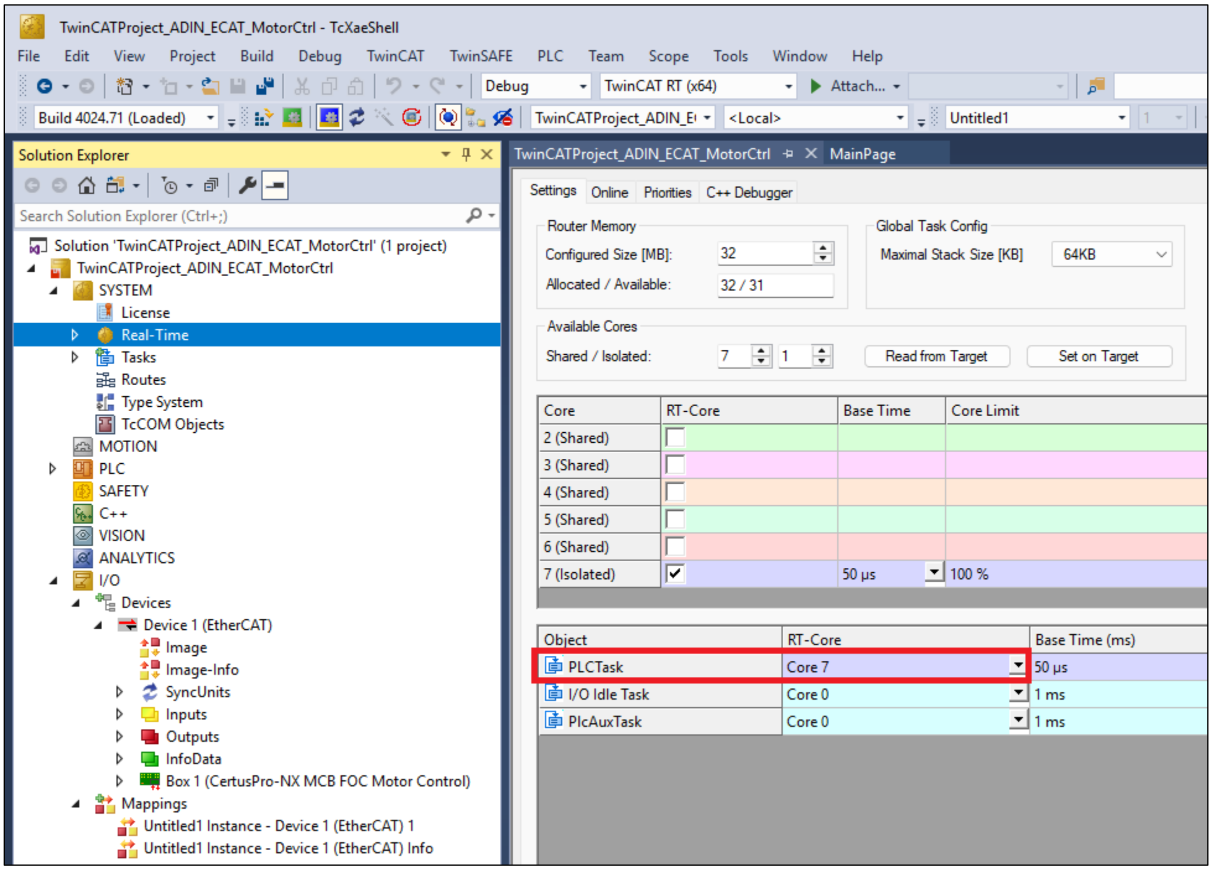

Figure 4.34. PLC Task CPU Core Assignment

6. Press **Ctrl + Shift + S** again to save the project.

4.3.6.5. Setting Cycle Time

1. In the **Solution Explorer** panel, expand the **SYSTEM** tree view and the **Tasks** tree view. Click **PLCTask**.
2. On the **Task** tab, modify the number of cycle ticks using the numeric up and down controls. The resulting cycle time is displayed alongside the setting.
3. For this design, the cycle ticks are set to **4**, resulting in a cycle time of **0.200 ms (200 µs)**.
4. Press **Ctrl + Shift + S** again to save the project.

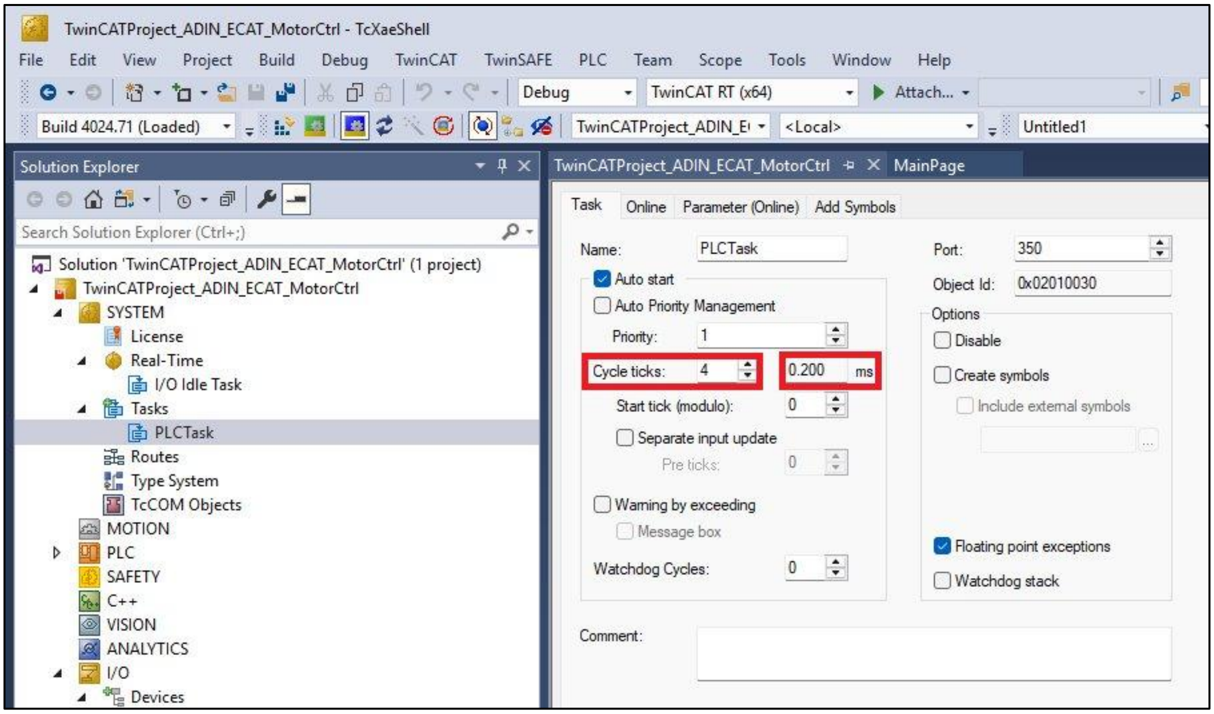

Figure 4.35. PLC Task Cycle Ticks Configuration

4.3.6.6. Replace Device

1. Right-click on **Box 1** and select **Change to Compatible Type…**

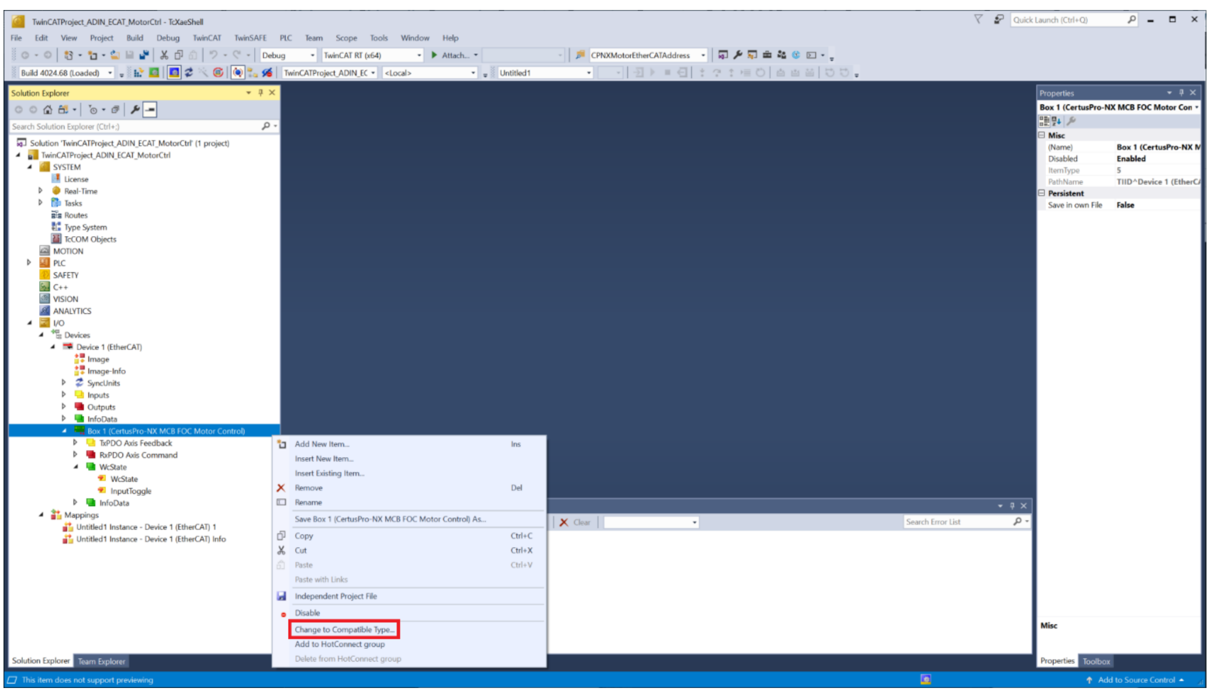

Figure 4.36. Changing an EtherCAT SubDevice to a Compatible Type

2. In the pop-up window, select **CertusPro-NX Single Axis FOC Motor Control**, and click **OK**.

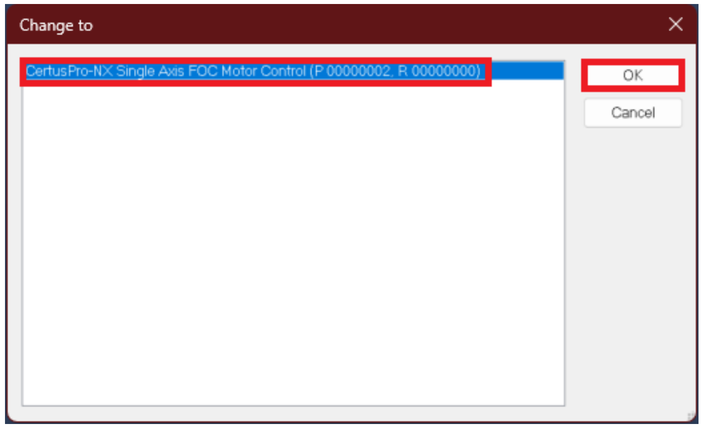

Figure 4.37. Compatible EtherCAT SubDevice List

4.3.6.7. Activating Configuration

1. Click on the **Activation Configuration** icon at the top, as shown in the figure below.

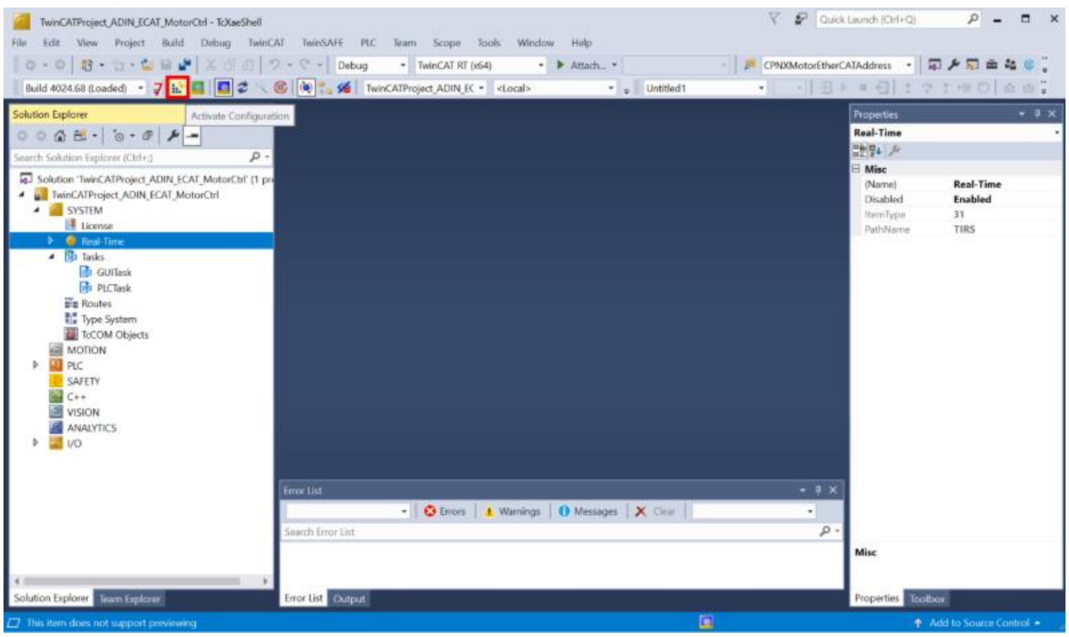

Figure 4.38. Activate the Configuration Button on the Toolbar

2. Click **OK** when the **Activate Configuration** prompt appears.

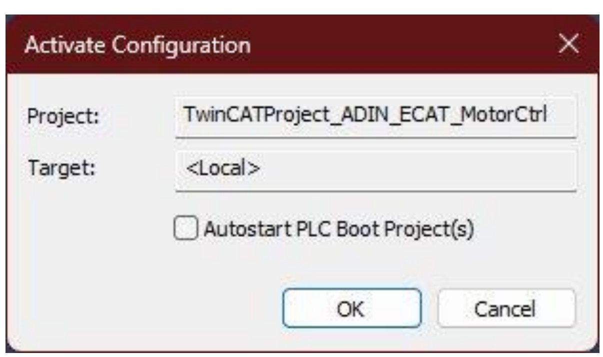

Figure 4.39. Activate Configuration Window

3. Click **Cancel** when prompted to restart the TwinCAT system in Run Mode.

Figure 4.40. Restart TwinCAT in Run Mode Window

4.4. Running TwinCAT

1. Power up the Motion Control Platform board and the Motor Control Solution Kit.
2. Open the **Programmer** in the Radiant software and verify that the automatically loaded **.bit** file is the one most recently generated by checking the file creation timestamp.
3. Load the generated **.bit** file into the FPGA.
4. Launch the TwinCAT XAE Shell from the Windows Start menu.
5. Open **TwinCATProject_ADIN _ECAT_MotorCtrl.sln** from the downloaded **.zip** file to load the TwinCAT project.

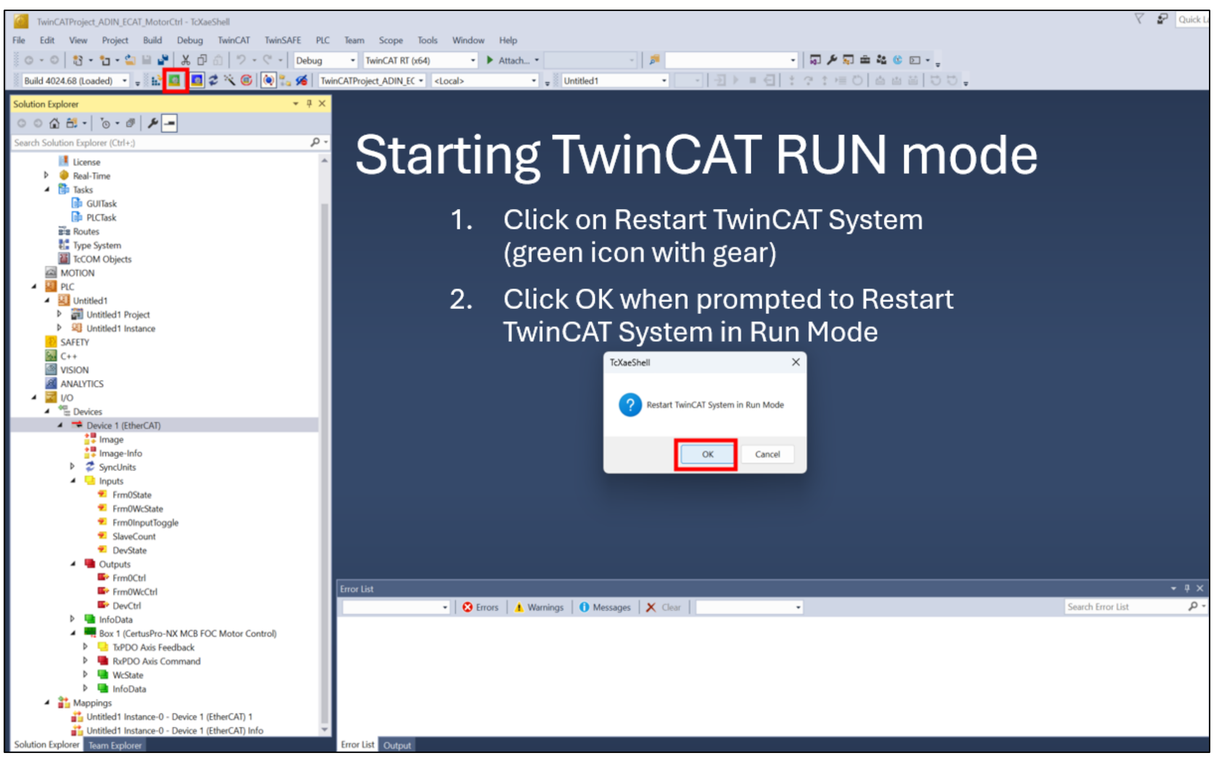

Figure 4.41. TwinCAT Run Mode Button on the Toolbar and Confirmation Window

6. To start the run, first click the **Restart** icon (the green icon with a gear) at the top. When the prompt message box appears, click **OK**.

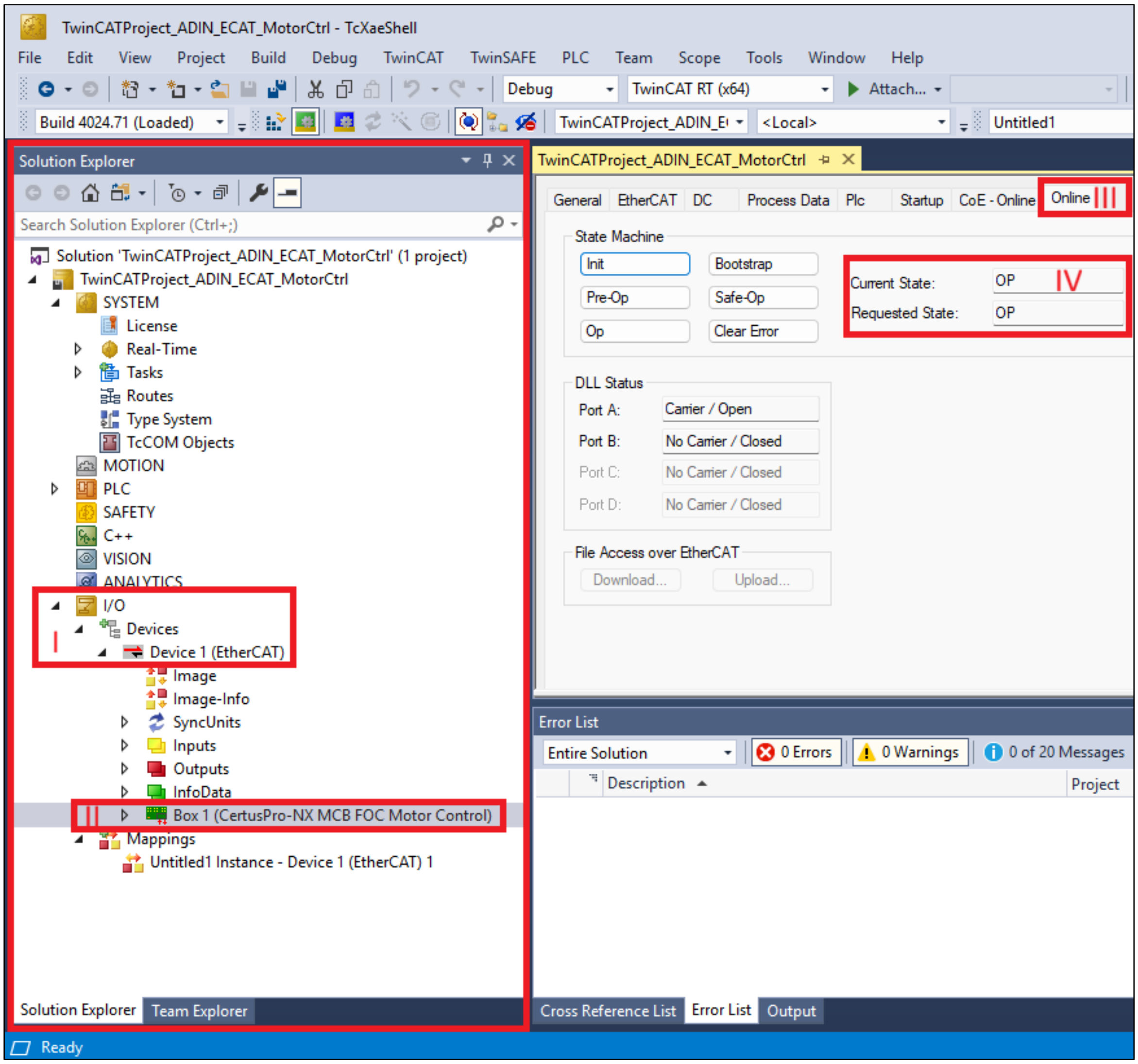

Figure 4.42. EtherCAT SubDevice State Machine Status

7. To validate that the EtherCAT connection has been established successfully, follow these steps:
   a. In the **Solution Explorer** tab, expand the **I/O** tree view, then expand **Devices**. You should see **Device 1 (EtherCAT)**. Click it to further expand the tree view.
   b. Double-click **Box 1 (CertusPro-NX MCB FOC Motor Control)**.
   c. Click the **Online** tab.
   d. Verify that the Current and Requested State display **OP**.
   e. On the Motion Control Platform board, confirm that the on-board **LED D106 is** illuminated, indicating that the SubDevice is in the Operational State.

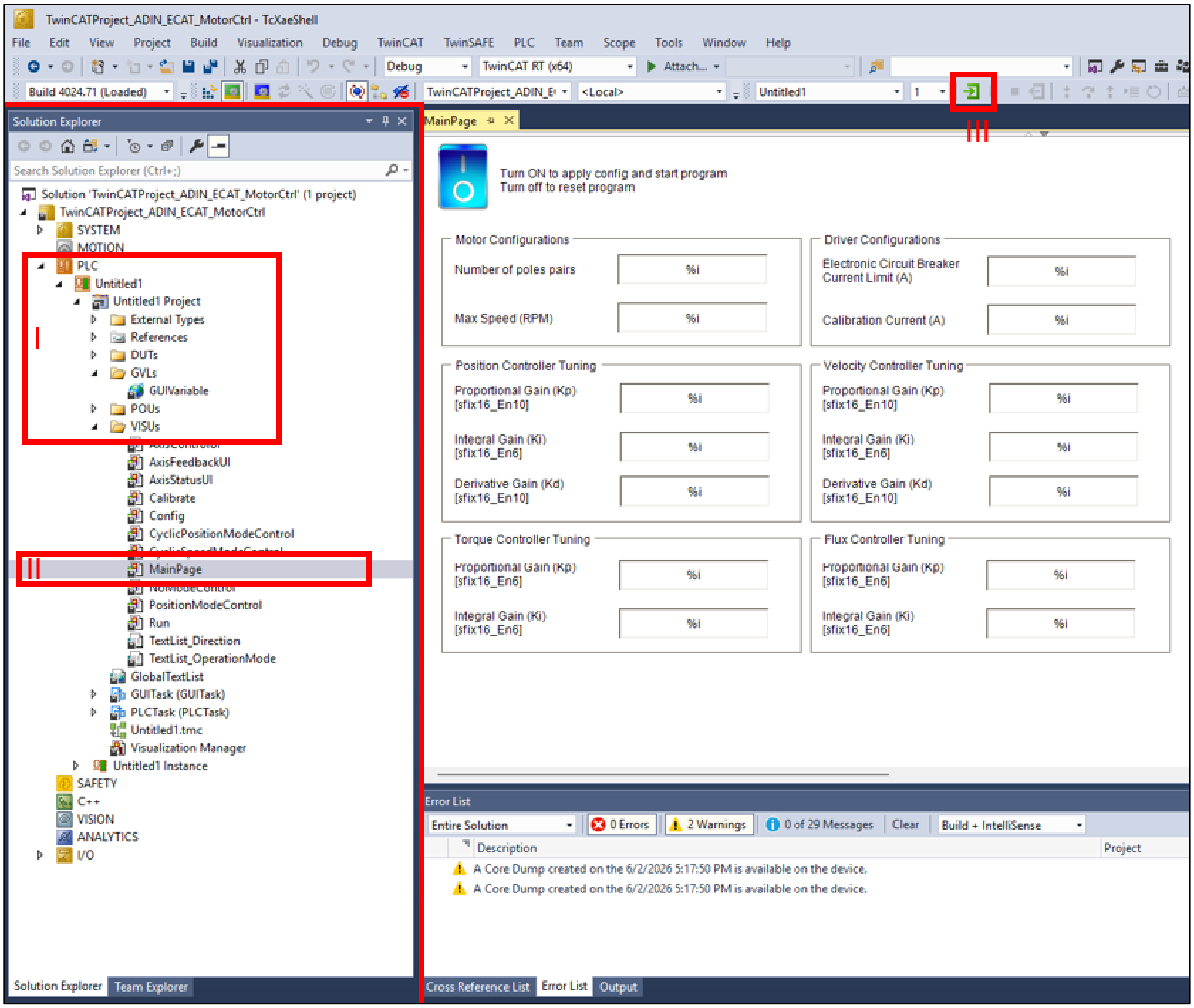

Figure 4.43. TwinCAT Project Visualization Main Page and PLC Login Button in the Toolbar

8. After verifying that the EtherCAT connection has been successfully established, we may proceed to start the PLC program.
   a. In the **Solution Explorer**, expand the **PLC** tree view, then navigate to **Untitled1** → **Untitled1 Project** → **VISUs**.
   b. Double-click **MainPage**, which will open the Main Page tab on the right.
   c. Click the **Login** icon (the green right-arrow icon) at the top.

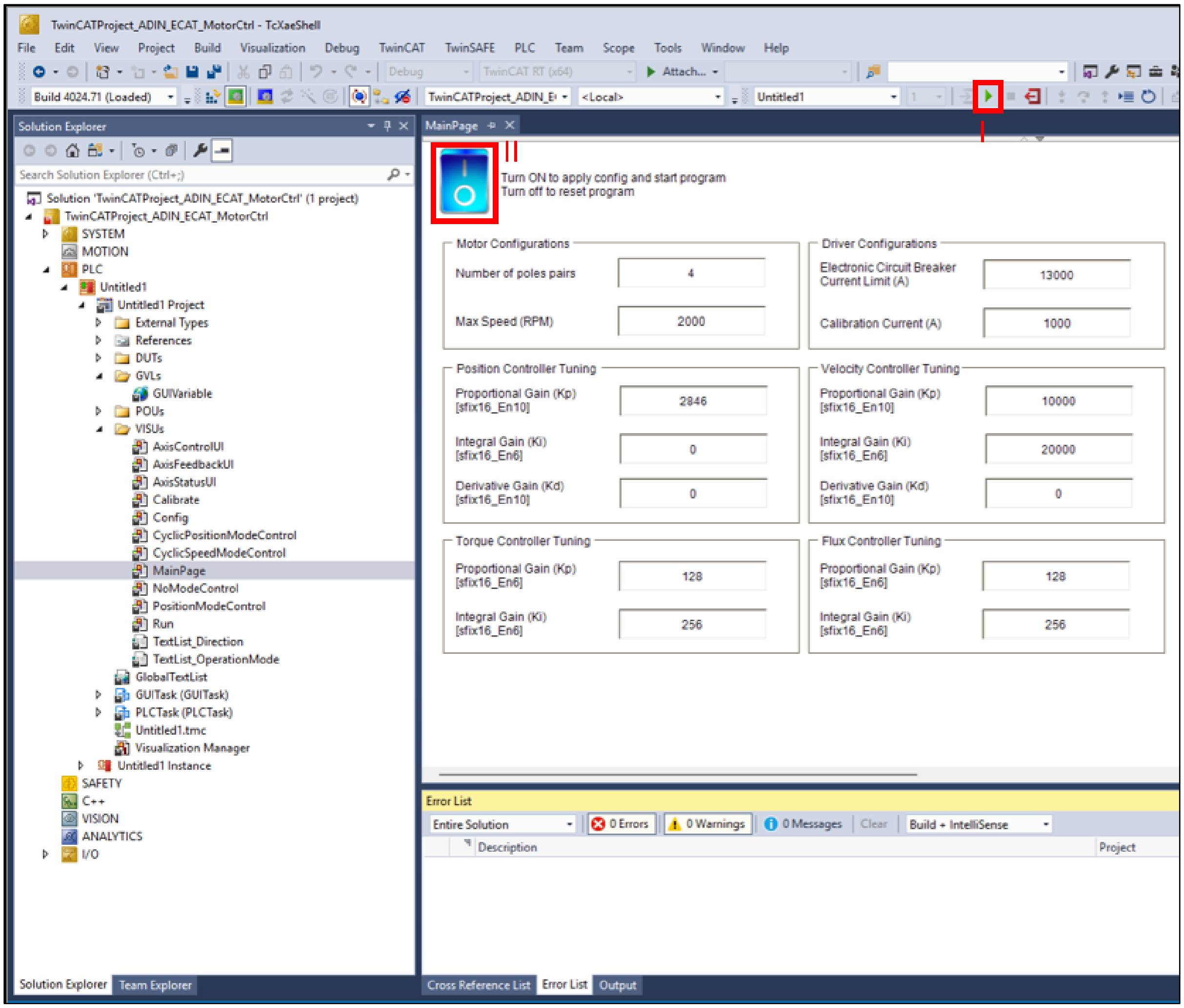

Figure 4.44. PLC Start Button in the Toolbar and Power Switch in Project Visualization Main Page

9. The **Play** button next to the Login button should now be enabled.
    a. Click the **Play** button to start the PLC.
    b. Click the **Power Switch** button below to start the program.

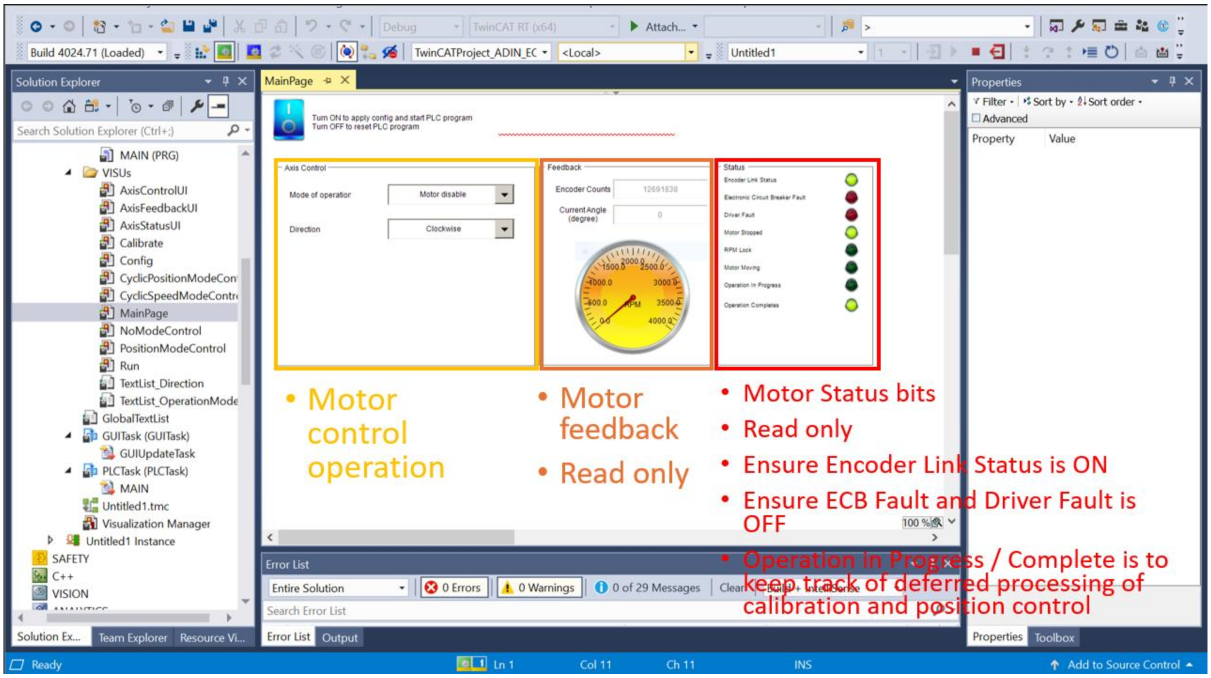

Figure 4.45. Motor Control, Feedback, and Status Visualization

10. When the program starts, a new GUI appears, allowing you to control the motor.
    a. On the left, two parameters can be selected under **Axis Control: Mode of Operator** and **Direction**. At startup, the **Mode of Operator** is set to **Disabled**, and four additional operating modes are available for selection. More details about these modes are described later.
    b. In the center, the **Feedback** box displays the current motor position in terms of angle (degrees), as well as the motor speed in RPM.
    c. On the right, several status indicators display the current system health, including **Encoder Link Status**, **Electronic Circuit Breaker Fault**, **Driver Fault**, and other related signals.

4.4.1. Calibration Mode

In Calibration mode, click the blue **Calibrate** button to start the operation. This process performs rotor alignment and captures the encoder offset (no homing is required). During calibration, some motor movement may occur, and the Axis Control UI will be disabled until the operation is complete.

While the calibration is in progress, the Operation in Progress status LED on the right will turn on, indicating that calibration is underway. After the process completes, the Operation Completes status LED will turn on. At this point, the Current Angle (degree) value should be reset to 0.

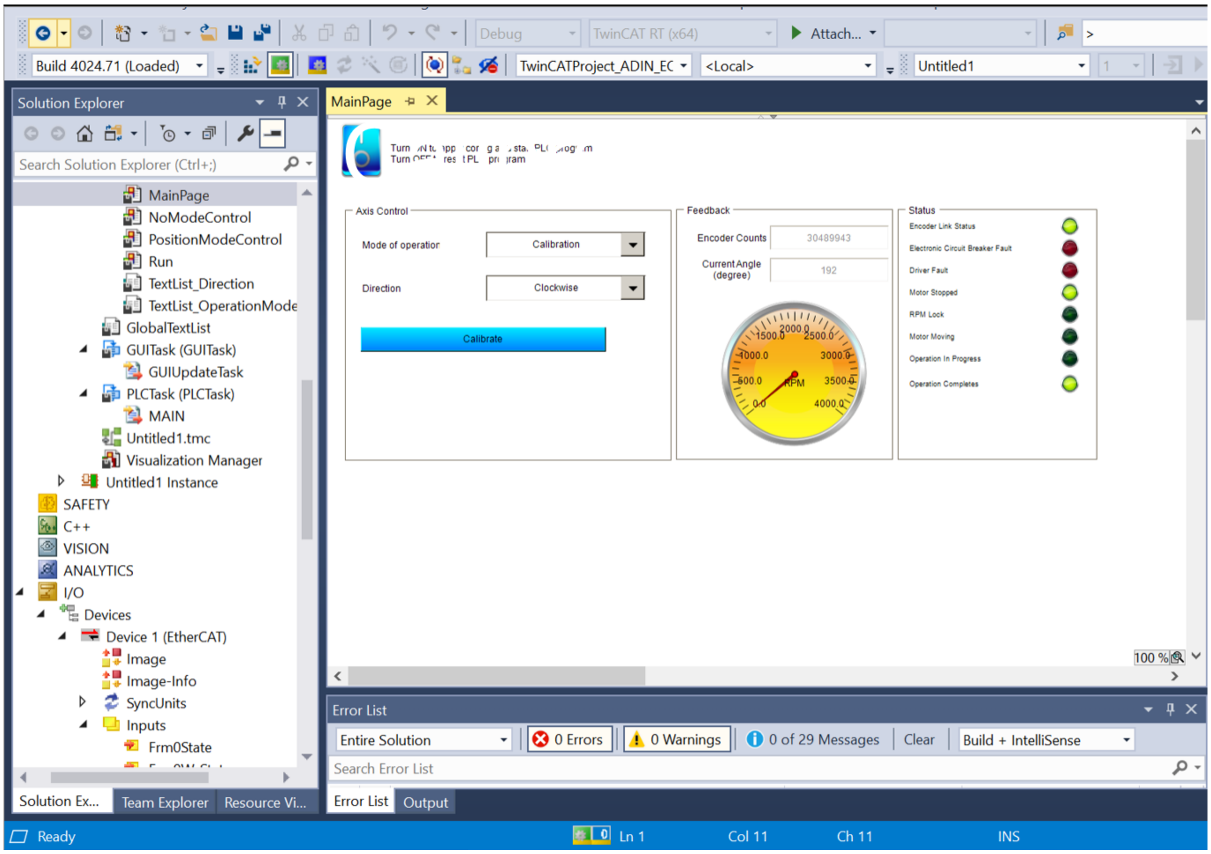

Figure 4.46. Calibration Mode Visualization

4.4.2. Position Control Mode

In Position Control mode, before clicking the blue **Trigger** button, set the desired **Direction**, **Full Rotation count**, and target **Angle (degree)**. After clicking **Trigger**, the motor will begin moving and the control UI will be disabled during the operation.

Similar to Calibration mode, the **Operation in Progress** status LED on the right will turn on while the motion is underway. Once the operation is complete, the **Operation Completes** status LED will turn on, and the final position feedback will be displayed in the **Current Angle (degree)** field.

Known issue: You may occasionally observe slight inaccuracies in the position angle due to a limitation in the Motor Control IP. This issue will be addressed in a future release.

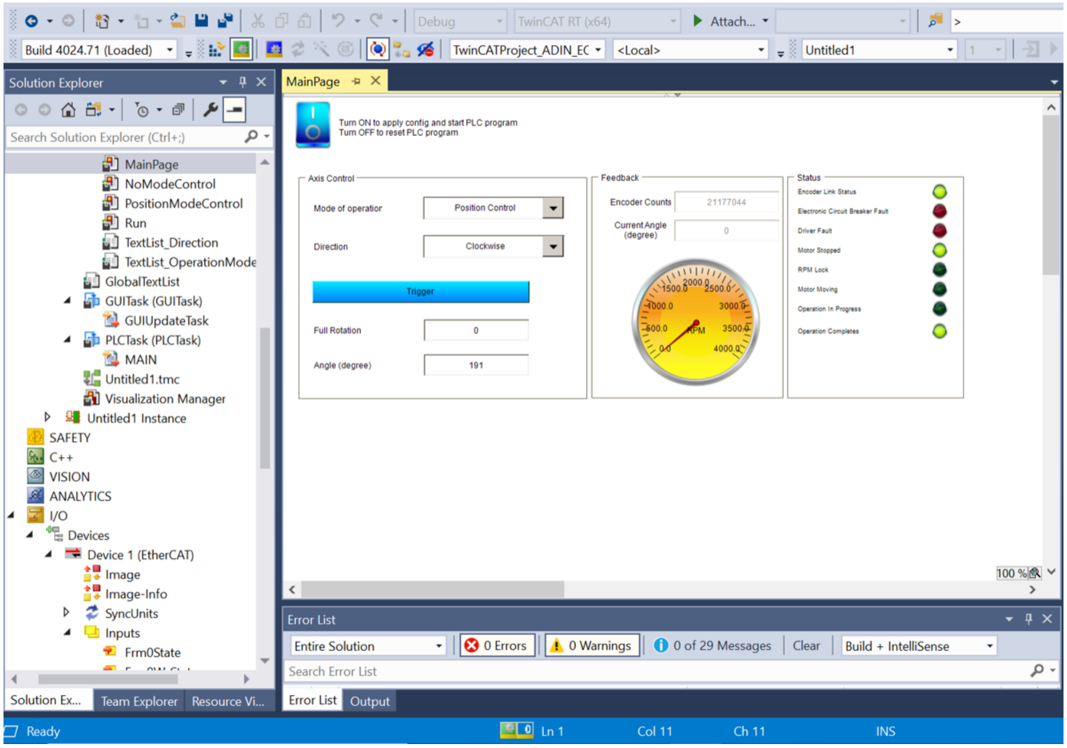

Figure 4.47. Position Control Mode Visualization

4.4.3. Cyclic Position Control Mode

In this mode, you can enter the desired target value in the **Angle (degree)** field, and the motor will move to the specified position. The value displayed in the **Current Angle (degree)** field within the **Feedback** box should reflect the target angle you set.

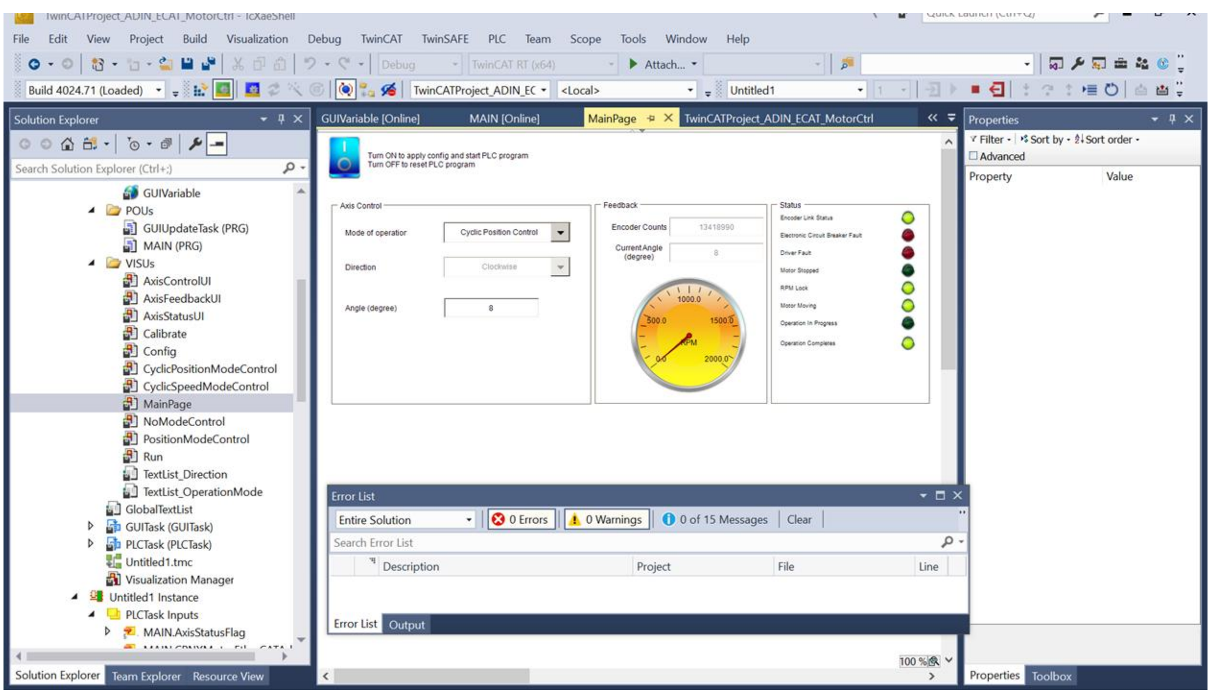

Figure 4.48. Cyclic Position Control Mode Visualization

4.4.4. Cyclic Velocity Mode

In this mode, first set the desired direction of motor rotation. Next, adjust the slider under **Speed (RPM)** to set the target speed. The current speed feedback is displayed on the yellow speedometer.

Note: With the current Motor Control Solution Kit, motor noise may occur when the RPM is below 120.

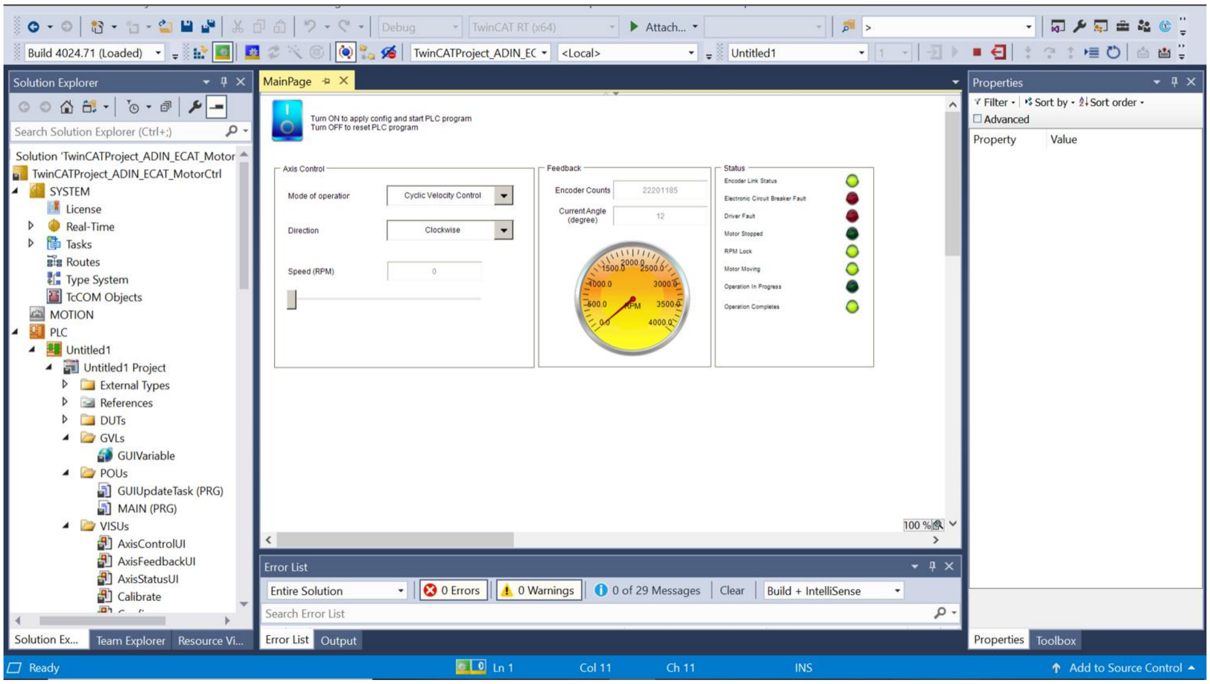

Figure 4.49. Cyclic Velocity Mode Visualization

5. Modifying Existing Motor Control Application

All object dictionary definitions, PDO mappings, configuration macros, and ESI/EEPROM data for this reference design are managed using the EtherCAT SubDevice Stack Code (SSC) tool. To get started, always refer to the *Application Note ET9300 (EtherCAT SubDevice Stack Code)* on the EtherCAT Technology Group (ETG) website and the *EtherCAT SubDevice Design Quick Guide* on the Beckhoff website.

You can customize the objects by modifying the application definition spreadsheet file. Whenever this spreadsheet is updated, a newly generated ESI file from the SSC tool must be imported into TwinCAT, as described in the Importing ESI File into TwinCAT section. If any changes are made to the process data, you may need to remove and re-add the device in TwinCAT and perform the PDO remapping again.

5.1. Example: Changing Default Motor Parameters

The application defines configuration SDO objects (0x8000–0x8005) for motor and controller configuration. The default values of these objects can be modified directly through the spreadsheet. Optionally, new values can be written to these objects at runtime. In this case, the values are applied during the transition from the Pre-Operational (PREOP) state to the Safe-Operational (SAFEOP) state.

To change the default pole pair count from 4 to 7 and the maximum speed from 4000 RPM to 3000 RPM, follow these steps:

1. Open **CPNX_MCB_ECAT_FOC_MotorControl.xlsx** application definition spreadsheet.
2. Locate the row corresponding to object **0x8000, Subindex 1 (Pole Pairs)**.
3. Change the value in the default column from 4 to 7.
4. Locate the row corresponding to object **0x8000, Subindex 2 (Max Speed RPM)**.
5. Change the value in the default column from **4000** to **3000**.
6. Save and close the Excel file.
7. Repeat the procedure described in the Import Application Definition File and Generate Stack Code and ESI File sections.
8. The updated default values appear in the regenerated **CPNX_MCB_ECAT_FOC_MotorControlObjects.h** file.
9. Copy the updated **CPNX_MCB_ECAT_FOC_MotorControlObjects.h** file into the **ethercat** folder of the reference design and recompile the code.

5.2. Example: Adding New Configuration SDO object

1. Open the **CPNX_MCB_ECAT_FOC_MotorControl.xlsx** application definition spreadsheet.
2. Add a new object within 0x8006 – 0x8FFF.
3. Save and close the Excel file.
4. Repeat the procedure described in the Import Application Definition File and Generate Stack Code and ESI File sections.
5. Copy the updated **CPNX_MCB_ECAT_FOC_MotorControlObjects.h** file into the **ethercat** folder of the reference design and recompile the code.
6. Add your own application logic to reference and apply the newly created object.
7. Recompile the code.

5.3. Example: Adding New Transmit Process Data

1. Open the **CPNX_MCB_ECAT_FOC_MotorControl.xlsx** application definition spreadsheet.
2. Add a new object in the range **0x7003 – 0x7FFF** for transmit process data.
3. Create a new TxPDO mapping in 0x1A01 and map it to the corresponding object index, subindex, and bit length.
4. Add a new subindex entry for new TxPDO mapping in the **0x1C13 SyncManager 3 Assignment**.

5. Save and close the Excel file.
6. Repeat the procedure described in the Import Application Definition File and Generate Stack Code and ESI File sections.
7. Copy the new **CPNX_MCB_ECAT_FOC_MotorControlObjects.h** and **ecat_def.h** into the **ethercat** folder in the reference design.
8. Add application-specific logic to reference and use the newly created object.
9. Add your own application logic to reference and apply the newly created object.
10. Modify **APPL_InputMapping()** to copy data from local memory to the process data buffer memory.
11. Recompile the code and follow the Generate Design section.

5.4. Example: Adding New Receive Process Data

1. Open the **CPNX_MCB_ECAT_FOC_MotorControl.xlsx** application definition spreadsheet.
2. Add a new object **0x6004 – 0x6FFF** for transmit process data.
3. Create a new RxPDO mapping under 0x1601 and map it to the corresponding object index, subindex, and bit length.
4. Add the new subindex entry for the RxPDO mapping to the 0x1C12 SyncManager 2 Assignment.
5. Save and close the Excel file.
6. Repeat the procedures described in Import Application Definition File and Generate Stack Code and ESI File sections.
7. Copy the new **CPNX_MCB_ECAT_FOC_MotorControlObjects.h** and **ecat_def.h** into the ethercat folder in reference design.
8. Add your own application logic to reference and apply the newly created object.
9. Modify **APPL_OutputMapping ()** to copy data from process data memory to local memory.
10. Recompile the code and refer to the Generate Design section.

5.5. Creating a New Application

To replace the motor control application entirely with your own design, follow the standard SSC Tool workflow to generate a new application, then integrate the output with this platform.

1. Create a new SSC Tool project as described in the Create new SSC Project section.
2. From the **Tool** menu, select **Application**, then click **Create new**.
3. Define all the objects in the newly created Excel application definition file.
4. Generate SubDevice files as described in the Generate Stack Code and ESI File section.
5. Remove the existing CPNX_MCB_ECAT_FOC_MotorControl.c, CPNX_MCB_ECAT_FOC_MotorControl.h and CPNX_MCB_ECAT_FOC_MotorControlObjects.h
6. Implement your custom logic in the empty APPL_* callback function stubs.
7. Recompile the code and follow the steps in the Generate Design section.

6. Debugging

If you encounter any issues while using the reference design, follow these steps to debug and resolve them:

1. When the network connection between the host PC and the board is established, LED0 (D63) should be illuminated.
2. LED D106 indicates the EtherCAT current running mode:
   - Off: The EtherCAT SubDevice is in the INIT state.
   - Blinking (200 ms on/off alternating): The EtherCAT SubDevice is in the Pre-operational state.
   - Single flash (200 ms on + 1000 ms off): The EtherCAT SubDevice is in the Safe-operational state.
   - On (static): The EtherCAT SubDevice is in the Operational state.
   - Flickering (50 ms on/off alternating): The EtherCAT SubDevice is booting and has not entered the INIT state.

   For more details on the LED run state, refer to the *ETG.1300 EtherCAT Indicator and Labeling Specification* document on the EtherCAT Technology Group website.
3. If you see the following error in the TwinCAT program shown in the figure below, this may indicate that you have reached the timebomb limit of 1 hour in the EtherCAT IP if you are using the evaluation license. You will need to restart the setup by power cycling the board, as well as repeat the steps in the Running TwinCAT section again.

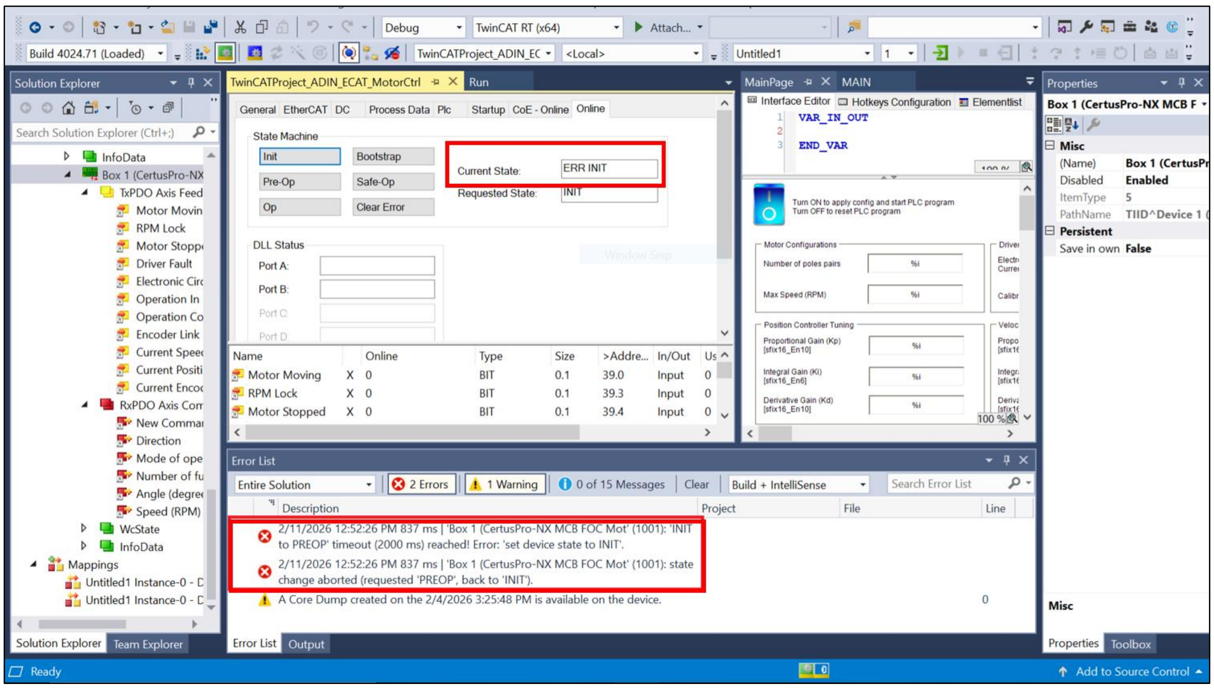

Figure 6.1. SubDevice Error in INIT State

4. If you see the error shown below, where the current state is stuck in the INIT state:
   a. Unplug the RJ-45 cable between the host PC and the board, then reconnect it. Repeat the steps in the Running TwinCAT section and ensure that LED D63 is flickering, which indicates EtherCAT link activity.
   b. If LED D63 is flickering but the device state remains in INIT, check the wiring of the encoder connection. You may connect to the device UART port at a baud rate of 115200. The UART output should indicate that the device is repeatedly retrying to establish communication with the encoder.

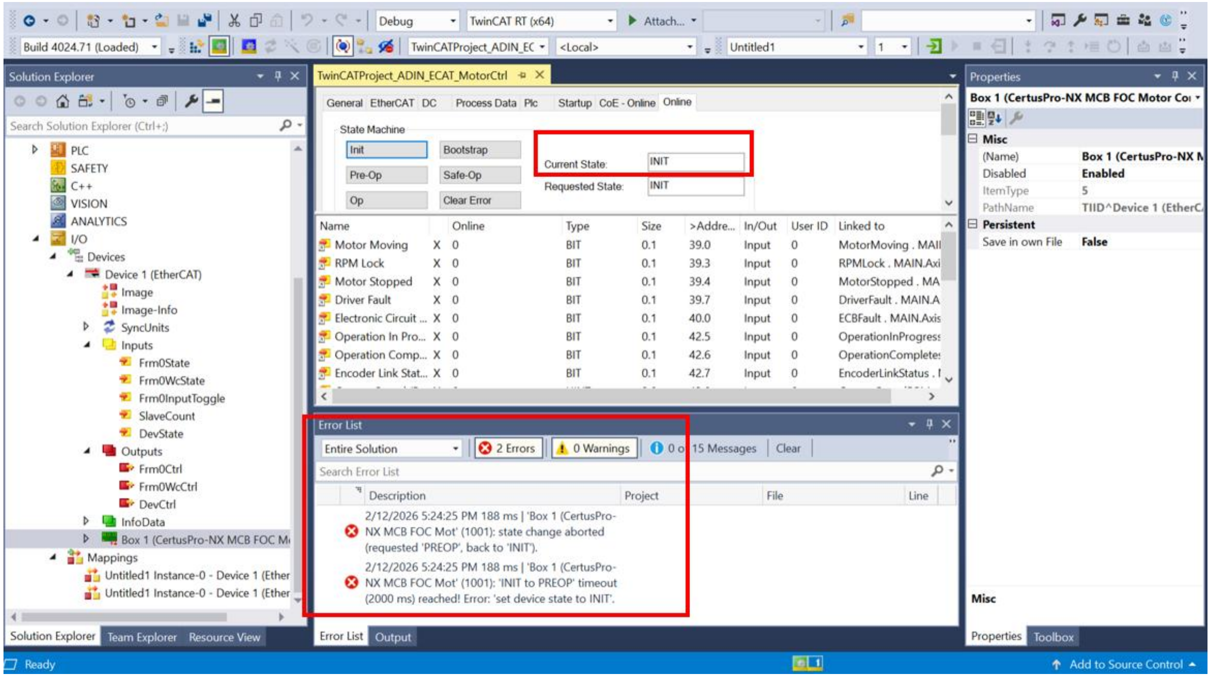

Figure 6.2. SubDevice Stuck in INIT State

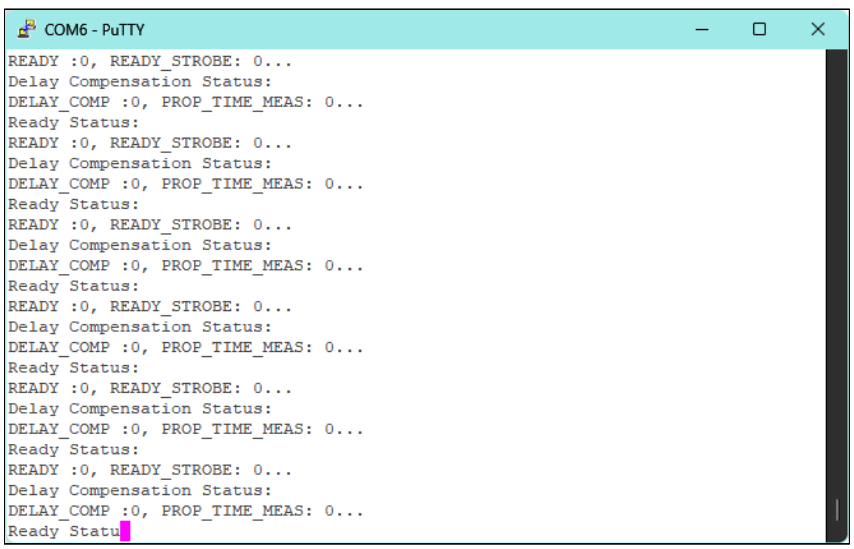

Figure 6.3. Serial Terminal Output Message for Encoder Initialization Failure

5. If you encounter the license-related error shown in the figure below, it likely indicates that you do not have a valid TwinCAT license or that the 7-day trial license has expired. Refer to the Trial License Generation section to generate a new license.

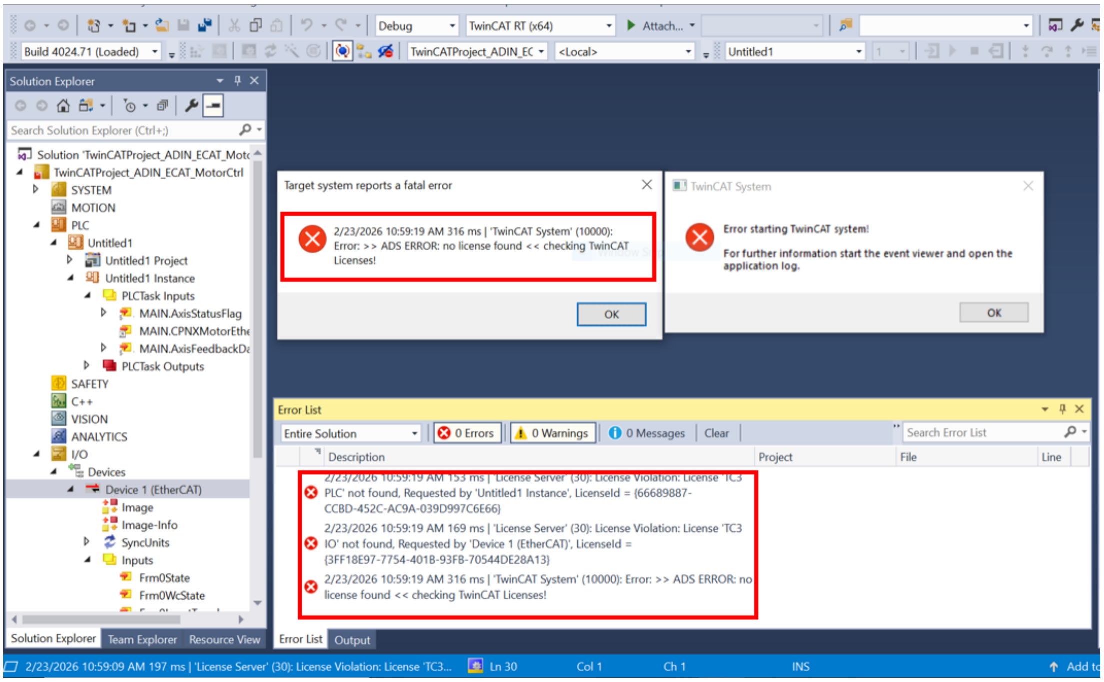

Figure 6.4. TwinCAT Project License Error

6. A Radiant software compilation failure due to a reference to the undefined module lsc_foc_motor_control can occur in the following scenario. If you have made changes to one or more IPs in the Propel project and regenerated the project, the required Verilog files may not be automatically added. After clicking the Run Radiant button and launching the Radiant software interface, you may notice that the Verilog files in the *lib/motor_control_m_codes* directory are missing. These files are required for the motor control IP to compile successfully.

   To resolve this issue, right-click **Input Files** in the Radiant software left panel, then select **Add -> Existing Files**. Add all the Verilog files from the *lib/motor_control_m_codes* directory to the project, and then proceed with the compilation again.

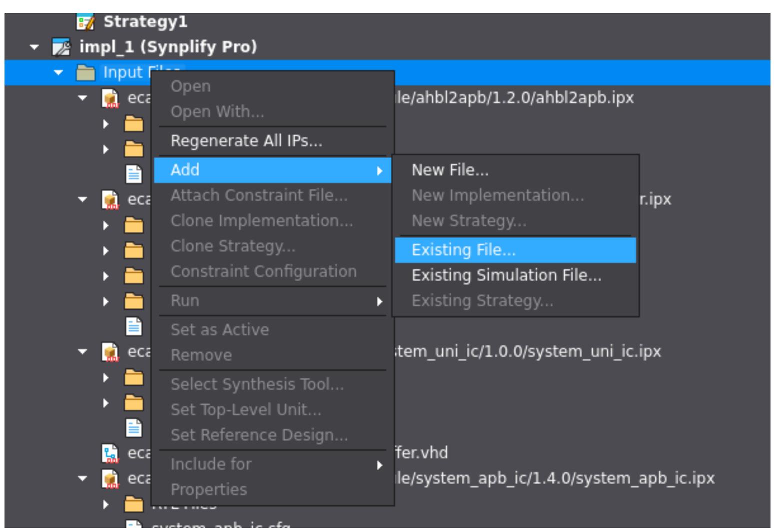

Figure 6.5. Options to Add Existing Design Files to the Radiant Software Project

7. If you encounter hold time violations within the EtherCAT IP in the reference design, try running the compilation with multiple seeds until you achieve a passing result. You can do this through the **Strategy** options by setting the **Placement Iterations** to more than one seed and enabling the **Stop Once Timing Is Met** option.

   Alternatively, visit the Knowledge Base page—Hold Time Violations When Compiling the EtherCAT Reference Design in Lattice Radiant 2025.2 to download the patch for Radiant software v2025.2 SP1, which improves the place-and-route algorithm to address the hold time violation in the design.

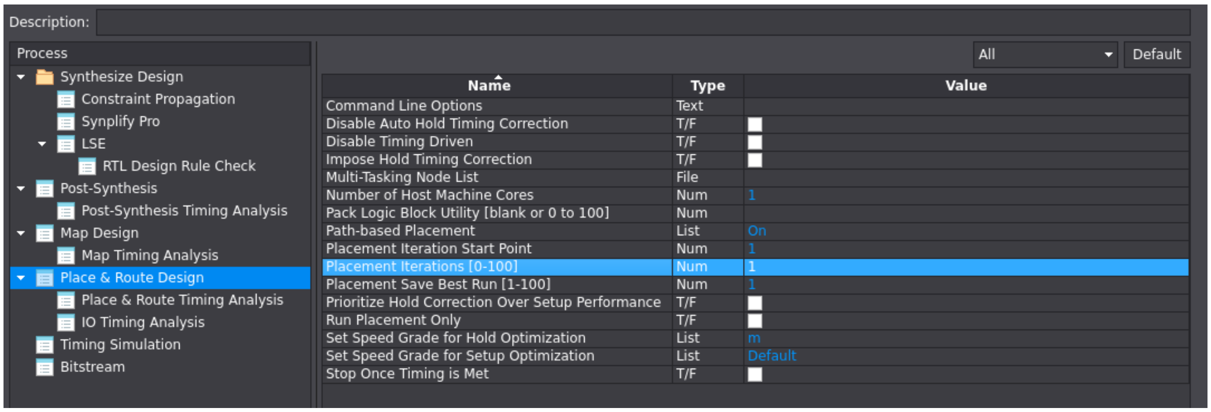

Figure 6.6. Enabling Additional Placement Iterations in the Strategy Options

7. Resource Utilization

Table 7.1. Resource Utilization

Instance	LUT4	Distributed RAM	PFU Registers	EBR	LRAM
Top	45,812	3,738	21,714	63	3
ethercat_inst	20,340	12	9,818	8	0
motor_control_inst	6,955	336	3,032	8	0
system_ospi_inst	6,185	3,216	1,267	0	0
riscv_rx_cpu_inst	5,106	6	3,595	18	0
motor_encoder_subsys_inst	3,009	0	1,304	3	0
system_uni_ic_inst	920	0	819	10	0
axi2ahbl_inst	786	54	253	0	0
system_data_mem_inst	656	114	382	16	0
system_uart_inst	707	0	609	0	0
system_tcm_inst	202	0	59	0	3
timer_inst	325	0	209	0	0
ahbl2apb_inst	238	0	144	0	0
gpio_inst	112	0	97	0	0
ahbl0_inst	98	0	7	0	0
system_apb_ic_inst	82	0	6	0	0
dual_boot_fpga_config_inst	55	0	55	0	0

Reference

- FOC Motor Control Reference Design User Guide (FPGA-RD-02325)
- Golden System Reference Design and Demo User Guide v3.0 for CertusPro-NX Devices (FPGA- RD-02322)
- Secure Connected Motion Control Platform web page
- RS485 Encoder Transceiver Board web page
- Trenz Electronic web page
- Anaheim Automation web page
- Heidenhain web page
- Lattice Radiant FPGA design software
- Lattice Solutions Reference Designs web page
- Lattice Solutions IP Cores web page
- Lattice Propel Design Environment web page
- Lattice Insights for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 1.0, May 2026

Section	Change Summary
All	Initial release.

www.latticesemi.com