CONTROL TECHNOLOGY CORPORATION

5300 EtherCAT Slave Controller

5300 EtherCAT Slave Controller Guide

January 3, 2023

Blank

> ⚠️ **WARNING:** Use of CTC Controllers and software is to be done only by experienced and qualified personnel who are responsible for the application and use of control equipment like the CTC controllers. These individuals must satisfy themselves that all necessary steps have been taken to assure that each application and use meets all performance and safety requirements, including any applicable laws, regulations, codes and/or standards. The information in this document is given as a general guide and all examples are for illustrative purposes only and are not intended for use in the actual application of CTC product. CTC products are not designed, sold, or marketed for use in any particular application or installation; this responsibility resides solely with the user. CTC does not assume any responsibility or liability, intellectual or otherwise for the use of CTC products.

The information in this document is subject to change without notice. The software described in this document is provided under license agreement and may be used and copied only in accordance with the terms of the license agreement. The information, drawings, and illustrations contained herein are the property of Control Technology Corporation. No part of this manual may be reproduced or distributed by any means, electronic or mechanical, for any purpose other than the purchaser's personal use, without the express written consent of Control Technology Corporation.

The information in this document is current as of the following Hardware and Firmware revision levels. Some features may not be supported in earlier revisions. See www.ctc-control.com for the availability of firmware updates or contact CTC Technical Support.

Model Number	Hardware Revision	Firmware Revision
5300 Controller	All Revisions	>= V050090R70.39

TwinCAT® is a registered trademark and EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

Windows® is a registered trademark of Microsoft Corporation, Redmond, WA.

Blank

Table of Contents

Blank

CHAPTER

1

[1] EtherCAT Slave Modes

Overview

The 5300 PLC Controller is capable of being both an EtherCAT Master or a Slave. As a slave device the 5300 has been tested with CTC's Incentive product suite, Beckhoff's TwinCAT, as well as passing the EtherCAT Technology Group's Conformance Testing Software suite to which Control Technology subscribes to a yearly license to ensure continued compatibility.

There are a number of modes of slave operation, all of which allow the QuickBuilder programming language to run simultaneous with Master operations although each has its own way of controlling IO and exposing variable data to the EtherCAT Master. They are:

- CTC_Standalone
- CTC_QBMode
- TwinCAT_Standalone
- TwinCAT_QBMode

Whenever a program is downloaded to a slave controller QuickBuilder will use the above program modes and match that to what is set in the controller. Discussed in the next section, there is a slave mode selection that is stored in the controller EEPROM so that at boot it knows what mode it will be running in. During the program download process if the mode does not match that of your program QuickBuilder will offer to change the setting for you and request you reboot the controller after a download.

CTC_Standalone

CTC_Standalone is used only with the CTC Incentive EtherCAT Master where all controller IO is public and under control of the Master. No register or variable data is available cyclically although a QuickBuilder program can run in the controller, it just can not write to the outputs or what is written will be overridden when the next cyclic update occurs from the Master. A special mapping file, discussed later, called QB_ECAT_Slave.txt is deleted from the controller project directory during a QuickBuilder program download. When the controller is offline from the Master VBIAS of the controller is turned off and outputs are not active. If the slave detects the Master offline from the slave all mapped output IO will be set to 0.

CTC_QBMode

CTC_QBMode is used only with the CTC Incentive EtherCAT Master. By default, nothing is public to the Master and only that which is programmatically selected within QuickBuilder is available. All other IO and data are local only to the QuickBuilder program making it a fully intelligent controller. The local QuickBuilder program and the remote EtherCAT Master can share an interface to differing outputs, although the same inputs can always be shared by each. When developing a QuickBuilder program there is an option on each IO as to whether it will be made available to the EtherCAT interface or remain private. The same applies to QuickBuilder variables where they can be accessed as Read, Write, or both Read/Write by the EtherCAT Master. Tables of variables are not supported although scalar variables of the type integer, 32-bit float, and 64 bits double can be made available as part of the cyclic Master data. A special mapping file, discussed later, called QB_ECAT_Slave.txt is deleted from the controller project directory during a QuickBuilder program download. If the controller detects the Master offline VBIAS remains on thus the outputs are active and all mapped IO and variables are set to zero. It is suggested the application also run a variable incremented by the Master application program that can be checked periodically by the local slave application. This not only ensures detection of the Master operating but also that its application program is active. The slave QuickBuilder program runs regardless of the state of the Master. If the slave detects the Master offline from the slave all mapped output IO and variables will be set to 0.

TwinCAT_Standalone

TwinCAT_Standalone can be used by any EtherCAT Master other than that of CTC. This mode makes all controller IO public and under the control of the Master. A unique exception from that of CTC_Standalone is if a QuickBuilder program is also running within the controller (possibly only controlling motion or other communications), variables from that program can be selectively accessed by the Master. The 5300 controller contains 8 slots for hardware modules. In order to support variable access up to an additional 8 slots can be used to automatically create virtual register access modules. Each module supports up to 16 read and 16 write slots in which variables can be mapped allowing for up to 128 public read and 128 write variables. Each read and write variable is treated independently. QuickBuilder will create the proper EtherCAT ESI xml file allowing the Master to reference the variables by the same name as that used within your QuickBuilder program. Variables may be integers (DINT), 32-bit floating point (REAL), or 64-bit precision doubles (LREAL). Table data is not supported. A special mapping file, discussed later, called QB_ECAT_Slave.txt is downloaded to the controller project directory along with any QuickBuilder program. If the slave detects the Master offline from the slave all mapped output IO and variables will be set to 0.

TwinCAT_QBMode

TwinCAT_QBMode can be used by any EtherCAT Master other than that of CTC. This mode makes all controller IO and registers/variables private to its QuickBuilder program and only those selected within the program are made available to the EtherCAT Master. Once again, the 8 virtual module slots are available. Any IO or register/variables selected within the QuickBuilder program as being public to the Master will be made available as part of the cyclic data. A special mapping file, discussed later, called QB_ECAT_Slave.txt is downloaded to the controller project directory along with any QuickBuilder program. It is suggested the application also run a variable incremented by the Master application

program that can be checked periodically by the local slave application. This not only ensures detection of the Master operating but also that its application program is active. The slave QuickBuilder program runs regardless of the state of the Master. If the slave detects the Master offline from the slave all mapped output IO and variables will be set to 0.

In all modes asynchronous SDO reads and writes can be used to access register and IO information. Reference the object definition section for further details.

Slave Mode Selection

By default, when the 5300 detects the Slave EtherCAT option module installed, it will make all IO public and map it as available as cyclic PDO data (controller Mapping Modes disabled). With a do nothing QuickBuilder program installed the controller is basically a slave to the EtherCAT Master.

At the controller level there are two alternate mode options other than none. CTC QB Mode, or TwinCAT Mode with each acting slightly different depending upon the QuickBuilder program loaded within the controller. Note that it is also important that the QuickBuilder program controller's ECAT Mode property be set properly to match that of the controller. For example, if the QuickBuilder program lists a number of variables to be made available to the Master but all "EtherCAT Mapping Modes" are disabled in the controller then only IO would be available to the Master. Since the controller must be able to respond to the EtherCAT Master properly during initialization the QuickBuilder program is not necessarily running at this time, hence the PDO mapping is done prior to a program running in TwinCAT mode. QuickBuilder will verify and offer to change the setting to match that of the program during the download process.

When the 5300 is running as an EtherCAT Slave, but in QB Mode, the resources that are public to the Master are selected during the creation of the Slave QB program. In Standalone mode all IO is public and controlled by the Master (such as TwinCAT or the Incentive Master) but in QB mode only the selected resources are public. Resources can have read, write, or read/write Master access selected, this includes registers, both integer and variants (string, constant, and array variables are not supported). In order to enable QuickBuilder mapping of IO and variables the controller must first be placed in that mode and rebooted. This is done by first logging in with Telnet and entering the appropriate command (case sensitive):

enable ethercat ctc qb mode

In CTC QB Mode the controller will allow QuickBuilder full control of all IO and only the IO and registers/variables selected within your programs will be mapped to the Incentive EtherCAT PDO cyclic data as read only, write only, or read/write. This is only for use with a CTC EtherCAT Master.

**QuickBuilder selected Slave
IO and registers/variables**

enable ethercat twincat mode

In TwinCAT QB Mode the controller will allow QuickBuilder full control of all IO and any IO and registers/variables selected as EtherCAT Slave within your program will be mapped to a virtual M3-REG module in slot 8 to 16 of the controller, shifted down from slot 16 as space fills. When QuickBuilder is translated a file called QB_ECAT_Slave.txt is created which when loaded into the controller along with your QuickBuilder program is read by the controller and mapped public for the EtherCAT Master PDO cyclic access. There is also a utility within QuickBuilder, discussed later, which will generate the EtherCAT ESI xml file for access to all the 5300 controllers in the network. This controller mode should only be used if the QuickBuilder program has TwinCAT_QBMode mode set within the controller properties.

QuickBuilder selected Slave IO

+ QB_ECAT_Slave.txt contents if TwinCAT_QBMode is set in your QuickBuilder program as virtual M3-REG modules in controller slots 9 to 16, shifting down from 16.

disable ethercat mapping mode

This command sets the controller back to the default where all IO is mapped to the EtherCAT Master. If TwinCAT_standalone is selected, within your QuickBuilder program, the contents of the QB_ECAT_Slave.txt file is also downloaded with your QB program and will be included in the PDO map.

All IO

+ QB_ECAT_Slave.txt contents if TwinCAT_standalone set in your QuickBuilder program as virtual M3-REG modules in controller slots 9 to 16, shifting down from 16.

Saving Settings
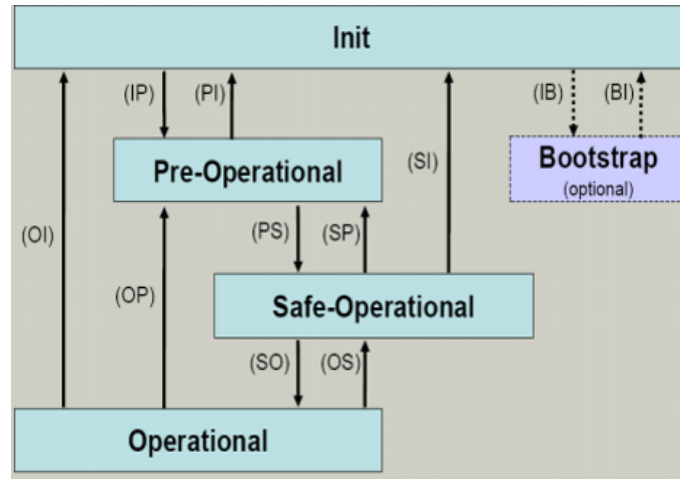
To write the changes to non-volatile memory:

        20096=1

Then reboot the controller to have the changes take effect:

        mon reboot

5300 EtherCAT Slave Registers

State Machine:

The following are registers that are useful in diagnosing EtherCAT network problems on the slave controller, modifying scan rates, and restricting PDO information.

13056 – (Read Only), Requested prior state from the EtherCAT Master unless INIT which is the default at power up and reset.

13057 - (Read Only), Present EtherCAT state where upper nibble represents the state the Master is requesting and the lower nibble the current Slave state. Example 84 would be the Master requesting OPERATIONAL state while the Slave is in SAFE-OP. Example of 88 would be the Master requested OPERATIONAL and the Slave is in the OPERATIONAL state.

13058 – (Read Only), Last AL Error. This would be if an error occurs when the Master requests a change of state of the Slave.

13059 – (Read Only), 5300 Slave mode where 1 is standalone where all IO is mapped to the Master and 0 is QuickBuilder mapping mode, where only the IO and registers selected within your QuickBuilder program are mapped. By default, the 5300 runs as a standalone EtherCAT Slave device. QuickBuilder may run but it does not control what is made public. To enable QuickBuilder mapping you must set the mapping within your program as well as run the telnet command "enable EtherCAT QB Mapping". This is needed so that if a QuickBuilder program is not loaded but QB Mapping is desired, the Slave will remain in the INIT mode and not make its resources public at power up.

13060 – (Read Only), -1 if not being used, any other value is the number of missed PDO cycles. A missed PDO cycle occurs when the Slave checks for refreshed scanned data and none is available. Typically, this is used to detect the Master being shut down. After a set timeout the Slave will disable output voltage on the outputs and default to the INIT state until the Master is back online.

13061 – (Read/Write), Rate in milliseconds that the Slave reads and writes all Master data. The default is 10 ms and can be set as low as 4 ms. The time it takes to read and write the Master data is calculated and

subtracted from this value and the remaining value is how long the scan thread sleeps, allowing other threads in the controller to run. The faster the scan the less time for communications and QuickBuilder programs to run.

13062 – (Read Only), the number of milliseconds it took to refresh the Master data (PDO) area. Typically, 1 to 2 milliseconds depending on the amount of data that needs to be updated.

13063 – (Read/Write), the maximum number of milliseconds it has taken to interact with the EtherCAT Slave logic in a scan cycle. This will be a larger number during initialization. To get a more accurate number zero out this register and monitor it during the OPERATIONAL state.

13064 – (Read), set to 1 if a QuickBuilder program is loaded and has mapped IO and memory variables for public EtherCAT access.

13065 – (Read/Write), used by a QuickBuilder program that is running in EtherCAT QB Mode. Normally this register is set to a default value of 3 where the Slave interacts with the Master normally. If set to a 0 the Slave will respond to EtherCAT requests but Master PDO data will remain unchanged and not be updated or processed. Set to 1 for the Master PDO received data to be updated but the writes to the Slave to be ignored. Set to a 2 for the Slave to process write requests but not update any PDO read data.

13066 – (Read Only), TxPDO size. The number of bytes read (transmitted) from the Slave to the Master within the cyclic PDO packet.

13067 – (Read Only), RxPDO size. The number of bytes written (received) to the Slave from the Master within the cyclic PDO packet.

[2] Incentive CTC QBMode

QB Mode Variant Table

The Master QB program can access the slave public registers using the special variant register 36840 for reads and 36841 for writes. These registers act as a table where the slave row is the EtherCAT slave number (minus 1 since 0 based and first slave is 1), the first column is the number of variables available in the row (1 to N). The second column, column 1, is the start of public variable storage starting with integers and followed by any floats and doubles. QuickBuilder has tools to display the slave mapping scheme for access based upon what is made public. The property of the variable shows the ECAT_RD_IDX and ECAT_WR_IDX, from the Master's perspective. The assigned index is what the Master QuickBuilder program would use in the variant array (36840 for reads and 36841 for writes) to access that particular variable. Note that if you change what is mapped public and then translate your program the IDX values could change, thus verify using the 'resource grid view' and use constants in your program when reference index values, that way you only have to change the constant value to the new index value. XVAR constants can be automatically imported into your program based on the mapping as discussed in the next section.

As an example, if a slave had an integer register 500 public, and one float 36705 and one double 36706 then the columns would map as follows:

Row (Slave #)	Column 0	Column 1	Column 2	Column 3
0 (slave #1)	3	Integer #500	Float 36706	Double 36706

The QuickBuilder example below shows a register named 'counter' with an override register of 500 assigned and an ECAT_RD_IDX of 1, thus it would be accessed by the EtherCAT Master as 36840[slave #][1]. The IDX is assigned during translation and can also be viewed using the Resource Grid.
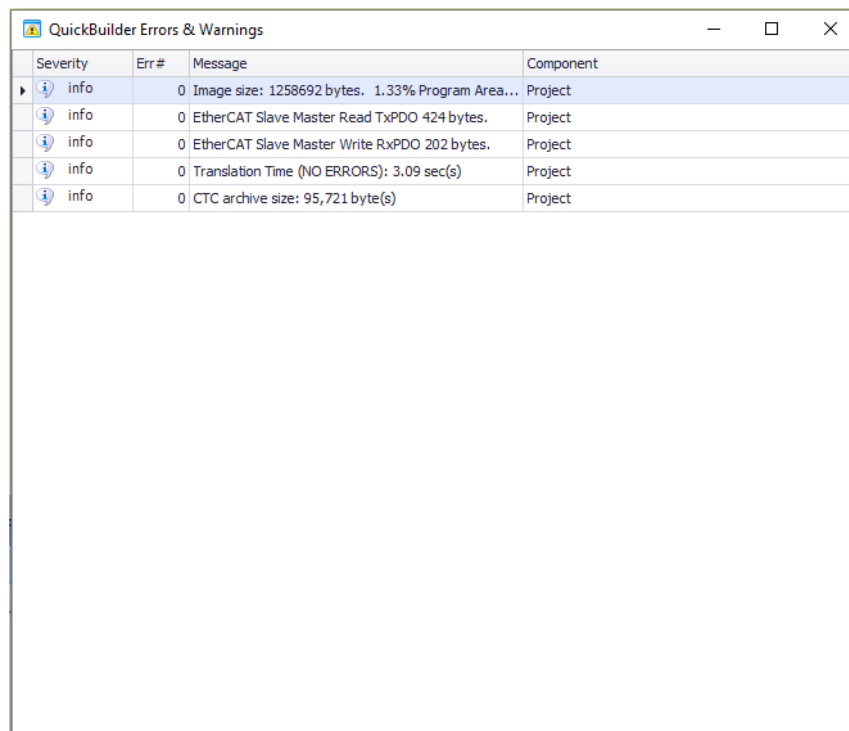
Each slave can have its own unique set of public registers and the only slaves currently with valid data are those that are 5300's running in Slave QB mode. Accessing a location with no data will return a 0. Note that modifying your Slave program and making other variables public and redefine the index during translation since integers are first, followed by floats, and then doubles. The next section will detail a method built into QuickBuilder that will allow you to reference the slave column data index as an XVAR constant. This allows you to program your Master application program and not have to constantly change an index should the slave program change, simply import its public register file and the assigned indexes will automatically change for you.

Master Application Program Import of Slave Register Indexes

When the 5300 is running as an EtherCAT Slave, but in QuickBuilder mode (CTC_QBMode), the resources that are public are saved in a comma delimited file, QB_ECAT_Slave.txt. You will also see that the QuickBuilder Errors & Warnings dialog that appears after a translation will give you a rough idea of the size of the TxPDO and RxPDO that the slave will be place into the Ethernet packet as it chains between devices. For all resources on the network the total should be less than 1500 bytes. Below would be 424 TxPDO bytes + 202 RxPDO bytes, or 626 total bytes usage. Note that if you are using Incentive PC as all your devices this can be much bigger as chained packets are allowed. Chained packets are not supported within the 5300 M3-41A EtherCAT Master module due to memory limitations.

Created during translation is the file QB_ECAT_Slave.txt file, stored within a subfolder of your project using the following naming convention: 'project name\controller name'. A sample record would be:

reg10, Variable, 10, int, read, 2

reg10 – Name used within the program for the variable

Variable – Either Variable or NVariable
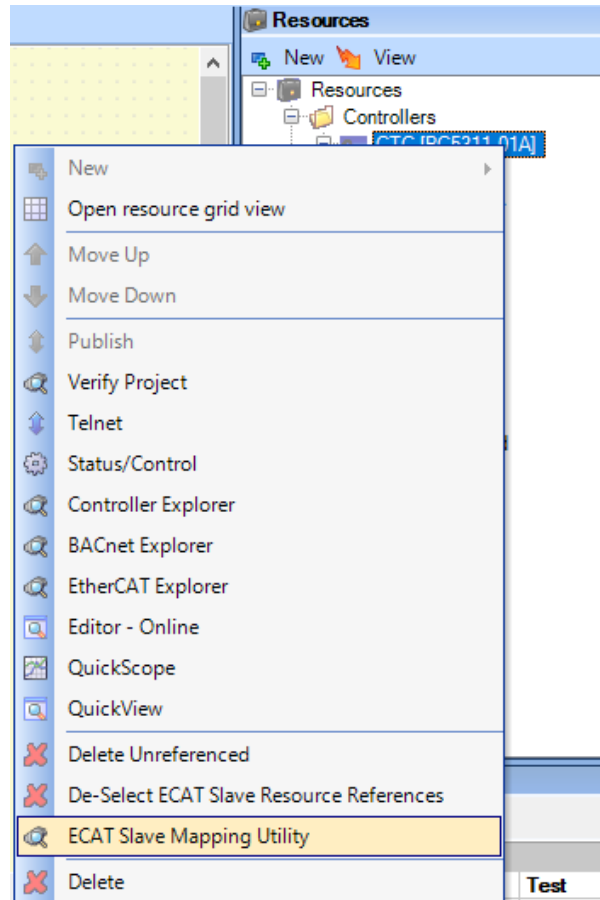
10 – Register number used by the slave program

Int – Data type, in this case integer, options are 'int, float32, or float64'.

read – Read or write access (36840 or 36841 variant at Master)

2 – Index location for access by the EtherCAT Master using the 36840/36841 variant table.

The Master QuickBuilder application program can import any number of these files as XVAR constants. This allows a Master program to reference the slave registers even when the slave program has changed by using constants that can be updated through an import utility. Right click on the controller's name and select the "ECAT Slave Mapping Utility" menu item:

The following form will appear:

First browse to the slave controller directory where the "QB_ECAT_Slave.txt" file exists for your project and upon selection information about all public variables will be displayed:

Upon selection:

Now prior to import you need to come up with a unique name to start the constant you want to create that will reference the index to be used when referencing the 36840/36841 variant table. For test purposes we will use PLC_01. Upon import the name will become "Your header" + "_" + "slave register name" + "_" + "read or write".

For this example, we will enter PLC_01 into the text field that says "Slave register header for Import" and then select the "Import Map" button. Select Yes when prompted are you sure:
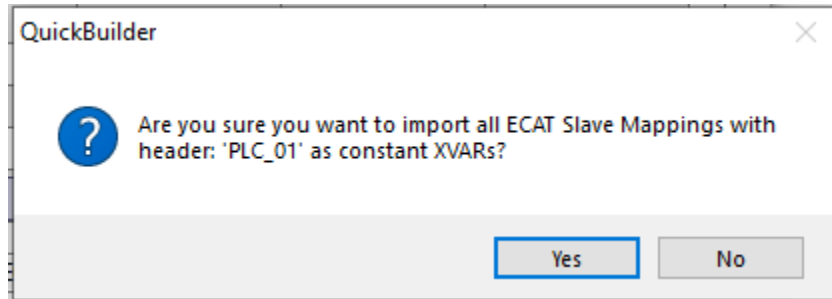
Once imported you will be directed to check the XVAR folder:
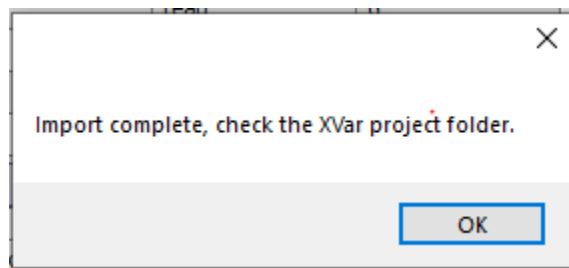
The symbols will appear within the XVars folder, they are in red since they are not reference by the program yet:
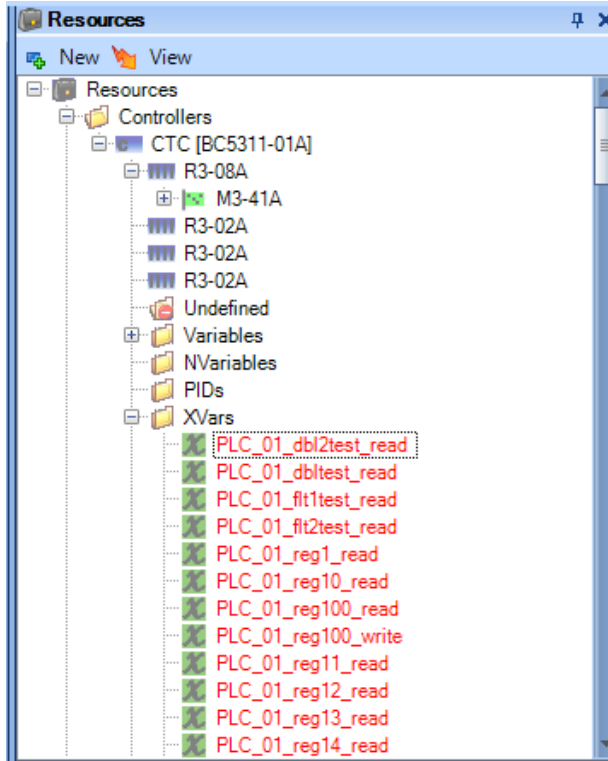
To read a variable named reg10 on slave 1 and store it to a local variable a sample line of code would be:

```
i = ECAT_READ[0][PLC_01_reg10_read];
```

The row is set to the slave number (minus 1 since zero based) and the column to the proper index generated by the slave file import. Note that the row number may also be a variable instead of the constant '0' used in the example.

If you look at the properties of the symbol you will see that the value of the constant matches that of the comma delimited file for register 10, index of 2.
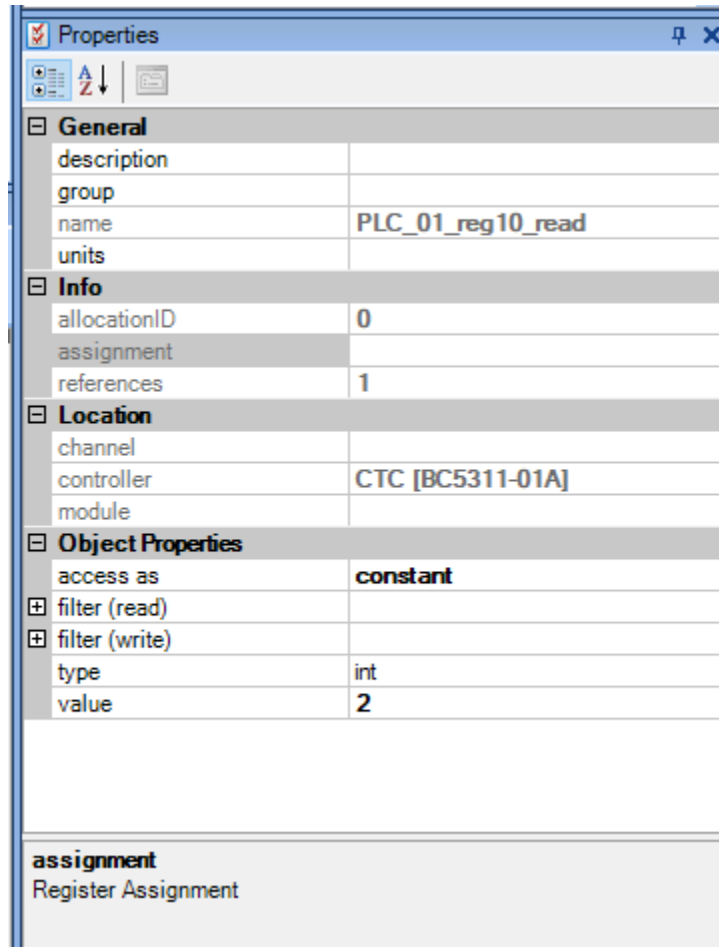
ECAT_READ is a variable you create in your program with an assignment of 36840, storage of table and type of any. ECAT_WRITE would be the same but with an assignment of 36841, storage of table and type of float64. You may rename these references to anything you like for your program.

Blank

CHAPTER

3

[3] EtherCAT ESI File

Creating TwinCAT and Generic ESI xml File

EtherCAT Masters, other than that of CTC's Incentive, require an ESI xml slave information file that supplies it with mapping and object information. There are two types of ESI files available, that of a simple slave device with only IO exposed (CTC5300.xml, installed as part of QuickBuilder at the following location: C:\Program Files (x86)\Control Technology Corporation\QuickBuilder) or an ESI file generated from within QuickBuilder that appends register and variables that are to be included in the cyclic PDO to the base CTC5300.xml file creating an alternate file called CTC5300_QB.xml. CTC5300_QB.xml can also be used when standalone and QB Mode slaves are mixed. To generate an ESI file for your Master based upon a QuickBuilder program you have written you use the ECAT Mode controller property, setting it to either TwinCAT_Standalone or TwinCAT_QBMode. The controller non-volatile memory should also be set to TwinCAT Mode as discussed in the previous section.
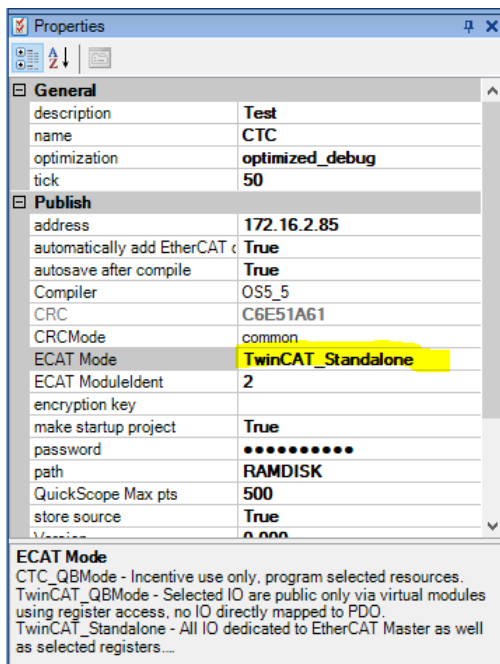
Once a QuickBuilder program is complete you can then create the EtherCAT ESI file by right clicking on the controller and selecting "ECAT Slave ESI Creation".

	New ▶
⊞	Open resource grid view
⬆	Move Up
⬇	Move Down
⚡	Publish
🔍	Verify Project
⬆	Telnet
⚙	Status/Control
🔍	Controller Explorer
🔍	BACnet Explorer
🔍	EtherCAT Explorer
⬛	Editor - Online
🖼	QuickScope
⬛	QuickView
✖	Delete Unreferenced
✖	De-Select ECAT Slave Resource References
🔍	ECAT Slave Mapping Utility
🔍	**ECAT Slave ESI Creation**
✖	Delete
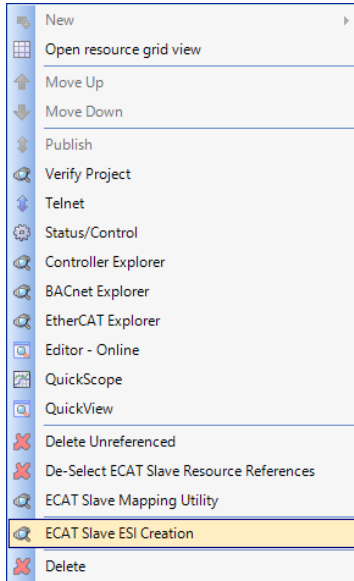
The below form will appear:
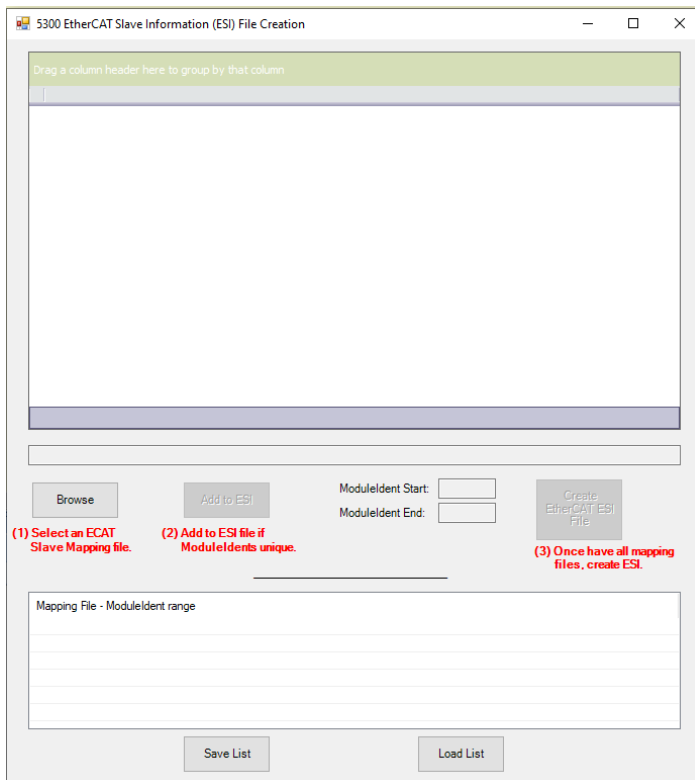
The first step in generating the ESI file is to select the QB_ECAT_Slave.txt file within each controller's directory. The controller's name is the name of the folder found in the Project folder directory. This is a simple comma delimited file listing all the variables and registers that will be mapped into the cyclic PDO data.
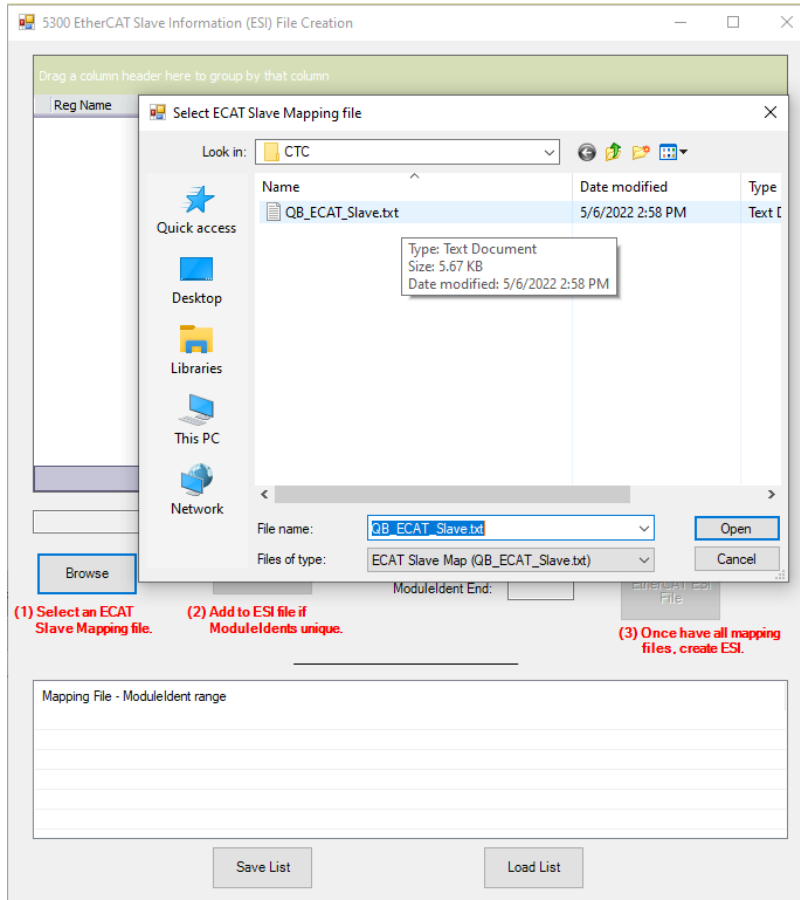
Once the file is selected all of the register and variable names used within your program will appear, including their assigned registers, data type, access type (read or write), as well as index. The index consists of two numbers separated by a dash. Following the ESI specification each module defined must have a unique module identification number or ModuleIdent. That is the first number and references the ModuleIdent set in the property section of the controller program. By default QuickBuilder starts with 2 and automatically increments to a maximum of 9 for each controller. You must assign a unique ModuleIdent number in your QuickBuilder program. You may set this to anything from 2 to 9999. It is suggested that they be 8 counts apart but that is not necessary in a simple system, just good programming practice should your system later grow.

The ModuleIdent is the first number, the second is the slot in the module, from 1 to 16, with read and write treated independently. As previously mentions a controller can have up to 8 virtual modules. The virtual modules appear like a real IO module to the EtherCAT Master and are identified as M3-REG modules when viewed by TwinCAT scan functions.

Below shows the project slave variables that have been made available to the EtherCAT Master as marked on the property section of each variable within QuickBuilder. The starting ModuleIdent is 2 and the last is 5, meaning 4 out of the 8 virtual modules are being used. After reviewing the displayed variables click on the "Add to ESI" button to add them to the mapping list.
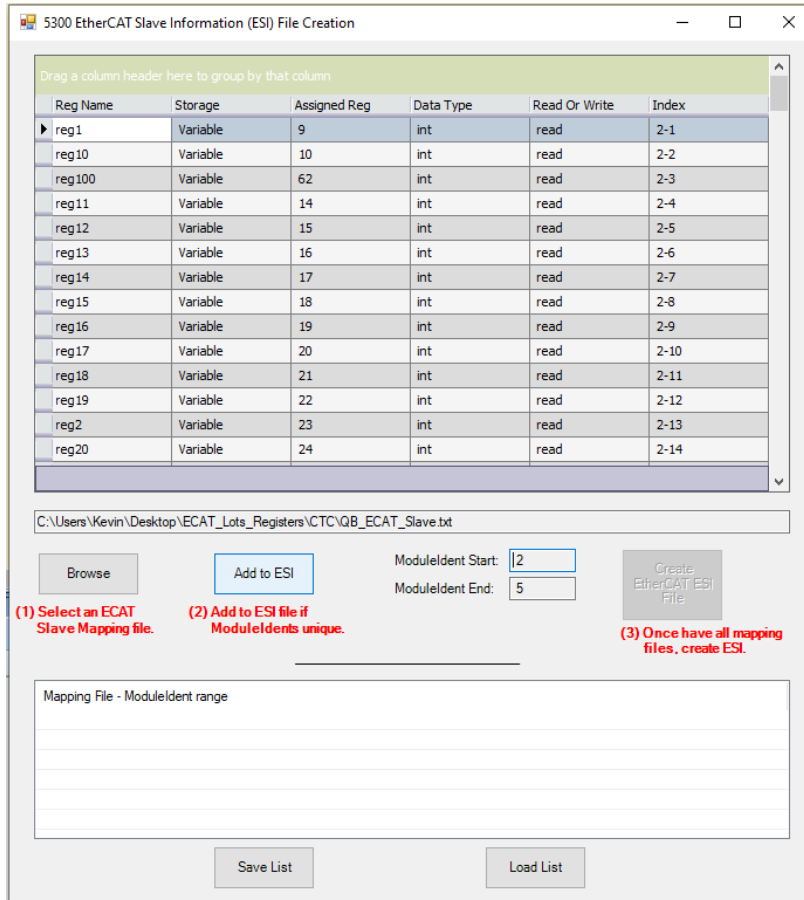
You will be presented with a verification dialog to make sure you wish to add this set of variables.
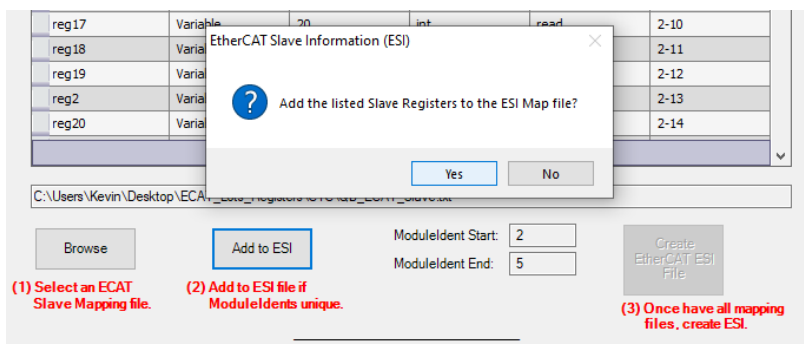
Once added you will see the referenced file as well as its ModuleIdents being used. If only one controller or multiple controllers with the same program are being used you may stop now.

If you wish to save what you have added temporarily and come back to it later you may click the "Save List" button. That highlighted in yellow will be saved to a file called CTC5300_Maplist.txt and placed in the project directory, one level above that of the controller's subdirectory. Use "Load List" to load it back and either create the ESI file or add more project/controller variable files.

Once all the desired EtherCAT slave QB_ECAT_Slave.txt files have been added click on the "Create EtherCAT ESI File". A properly formatted ESI xml file will be generated called CTC5300_QB.xml. This file should be placed in the proper directory for your EtherCAT Master's reference. For TwinCAT this is typically the "C:\TwinCAT\3.1\Config\Io\EtherCAT" directory.

Below is an example of this project loaded by TwinCAT as it scanned a 5300 controller, note the M3-REG modules:

[4] TwinCAT Project

Sample Project

The 5300 PLC Controller is capable of being either an EtherCAT Master or a Slave. As a slave device the 5300 has been tested with Beckhoff's TwinCAT product as well as has passed the EtherCAT Technology Groups Conformance Testing Software suite to which Control Technology subscribes to an annual license

This section will create a sample TwinCAT project to interact with the 5300 controllers as a slave device using QuickBuilder TwinCAT_Standalone mode, where all 5300 IO is available to TwinCAT. Once TwinCAT is installed the CTC EtherCAT ESI file, CTC5300.xml, should be copied to the "C:\TwinCAT\3.1\Config\Io\EtherCAT" directory. Invoke TwinCAT 3 and select "New TwinCAT Project" as highlighted below:
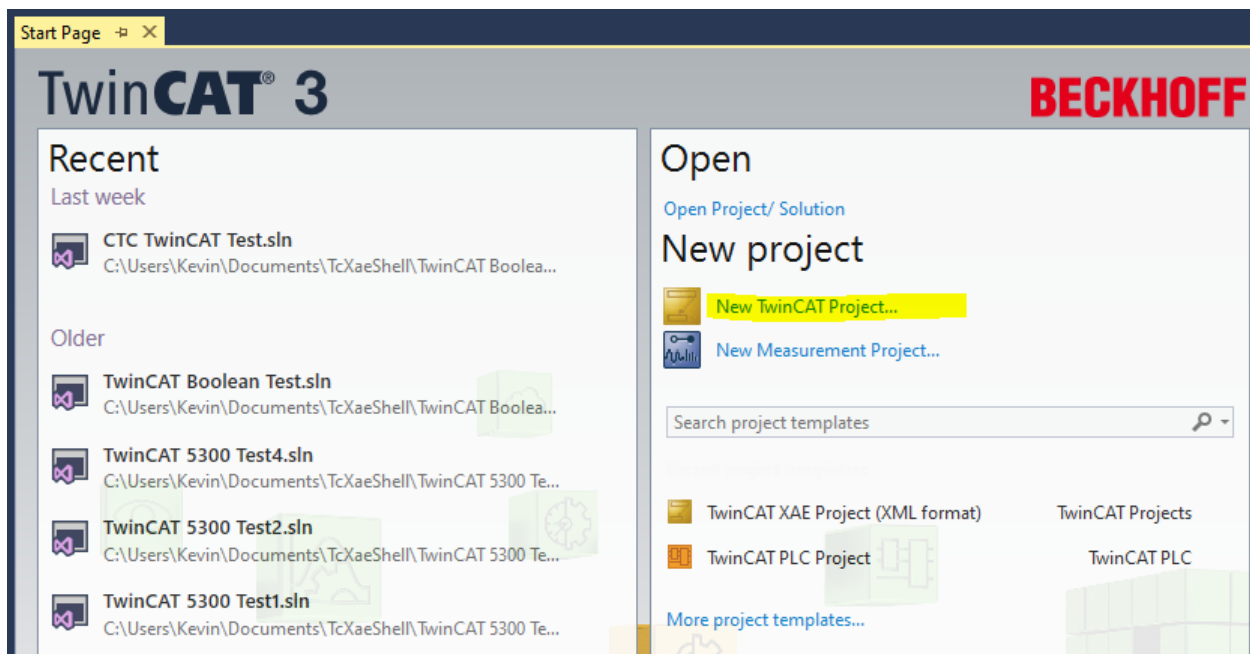
Set the desired project name and directory, in this example the project is called "CTC5300_Test_Project":
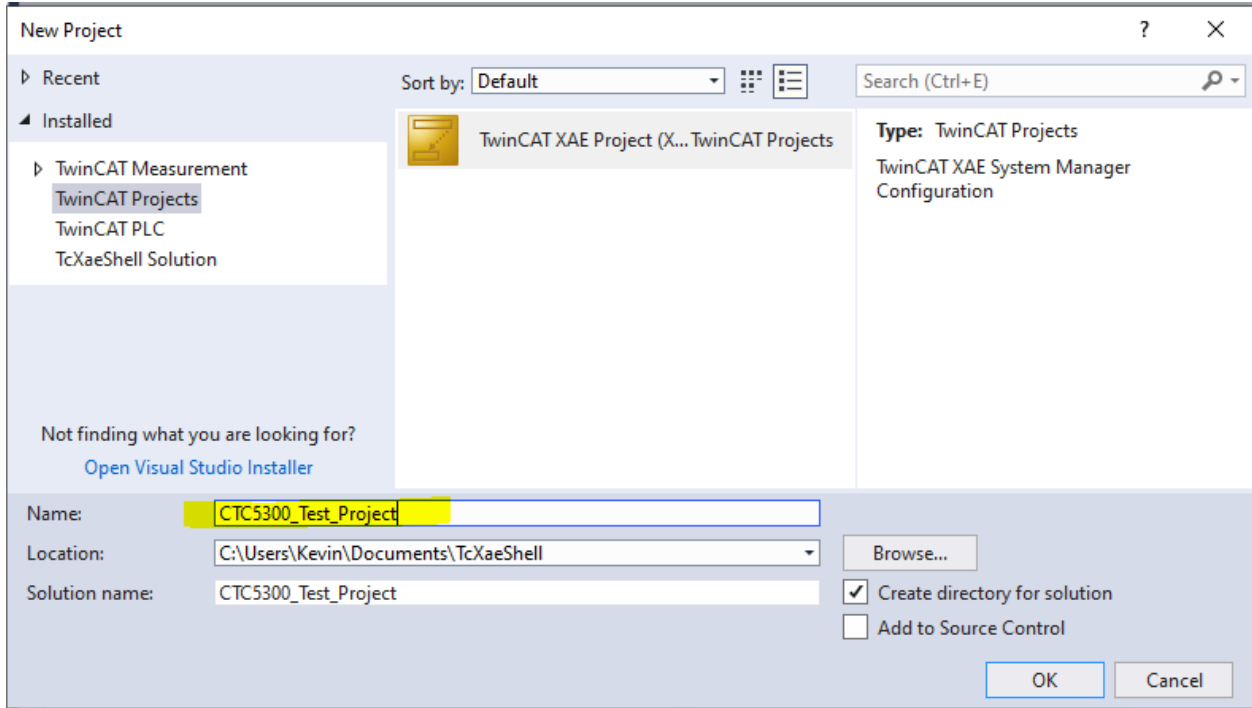
Within Solution Explorer will appear:
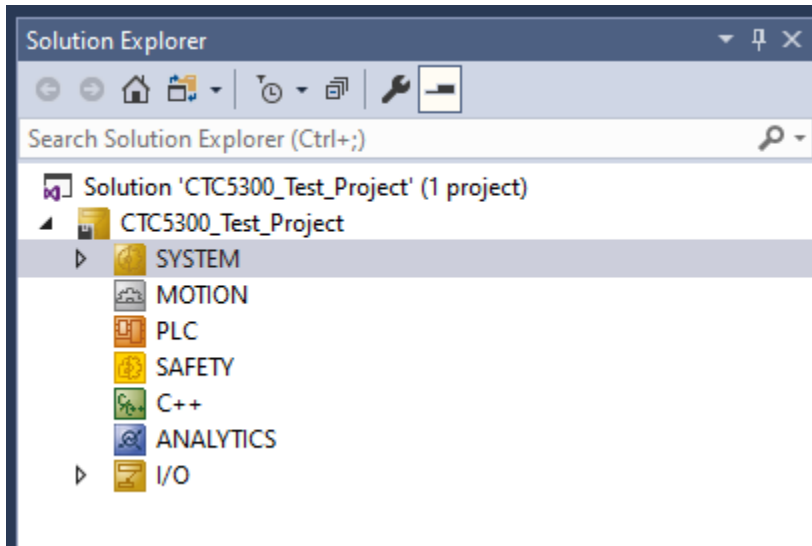
Assuming you are using a demo version, every 7 days a Security Code must be entered. To do so double click on "License" under System and the following will appear. Click on "7 Days Trial License" and enter the Security Code before proceeding:
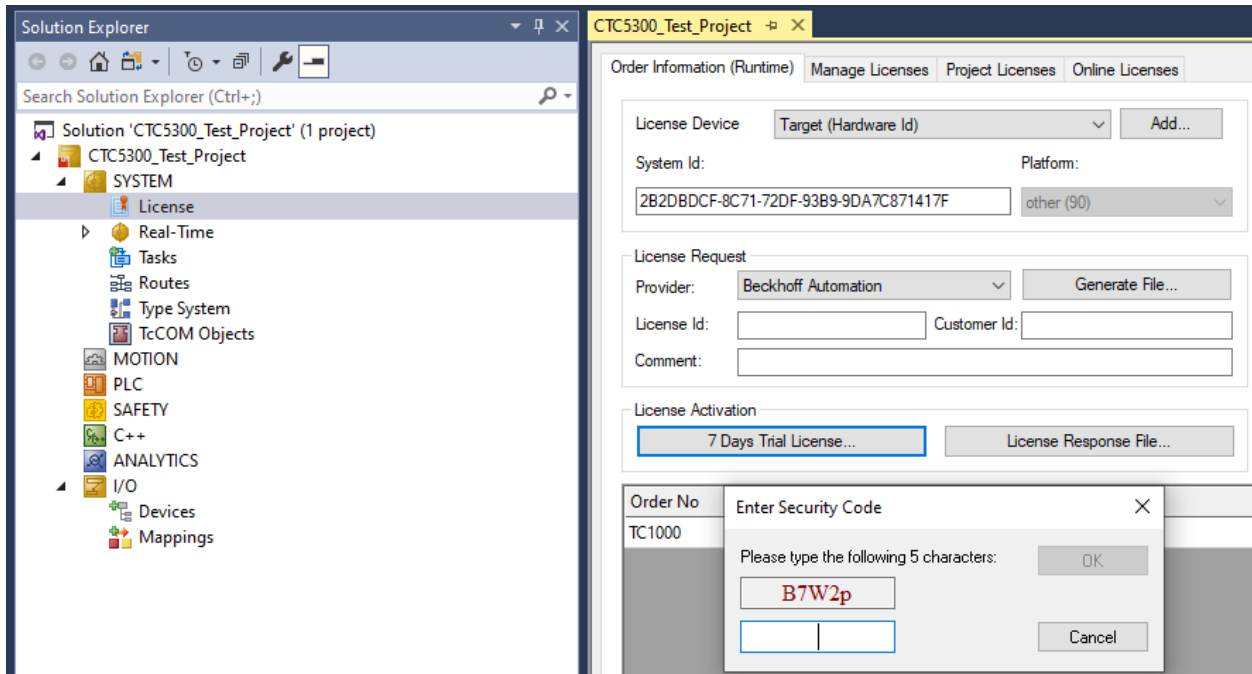
With a 5300 slave controller attached to the PC running TwinCAT the software can scan the network and determine what IO is available. The controller should be set to Standalone mode, where TwinCAT will control all IO.

Next, we will setup TwinCAT for EtherCAT operation. Note that you need two Ethernet ports on your PC typically, one for your main house network and the other for EtherCAT devices. Right click on I/O->Devices and select "Add New Item":
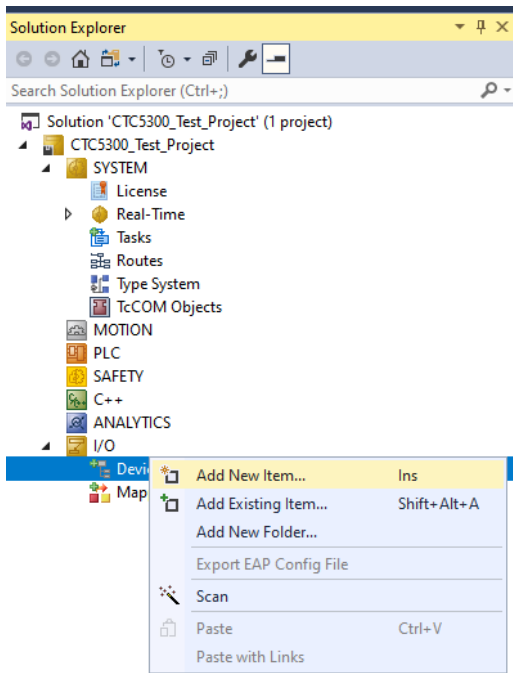
An Insert Device dialog box will appear, select EtherCAT Master and OK. If prompted as to which Ethernet port to use you should select the one your EtherCAT devices are on:
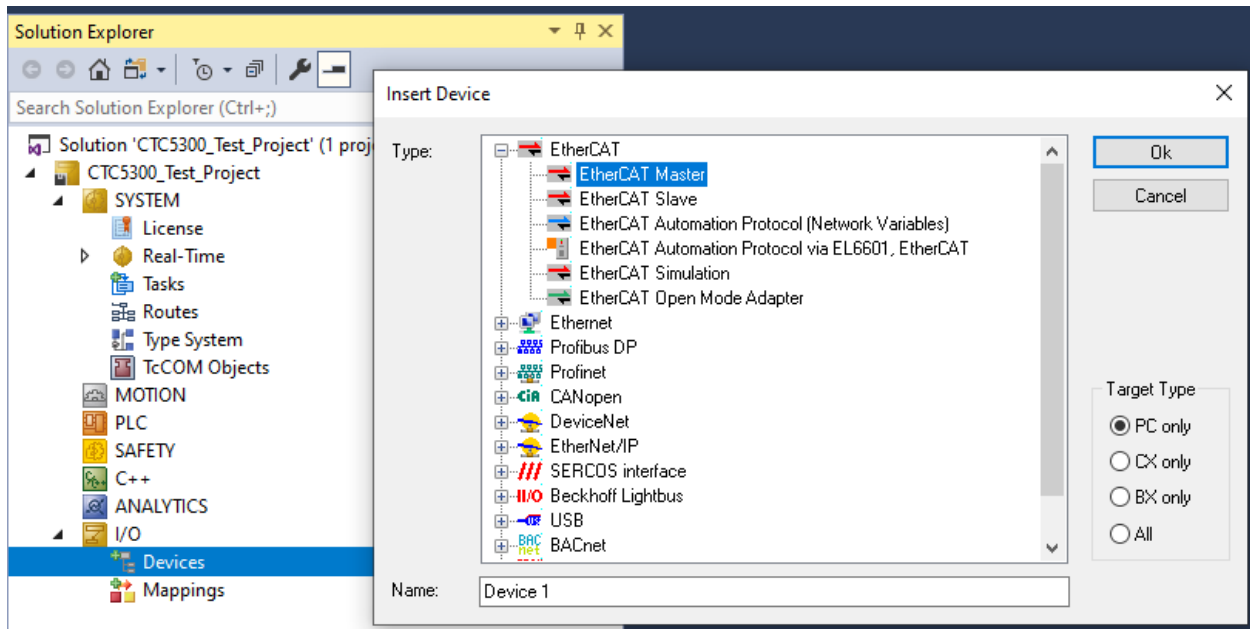
Right click on the EtherCAT Device and select 'Scan':
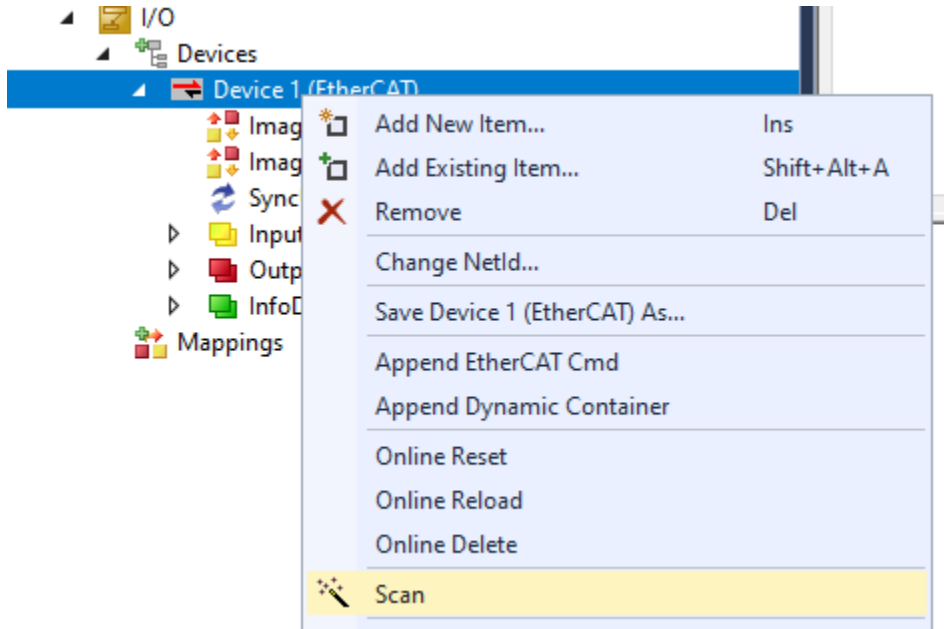
A dialog to select the Ethernet port may appear:
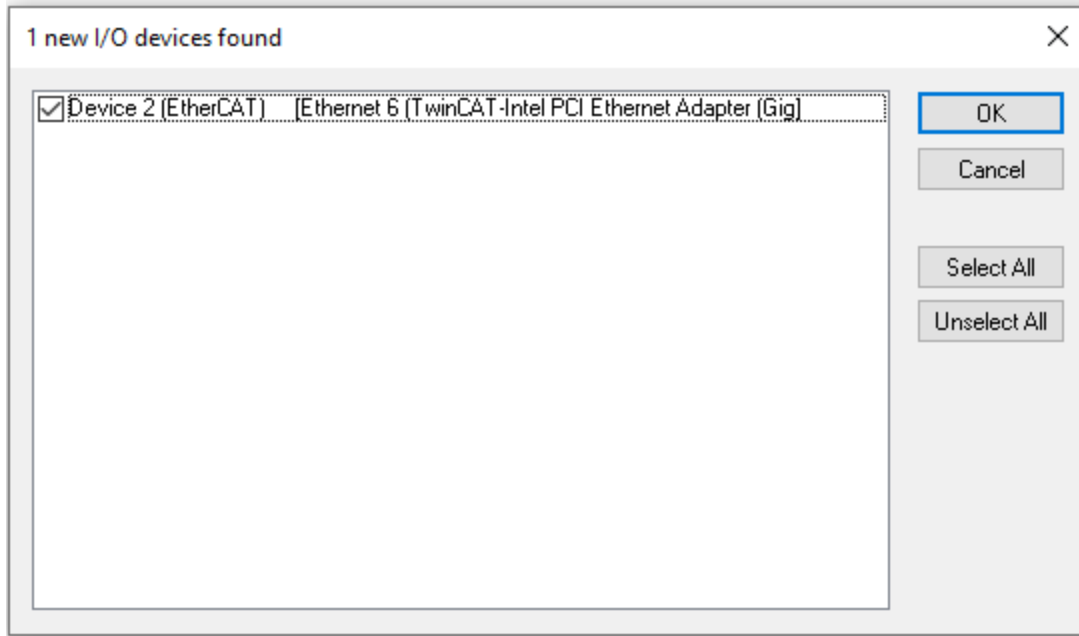
A CTC 5300 PLC is on the network in this example. All modules that are in the PLC rack will appear and can be assigned as TwinCAT IO:

Should you change the mapping in the controller within an existing project you may have to select the "Load PDO info from device" to refresh TwinCAT, should it not match what is expected. This is shown below on the Process Data tab.

To create a program right click on SYSTEM->PLC and select "Add New Item". A dialog will appear and select "Standard PLC Project" followed by Add:

The project in this example was called CTC5300. Expand and you will see the following:
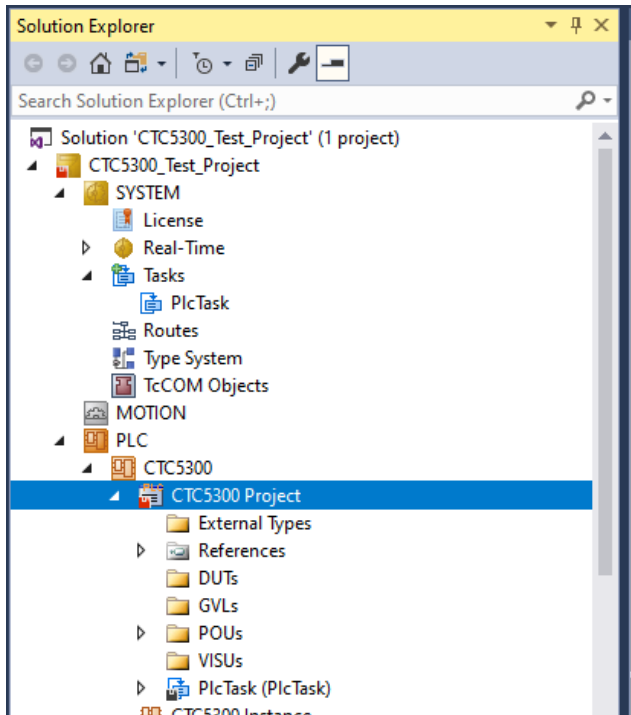
Right click "GVLs" and add a Global Variable List:
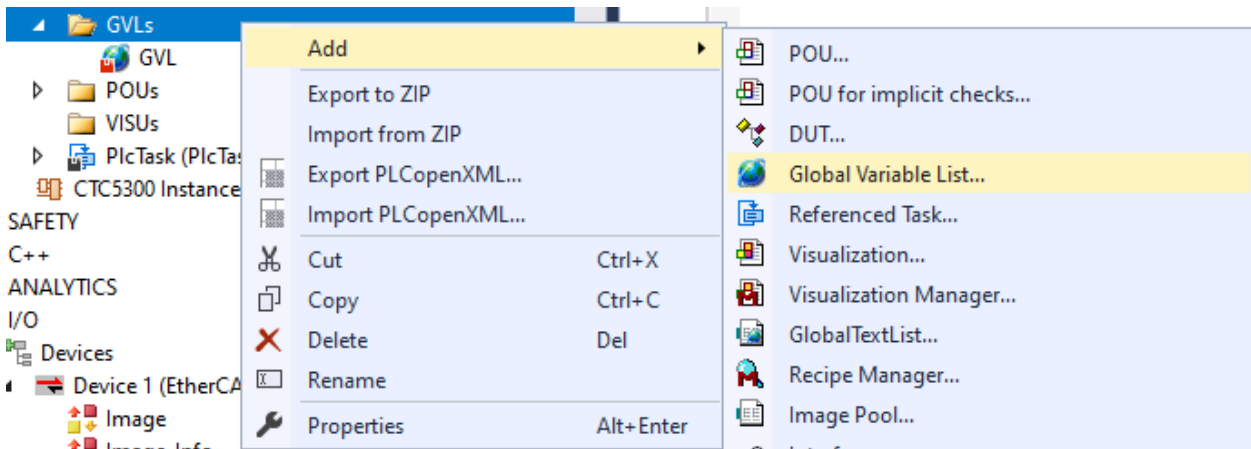
Select the Text view (top right of GVL window) and add the following Digital Input and Output variables. Our test will simply write data to the M3-10A outputs that are looped back to the individual inputs:

```
GVL* ┐ X
   1      {attribute 'qualified_only'}
   2      VAR_GLOBAL
   3          DigitalInput1 AT %I* : USINT;
   4          DigitalOutput1 AT %Q* : USINT;
   5
   6          DigitalInput2 AT %I* : USINT;
   7          DigitalOutput2 AT %Q* : USINT;
   8      END_VAR
```

```
MAIN* ┐ X  GVL
   1      PROGRAM MAIN
   2      VAR
   3          Inputs1 : USINT;
   4          Inputs2 : USINT;
   5          MyCnt : UINT := 1;
   6          aByteArray : ARRAY [0 .. 3] OF BYTE; // Array must start with LSB
   7          pt : POINTER TO UINT;
   8          My_Timer: TON;
   9      END_VAR
  10
  11

   1      My_Timer( IN:= NOT My_Timer.Q, PT:= T#5MS);   // Set timer for next update
   2      IF My_Timer.Q THEN
   3          MyCnt := MyCnt + 1;
   4          pt := ADR(aByteArray);   // Get the address of the byte array to store the integer counter to
   5          pt^ := MyCnt;   // Save integer counter to byte array
   6          // Write each byte of array to the 5300 outputs, one byte at a time
   7          GVL.DigitalOutput1 := aByteArray[0];
   8          GVL.DigitalOutput2 := aByteArray[1];
   9          // Read the first byte input back from the 5300 just for monitoring purposes
  10          Inputs1 := GVL.DigitalInput1;
  11          Inputs2 := GVL.DigitalInput1;
  12      END_IF;
```

Now that the program is entered the IO variables must be assigned to physical IO of the controller. Select "Change Link" for each of the PlcTask Inputs and Outputs.
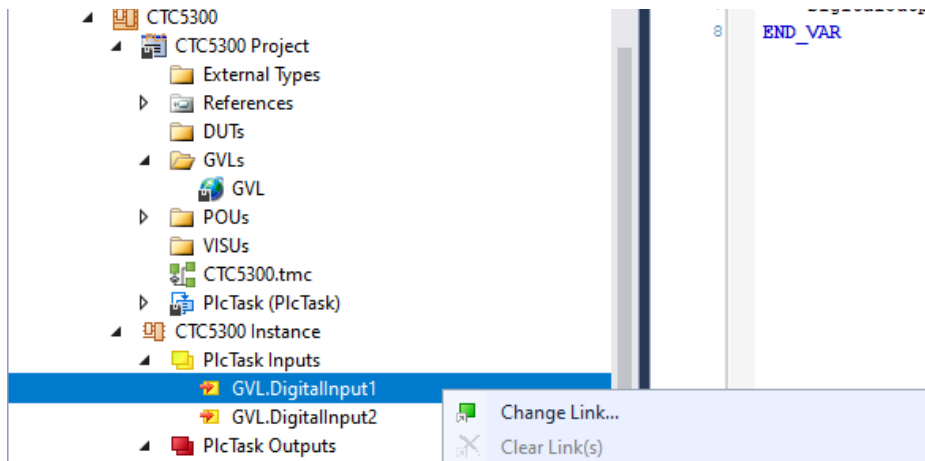
In the dialog box that opens select the checkbox under "Show Variable Types", "Matching Type". Assign Byte 0 to GVL.DigitalInput1 and Byte 1 to GVL.DigitalInput2.
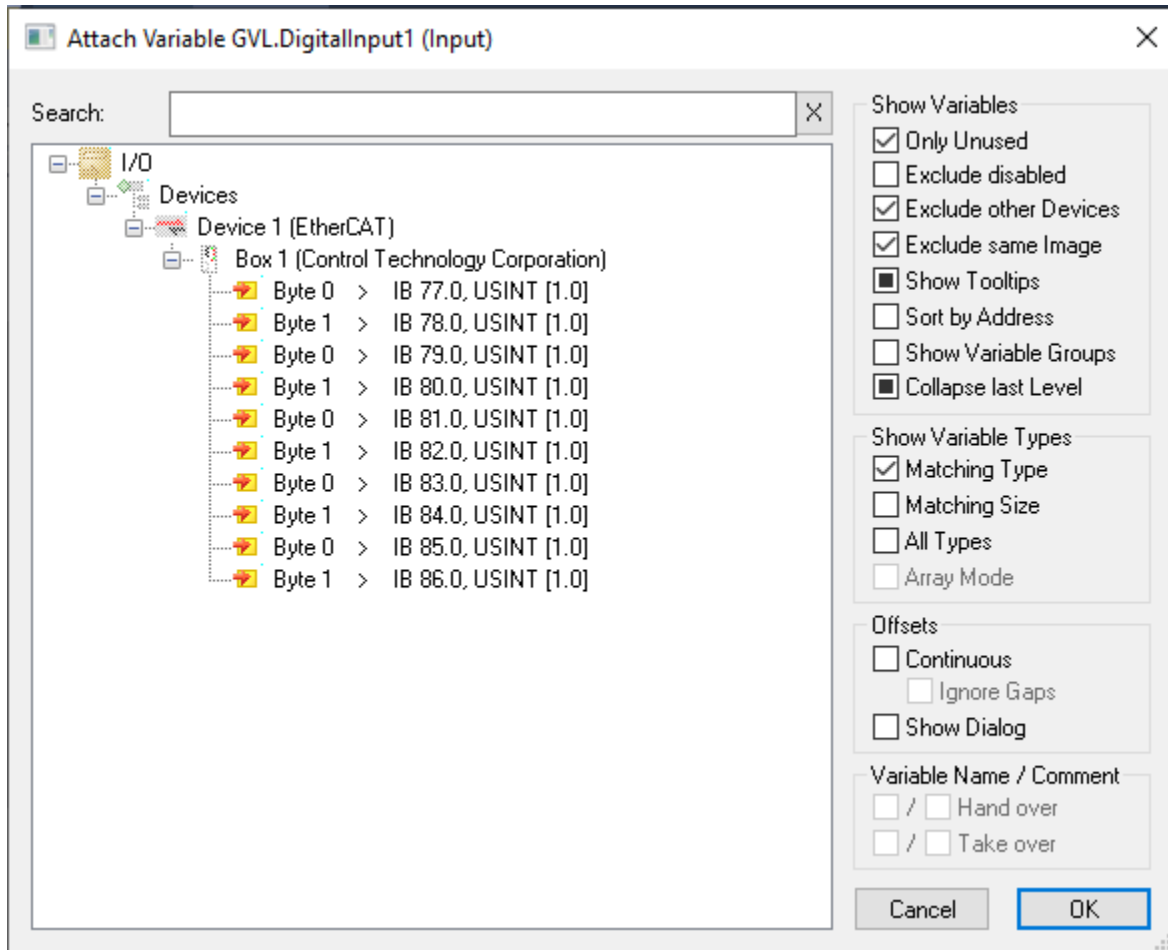
Now select the Change Link under the Plc Task Outputs and assign outputs as above except Byte 0 to GVL.DigitalOutput1 and Byte 1 to GVL.DigitalOutput2.

The program must be built. Right click on the Solution and select Build Solution:
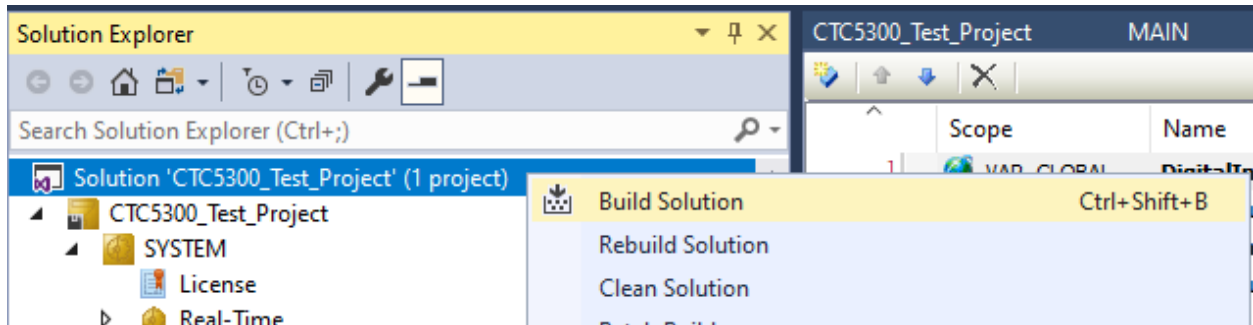
The output tab should show there are no errors:
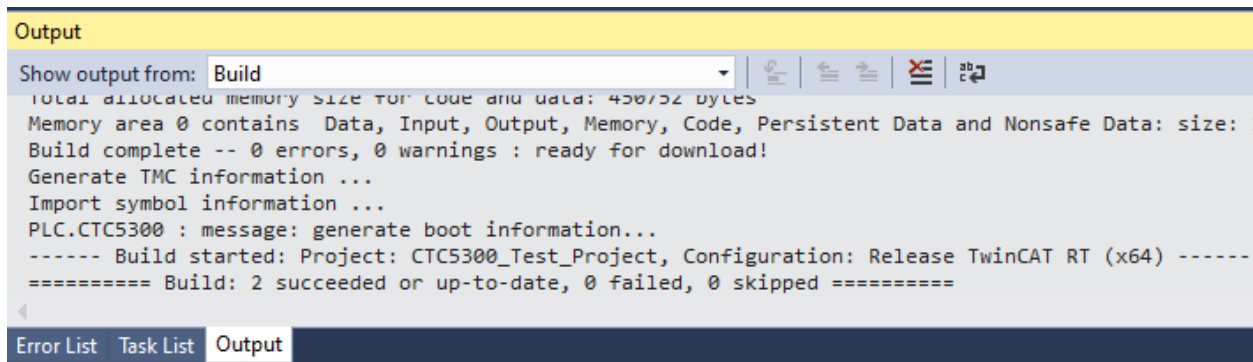
Click the "Activate Configuration" toolbar icon:
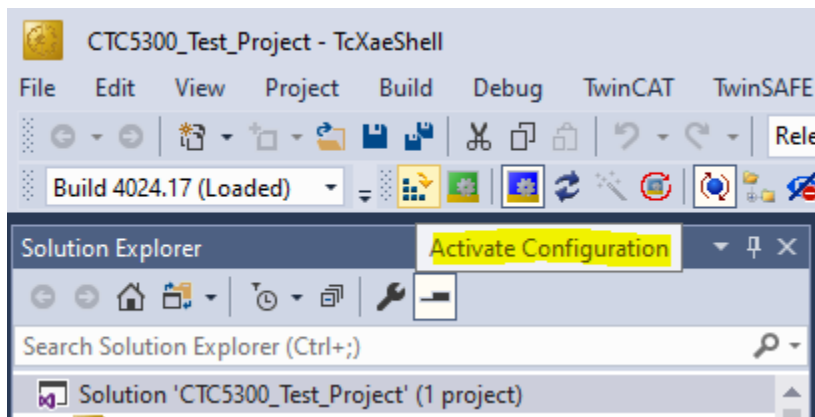
Select "OK" to the Activate Configuration dialog:
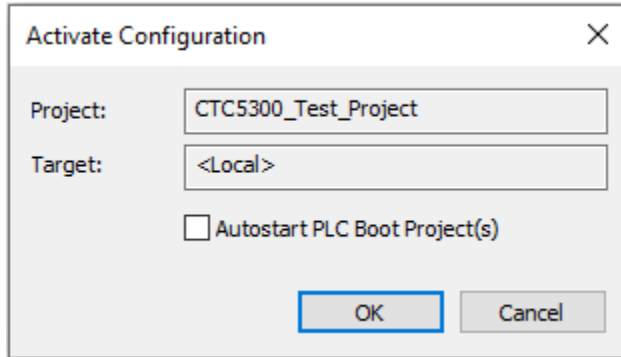
Select OK to Restart TwinCAT System in Run Mode:
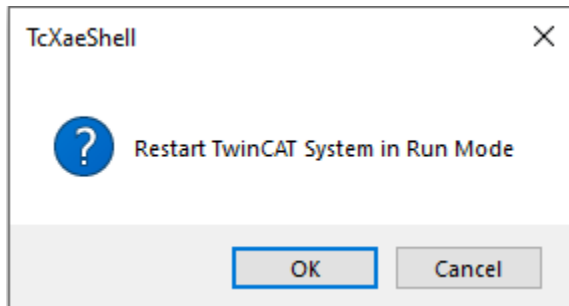
The program now has to be loaded into the real time core of TwinCAT to execute. Select the green Login toolbar icon:
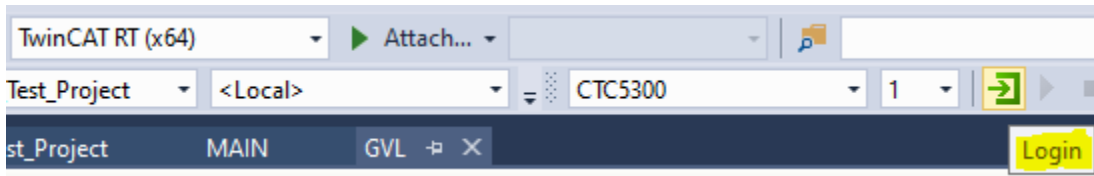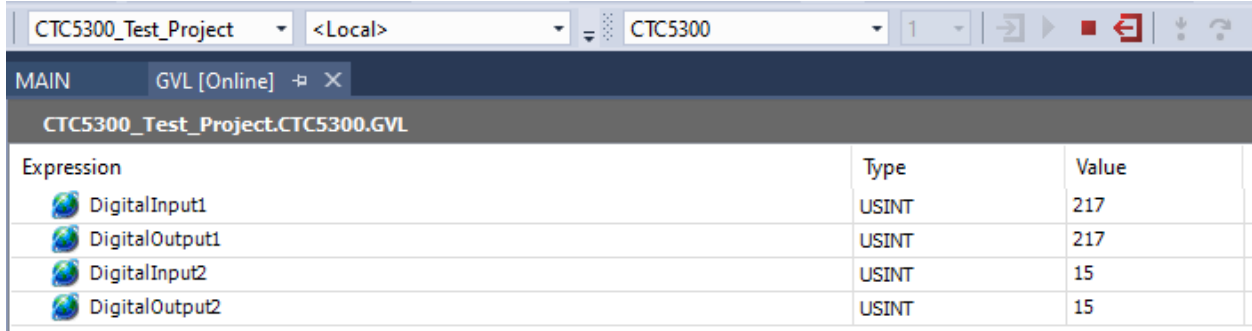
Select Yes to the TwinCAT PLC Control dialog to create 'Port_851':
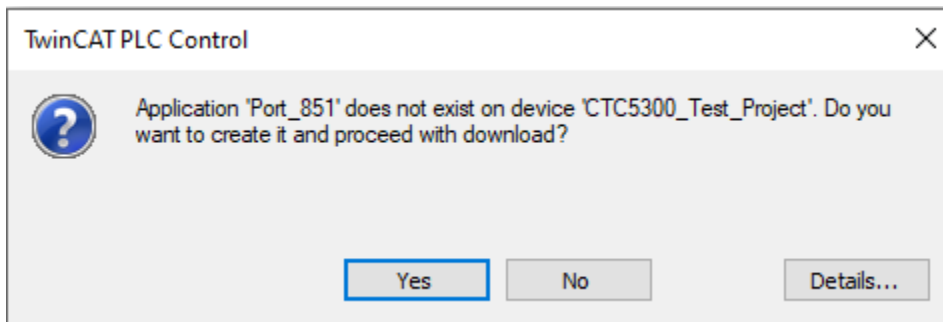
The program is now loaded and the green Start toolbar icon must be selected to start the program, cycling the IO ports:

The digital outputs will start incrementing and if a loopback connector is in place the digital inputs will increment as well. Click the red stop icon to stop the program and the red left facing arrow to log off interacting with the PLC. Reference TwinCAT documentation for additional details.

[5] Slave PDO Data Structure

5300 PDO Data Storage Structure

Standalone Mode

Data placement in the PDO is expected to follow the mapping logic of 0x1C12 and 0x1C13 where the modules represented by 0x1600 to 0x160F and 0x1A00 to 0x1A0F represent the IO in each slot of the controller. This is the standard for EtherCAT slave devices. Therefore, data mapping is based upon module placement in the PLC rack.

QuickBuilder Mode (CTC only, not TwinCAT)

TxPDO (read by Master)

Object 0x2101 defines what is available to determine the below mappings.

1. Digital inputs are first, bit oriented but breaking on full byte boundary if uneven where upper unused bits would not be used.
2. Digital outputs being read are next, bit oriented but breaking on full byte boundary if uneven where upper unused bits would not be used. This is currently not implemented.
3. Analog inputs, 4 bytes each, little endian
4. Analog outputs being read, 4 bytes each, little endian. Currently not implemented.
5. Integer registers, 4 bytes each.
6. Float32 registers, 4 bytes each.
7. Float64 (double), 8 bytes each.

RxPDO (written by Master)

Object 0x2102 defines what is available to determine the below mappings.

1. Digital outputs are first, bit oriented but breaking on full byte boundary if uneven where upper unused bits would not be used.
2. Analog outputs are next, 4 bytes each little endian
3. Integer registers, 4 bytes each.

4. Float32 registers, 4 bytes each.
5. Float64 (double), 8 bytes each.

[6] Slave Object Dictionary

5300 Slave Object Dictionary

Code	Data Type	Range
SINT	Signed 8-bit integer	-128 to 127
USINT	Unsigned 8-bit integer	0 to 255
INT	Signed 16-bit integer	-32,768 to 32,767
UINT	Unsigned 16-bit integer	0 to 65,535
DINT	Signed 32-bit integer	-2,147,483,648 to 2,147,483,647
UDINT	Unsigned 32-bit integer	0 to 4,294,967,295
REAL	32-bit float	-
LREAL	64-bit double	-
STRING	Character string	-

The 5300 slave supports SDOINFO where a Master can interrogate a slave device to discover what objects are available without a direct EtherCAT Esi xml file reference. In the following example the 5300 is configured as follows:

Slot 1 – M3-10A, 16 Digital Input / 16 Digital Output
Slot 2 – M3-10A, 16 Digital Input / 16 Digital Output
Slot 3 – M3-10A, 16 Digital Input / 16 Digital Output
Slot 4 – M3-10A, 16 Digital Input / 16 Digital Output
Slot 5 – M3-10B, 16 Digital Input / 16 Digital Output
Slot 6 – Empty
Slot 7 – M3-34B (4) 4-20ma Outputs and (4) 0-10V Outputs

Scanning the network and selecting a 5300 slave in standalone mode TwinCAT would display the following objects available:

Index	Name	Flags	Value
1000	Device Type	RO	0x00001389 (5001)
1008	Manufacturer Device Name	RO	CTC5300
1009	Manufacturer Hardware Version	RO	1.0
100A	Manufacturer Software Version	RO	R70.33
1018:0	Identity Object	RO	> 4 <
1600:0	Module 0 Entries	RO	> 16 <
1601:0	Module 1 Entries	RO	> 16 <
1602:0	Module 2 Entries	RO	> 16 <
1603:0	Module 3 Entries	RO	> 16 <
1604:0	Module 4 Entries	RO	> 16 <
1605:0	Module 5 Entries	RO	> 16 <
1606:0	Module 6 Entries	RO	> 16 <
1607:0	Module 7 Entries	RO	> 16 <
1608:0	Module 8 Entries	RO	> 16 <
1609:0	Module 9 Entries	RO	> 16 <
160A:0	Module 10 Entries	RO	> 16 <
160B:0	Module 11 Entries	RO	> 16 <
160C:0	Module 12 Entries	RO	> 16 <
160D:0	Module 13 Entries	RO	> 16 <
160E:0	Module 14 Entries	RO	> 16 <
160F:0	Module 15 Entries	RO	> 16 <
1A00:0	Module 0 Entries	RO	> 16 <
1A01:0	Module 1 Entries	RO	> 16 <
1A02:0	Module 2 Entries	RO	> 16 <
1A03:0	Module 3 Entries	RO	> 16 <
1A04:0	Module 4 Entries	RO	> 16 <
1A05:0	Module 5 Entries	RO	> 16 <
1A06:0	Module 6 Entries	RO	> 16 <
1A07:0	Module 7 Entries	RO	> 16 <
1A08:0	Module 8 Entries	RO	> 16 <
1A09:0	Module 9 Entries	RO	> 16 <
1A0A:0	Module 10 Entries	RO	> 16 <
1A0B:0	Module 11 Entries	RO	> 16 <
1A0C:0	Module 12 Entries	RO	> 16 <
1A0D:0	Module 13 Entries	RO	> 16 <
1A0E:0	Module 14 Entries	RO	> 16 <
1A0F:0	Module 15 Entries	RO	> 16 <
1C00:0	Sync Manager Communication Type	RO	> 4 <
1C12:0	Sync Manager 2 PDO Assignment	RO	> 6 <
1C13:0	Sync Manager 3 PDO Assignment	RO	> 5 <
1C32:0	SyncMgrParam	RO	> 6 <
2000:0	Register Offset Base	RO	> 5 <

Index	Name	Flags	Value
⊞ 2001:0	Integer Reg Bank 1	RO	> 16 <
⊞ 2002:0	Integer Reg Bank 2	RO	> 16 <
⊞ 2003:0	Integer Reg Bank 3	RO	> 16 <
⊞ 2004:0	Double Reg Bank 4	RO	> 16 <
⊞ 2005:0	Double Reg Bank 5	RO	> 16 <
⊞ 2100:0	PDO Size	RO	> 4 <
⊞ 2101:0	Mapping Read Information	RO	> 9 <
⊞ 2102:0	Mapping Write Information	RO	> 9 <
⊞ 2103:0	Mapping Write Information	RO	> 4 <
⊞ 2104:0	Register Read Assignment	RO	> 9 <
⊞ 2105:0	Register Write Assignment	RO	> 9 <
⊞ 3000:0	Digital Input Bits	RO	> 80 <
⊞ 3001:0	Digital Input Bits	RO	> 0 <
⊞ 3002:0	Digital Input Bits	RO	> 0 <
⊞ 3003:0	Digital Input Bits	RO	> 0 <
⊞ 3004:0	Digital Input Bytes	RO	> 10 <
⊞ 3005:0	Digital Input Words	RO	> 5 <
⊞ 3006:0	Digital Input DWords	RO	> 3 <
⊞ 3100:0	Digital Output Bits	RO	> 80 <
⊞ 3101:0	Digital Output Bits	RO	> 0 <
⊞ 3102:0	Digital Output Bits	RO	> 0 <
⊞ 3103:0	Digital Output Bits	RO	> 0 <
⊞ 3104:0	Digital Output Bytes	RO	> 10 <
⊞ 3105:0	Digital Output Words	RO	> 5 <
⊞ 3106:0	Digital Output DWords	RO	> 3 <
⊞ 6000:0	M3-10A +VS 16DI/16DO NPN	RO	> 2 <
⊞ 6010:0	M3-10A +VS 16DI/16DO NPN	RO	> 2 <
⊞ 6020:0	M3-10A +VS 16DI/16DO NPN	RO	> 2 <
⊞ 6030:0	M3-10A +VS 16DI/16DO NPN	RO	> 2 <
⊞ 6040:0	M3-10B +5V 16DI/16DO NPN	RO	> 2 <
⊞ 6050:0	NONE	RO	> 0 <
⊞ 6060:0	NONE	RO	> 0 <
⊞ 6070:0	NONE	RO	> 0 <
⊞ 6080:0	NONE	RO	> 0 <
⊞ 6090:0	NONE	RO	> 0 <
⊞ 60A0:0	NONE	RO	> 0 <
⊞ 60B0:0	NONE	RO	> 0 <
⊞ 60C0:0	NONE	RO	> 0 <
⊞ 60D0:0	NONE	RO	> 0 <
⊞ 60E0:0	NONE	RO	> 0 <
⊞ 60F0:0	NONE	RO	> 0 <

7000:0	M3-10A +VS 16DI/16DO NPN	RO	> 2 <			
7010:0	M3-10A +VS 16DI/16DO NPN	RO	> 2 <			
7020:0	M3-10A +VS 16DI/16DO NPN	RO	> 2 <			
7030:0	M3-10A +VS 16DI/16DO NPN	RO	> 2 <			
7040:0	M3-10B +5V 16DI/16DO NPN	RO	> 2 <			
7050:0	NONE	RO	> 0 <			
7060:0	M3-34B 4 4-20ma 4 0-10V	RO	> 8 <			
7070:0	NONE	RO	> 0 <			
7080:0	NONE	RO	> 0 <			
7090:0	NONE	RO	> 0 <			
70A0:0	NONE	RO	> 0 <			
70B0:0	NONE	RO	> 0 <			
70C0:0	NONE	RO	> 0 <			
70D0:0	NONE	RO	> 0 <			
70E0:0	NONE	RO	> 0 <			
70F0:0	NONE	RO	> 0 <			
F000:0	Module Device Profile	RO	> 5 <			
F040:0	Detected Address List	RO	> 16 <			
F050:0	Detected Module List	RO	> 16 <			

Object definitions:

Index	Subindex	Object Type	Description	Data Type	Access	PDO	Initial Value
0x1000	0x00	VAR	Device Type	UDINT	RO	No	5001
0x1008	0x00	VAR	Manufacturer Device Name	STRING	RO	No	CTC5300
0x1009	0x00	VAR	Manufacturer Hardware Version	STRING	RO	No	1.0
0x100A	0x00	VAR	Manufacturer Software Version	STRING	RO	No	R70.33
0x1018	0x00	RECORD	Number of Entries	USINT	RO	No	0x04
	0x01	-	Vendor ID	UDINT	RO	No	0x00000989
	0x02	-	Product Code	UDINT	RO	No	0x00000002
	0x03	-	Revision Number	UDINT	RO	No	0x00007033
	0x04	-	Serial Number	UDINT	RO	No	0x00000000
0x1600	0x00	RECORD	Module 0 - Number of Entries in RxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1601	0x00	RECORD	Module 1 - Number of Entries in RxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1602	0x00	RECORD	Module 2 - Number of Entries in RxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000

0x1603	0x00	RECORD	Module 3 - Number of Entries in RxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1604	0x00	RECORD	Module 4 - Number of Entries in RxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1605	0x00	RECORD	Module 5- Number of Entries in RxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1606	0x00	RECORD	Module 6 - Number of Entries in RxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1607	0x00	RECORD	Module 7 - Number of Entries in RxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1608	0x00	RECORD	Module 8 - Number of Entries in RxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1609	0x00	RECORD	Module 9 - Number of Entries in RxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x160A	0x00	RECORD	Module 10 - Number of Entries in RxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x160B	0x00	RECORD	Module 11- Number of Entries in RxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x160C	0x00	RECORD	Module 12 - Number of Entries in RxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x160D	0x00	RECORD	Module 13 - Number of Entries in RxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x160E	0x00	RECORD	Module 14 - Number of Entries in RxPDO	USINT	RO	No	0x00

	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x160F	0x00	RECORD	Module 15 - Number of Entries in RxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1A00	0x00	RECORD	Module 0 - Number of Entries in TxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1A01	0x00	RECORD	Module 1 - Number of Entries in TxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1A02	0x00	RECORD	Module 2 - Number of Entries in TxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1A03	0x00	RECORD	Module 3 - Number of Entries in TxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1A04	0x00	RECORD	Module 4 - Number of Entries in TxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1A05	0x00	RECORD	Module 5- Number of Entries in TxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1A06	0x00	RECORD	Module 6 - Number of Entries in TxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1A07	0x00	RECORD	Module 7 - Number of Entries in TxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1A08	0x00	RECORD	Module 8 - Number of Entries in TxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1A09	0x00	RECORD	Module 9 - Number of Entries in TxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000

0x1A0A	0x00	RECORD	Module 10 - Number of Entries in TxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1A0B	0x00	RECORD	Module 11- Number of Entries in TxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1A0C	0x00	RECORD	Module 12 - Number of Entries in TxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1A0D	0x00	RECORD	Module 13 - Number of Entries in TxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1A0E	0x00	RECORD	Module 14 - Number of Entries in TxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1A0F	0x00	RECORD	Module 15 - Number of Entries in TxPDO	USINT	RO	No	0x00
	0x01 to 0x10		Objects mapped in the PLC slot	UDINT	RO	No	0x00000000
0x1C00	0x00	ARRAY	SM (Sync Manager) Communication Type Number of Entries	USINT	RO	No	0x04
	0x01 0x02 0x03 0x04		SM0 type: Mailbox Rx SM1 type: Mailbox Tx SM2 type: TxPDO SM3 type: RxPDO	UDINT	RO	No	0x00000001 0x00000002 0x00000003 0x00000004
0x1C12	0x00	ARRAY	RxPDO assign number of 0x16xx entries	USINT	RO	No	0x00
	0x01 to n			UINT	RO	No	0x0000
0x1C13	0x00	ARRAY	TxPDO assign number of 0x1Axx entries	USINT	RO	No	0x00
	0x01 to n			UINT	RO	No	0x0000
0x1C32	0x00	RECORD	Sync Mgr Output Number of Parameters	USINT	RO	No	0x06
	0x01		Sync mode	UINT	RO	No	0x0000
	0x02		Cycle Time	UDINT	RO	No	0x00000000
	0x03		Shift Time	UDINT	RO	No	0x00000000

	0x04		Sync modes supported	UINT	RO	No	0x0000
	0x05		Minimum Cycle Time	UDINT	RO	No	0x0007A120
	0x06		Calc and Copy Time	UDINT	RO	No	0x0000000

0x2000: Register Offset

This Object sets the value that is added to the Subindex for access to the 5300 PLC registers for objects 0x2001 to 0x2005. Subindex 1 to 3 are for Integer access and Subindex 4 and 5 are for double type access. This is necessary since the Subindex range is only 0x1 to 0xff and there are over 36000 registers. As an example, setting 0x2001 Subindex 1 to a 0 would result in any read/write to 0x2001 Subindex 1 to access register 1 in the controller; 0x2001 Subindex + 0x2000 Subindex 1 offset = 1 + 0 = register 1. 0x2001 Subindex 2 would result in access register 2.

Subindex		Description	Data Type	Access	PDO	Initial Value
0x00		Number of Entries	USINT	RO	No	5
0x01		Offset for register access using object 0x2001. This value is added to the Subindex used to access 0x2001 in order to determine the register number to access.	DINT	RW	No	0
0x02		Offset for register access using object 0x2002. This value is added to the Subindex used to access 0x2002 in order to determine the register number to access.	DINT	RW	No	0
0x03		Offset for register access using object 0x2003. This value is added to the Subindex used to access 0x2003 in order to determine the register number to access.	DINT	RW	No	0
0x04		Offset for register access using object 0x2004. This value is added to	DINT	RW	No	0

			Data Type	Access	PDO	Initial Value
		the Subindex used to access 0x2004 in order to determine the register number to access.				
0x05		Offset for register access using object 0x2005. This value is added to the Subindex used to access 0x2005 in order to determine the register number to access.	DINT	RW	No	0

0x2001: Integer Register Bank 1

This Objects' Subindex is added to object 0x2000 subindex 1 value to determine the register within the 5300 PLC to access. Contents are treated as an integer, DINT.

Subindex		Description	Data Type	Access	PDO	Initial Value
0x00		Number of Entries	USINT	RO	No	16
0x01 – 0x10		Subindex + 0x2000 subindex 1 value determines register to access	DINT	RW	No	0

0x2002: Integer Register Bank 2

This Objects' Subindex is added to object 0x2000 subindex 2 value to determine the register within the 5300 PLC to access. Contents are treated as an integer, DINT.

Subindex		Description	Data Type	Access	PDO	Initial Value
0x00		Number of Entries	USINT	RO	No	16
0x01 – 0x10		Subindex + 0x2000 subindex 2 value determines register to access	DINT	RW	No	0

0x2003: Integer Register Bank 3

This Objects' Subindex is added to object 0x2000 subindex 3 value to determine the register within the 5300 PLC to access. Contents are treated as an integer, DINT.

Subindex		Description	Data Type	Access	PDO	Initial Value
0x00		Number of Entries	USINT	RO	No	16
0x01 – 0x10		Subindex + 0x2000 subindex 3 value determines register to access	DINT	RW	No	0

0x2004: Double Register Bank 4

This Objects' Subindex is added to object 0x2000 subindex 4 value to determine the register within the 5300 PLC to access. Contents are treated as a double, LREAL.

Subindex		Description	Data Type	Access	PDO	Initial Value
0x00		Number of Entries	USINT	RO	No	16
0x01 – 0x10		Subindex + 0x2000 subindex 4 value determines register to access	LREAL	RW	No	0

0x2004: Double Register Bank 5

This Objects' Subindex is added to object 0x2000 subindex 5 value to determine the register within the 5300 PLC to access. Contents are treated as a double, LREAL.

Subindex		Description	Data Type	Access	PDO	Initial Value
0x00		Number of Entries	USINT	RO	No	16
0x01 – 0x10		Subindex + 0x2000 subindex 5 value determines register to access	LREAL	RW	No	0

0x2100: PDO Size

This Object contains the maximum PDO size allowed and that which is actually being used.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of Entries	USINT	RO	No	4
0x01	RxPDO Required Size – This size will reflect the size of the RxPDO with all data mapped and should be what the EtherCAT Master sets the Sync Manager to. This is the size to be used during Safe_OP and OP cycles.	DINT	RO	No	0

0x02	RxPDO Maximum Size	DINT	RO	No	1024
0x03	TxPDO Requried Size - This size will reflect the size of the TxPDO with all data mapped and should be what the EtherCAT Master sets the Sync Manager to. This is the size to be used during Safe_OP and OP cycles.	DINT	RO	No	0
0x04	TxPDO Maximum Size	DINT	RO	No	1024

0x2101: TxPDO (read by Master, sent by the Slave) data type available and quantity

This Object contains a bit mask of the type of data available as well as the quantity of that data type that can be expected in the TxPDO. Little Endian representation within the PDO.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of Entries	USINT	RO	No	9
0x01	TxPDO data available mask: Bit 7 - Digital Input Bit 6 - Digital Output Bit 5 - Analog Input Bit 4 - Analog Output Bit 3 - Regs Int Bit 2 - Regs Float Bit 1 - Regs Double Bit 0 - (TBD)	USINT	RO	No	0
0x02	Reserved	INT	RO	No	0
0x03	Number of Doubles (64 bit)	INT	RO	No	0
0x04	Number of Floats (32 bit)	INT	RO	No	0
0x05	Number of Integers (32 bit)	INT	RO	No	0
0x06	Number of Analog Outputs	INT	RO	No	0
0x07	Number of Analog Inputs	INT	RO	No	0
0x08	Number of Digital Outputs	INT	RO	No	0
0x09	Number of Digital Inputs	INT	RO	No	0

0x2102: RxPDO (sent by the Master, received by the Slave) data type available and quantity

This Object contains a bit mask of the type of data available as well as the quantity of that data type that can be expected in the RxPDO. Little Endian representation within the PDO.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of Entries	USINT	RO	No	9
0x01	RxPDO data available mask: Bit 7 - Digital Output Bit 6 - Analog Output Bit 5 – Register Integer Bit 4 – Register Float (32 bit)	USINT	RO	No	0

	Bit 3 – Register Double (64 bit) Bit 2 - Reserved Bit 1 - Reserved Bit 0 - Reserved				
0x02	Reserved	INT	RO	No	0
0x03	Reserved	INT	RO	No	0
0x04	Reserved	INT	RO	No	0
0x05	Number of Doubles (64 bits)	INT	RO	No	0
0x06	Number of Floats (32 bits)	INT	RO	No	0
0x07	Number of Integers (32 bits)	INT	RO	No	0
0x08	Number of Analog Outputs	INT	RO	No	0
0x09	Number of Digital Outputs	INT	RO	No	0

0x2103: Operating Mode

The 5300 Controller can operate in two modes. The first as a standalone dedicated controller where all IO is controlled by, and exposed to, the EtherCAT Master. The second where a QuickBuilder program can map individual IO as public along with registers and variants but restrict what the EtherCAT Master can see and control.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of Entries	USINT	RO	No	4
0x01	Operating Mode, 1 all IO is public and controlled by Master, 0 QuickBuilder (QB) Mode where IO and variables are enabled public individually by a QB program.	USINT	RO	No	Set by eeprom
0x02	QuickBuilder Mapping Mode (Register 13065) . Normally this register is set to a default value of 3 where the Slave interacts with the Master normally. If set to a 0 the Slave will respond to EtherCAT requests but Master PDO data will remain unchanged and not be updated or processed. Set to 1 for the Master PDO received data to be updated but the writes to the Slave to be ignored. Set to a 2 for the Slave to process write requests but not update any PDO read data.	USINT	RO	No	0
0x03	QuickBuilder Mapping Complete. If set to a 1 then it is OK for the EtherCAT Master to access the system as QuickBuilder has setup mapping of all IO and variable access	USINT	RO	No	0

	to PDOs and the user application program is running.				
0x04	Reserved	USINT	RO	No	

0x2104: TxPDO, assigned public registers numbers (Quickbuilder Mode Only)

This Object contains the assigned register number for each public variable type and the offset in the table to read register number from. This is for QuickBuilder mode only and reflects those registers that the application program makes public and gives a way for the Master to read the assigned register numbers from the table. The index should be set to 0 prior to reading the desired table entry and it will automatically increment with each read. This is for registers read by the Master and contained within the PDO.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of entries	USINT	RO	No	9
0x01	Number of integers (32 bits)	INT	RO	No	0
0x02	Integer index (base 0)	INT	RW	No	0
0x03	Integer register number for index set	INT	RO	No	0
0x04	Number of floats (32 bits)	INT	RO	No	0
0x05	Float index (base 0)	INT	RW	No	0
0x06	Float register number for index set	INT	RO	No	0
0x07	Number of doubles (64 bits)	INT	RO	No	0
0x08	Double index (base 0)	INT	RW	No	0
0x09	Double register number for index set	INT	RO	No	0

0x2105: RxPDO, assigned public registers numbers (Quickbuilder Mode Only)

This Object contains the assigned register number for each public variable type and the offset in the table to read register number from. This is for QuickBuilder mode only and reflects those registers that the application program makes public and gives a way for the Master to read the assigned register numbers from the table. The index should be set to 0 prior to reading the desired table entry and it will automatically increment with each read. This is for registers written by the Master and contained within the PDO.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of entries	USINT	RO	No	9
0x01	Number of integers (32 bits)	INT	RO	No	0
0x02	Integer index (base 0)	INT	RW	No	0
0x03	Integer register number for index set	INT	RO	No	0
0x04	Number of floats (32 bits)	INT	RO	No	0
0x05	Float index (base 0)	INT	RW	No	0
0x06	Float register number for index set	INT	RO	No	0
0x07	Number of doubles (64 bits)	INT	RO	No	0
0x08	Double index (base 0)	INT	RW	No	0
0x09	Double register number for index set	INT	RO	No	0

0x3000: Digital Input Bits 1 to 255 State

This Object causes the referenced installed digital input bit to be returned where the first bit references the 5300 PLC register #2001. The input must actually exist to contain an entry.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of entries	USINT	RO	No	0
0x01 to n	Digital Input Bit	BOOLEAN	RO	No	0 or 1

0x3001: Digital Input Bits 256 to 511 State

This Object causes the referenced installed digital input bit to be returned where the first bit references the 5300 PLC register #2256. The input must actually exist to contain an entry.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of entries	USINT	RO	No	0
0x01 to n	Digital Input Bit	BOOLEAN	RO	No	0 or 1

0x3002: Digital Input Bits 512 to 767 State

This Object causes the referenced installed digital input bit to be returned where the first bit references the 5300 PLC register #2512. The input must actually exist to contain an entry.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of entries	USINT	RO	No	0
0x01 to n	Digital Input Bit	BOOLEAN	RO	No	0 or 1

0x3003: Digital Input Bits 768 to 1023 State

This Object causes the referenced installed digital input bit to be returned where the first bit references the 5300 PLC register #2768. The input must actually exist to contain an entry.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of entries	USINT	RO	No	0
0x01 to n	Digital Input Bit	BOOLEAN	RO	No	0 or 1

0x3004: Digital Input Bytes

This Object causes the referenced installed digital input byte to be returned where the first byte references the 5300 PLC register #11201. The input must actually exist to contain an entry.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of entries	USINT	RO	No	0
0x01 to n	Digital Input Byte	USINT	RO	No	0 to 0xFF

0x3005: Digital Input Words

This Object causes the referenced installed digital input 16-bit word to be returned where the first word references the 5300 PLC register #11101. The input must actually exist to contain an entry.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of entries	USINT	RO	No	0
0x01 to n	Digital Input Word	UINT	RO	No	0 to 0xFFFF

0x3006: Digital Input DWords

This Object causes the referenced installed digital input 32-bit word (DWord) to be returned where the first DWord references the 5300 PLC register #11001. The input must actually exist to contain an entry.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of entries	USINT	RO	No	0
0x01 to n	Digital Input DWord	UDINT	RO	No	0 to 0xFFFFFFFF

0x3100: Digital Output Bits 1 to 255 State

This Object causes the referenced installed digital output bit to be returned where the first bit references the 5300 PLC register #1001. The output must actually exist to contain an entry.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of entries	USINT	RO	No	0
0x01 to n	Digital Output Bit	BOOLEAN	RW	No	0 or 1

0x3101: Digital Output Bits 256 to 511 State

This Object causes the referenced installed digital output bit to be returned where the first bit references the 5300 PLC register #1256. The output must actually exist to contain an entry.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of entries	USINT	RO	No	0
0x01 to n	Digital Output Bit	BOOLEAN	RW	No	0 or 1

0x3102: Digital Output Bits 512 to 767 State

This Object causes the referenced installed digital output bit to be returned where the first bit references the 5300 PLC register #1512. The output must actually exist to contain an entry.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of entries	USINT	RO	No	0
0x01 to n	Digital Output Bit	BOOLEAN	RW	No	0 or 1

0x3103: Digital Output Bits 768 to 1023 State

This Object causes the referenced installed digital output bit to be returned where the first bit references the 5300 PLC register #1768. The output must actually exist to contain an entry.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of entries	USINT	RO	No	0
0x01 to n	Digital Output Bit	BOOLEAN	RW	No	0 or 1

0x3104: Digital Output Bytes

This Object causes the referenced installed digital output byte to be returned where the first byte references the 5300 PLC register #10201. The output must actually exist to contain an entry.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of entries	USINT	RO	No	0
0x01 to n	Digital Output Byte	USINT	RW	No	0 to 0xFF

0x3105: Digital Output Words

This Object causes the referenced installed digital output 16-bit word to be returned where the first word references the 5300 PLC register #10101. The output must actually exist to contain an entry.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of entries	USINT	RO	No	0
0x01 to n	Digital Output Word	UINT	RO	No	0 to 0xFFFF

0x3106: Digital Output DWords

This Object causes the referenced installed digital output 32-bit word (DWord) to be returned where the first DWord references the 5300 PLC register #10001. The output must actually exist to contain an entry.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of entries	USINT	RO	No	0
0x01 to n	Digital Output DWord	UDINT	RW	No	0 to 0xFFFFFFFF

0x6000 to 0x60F0: Input area, process data input objects (mapped to TxPDOs)

The 5300 PLC uses what is called slot based addressing where each 16 of the Input area represents the data area of the module inserted into that rack slot of the controller. Thus, 0x6000 to 0x600F is the entry for each input section of the module inserted into the first controller slot, 0x6010 to 0x601F the second, etc. As an example, a M3-10A in the first slot would have two entries, each of 8 bits for its 16 bits of total input data, using two entries at 0x6000.1 and 0x6000.2.

```
⊟  6000:0      M3-10A +VS 16DI/16DO NPN         RO        > 2 <
   6000:01   Digital Inputs                    RO        0x00 (0)
   6000:02   Digital Inputs                    RO        0x00 (0)
```

This will return the actual data as it is read by the controller.

An Analog card input would have 16-bit entries, one for each channel. With EtherCAT the logical mapping of how objects interact are as follows:

0x1C13 <- 0x1Axx <- 0x60xx

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of input entries	USINT	RO	No	0
0x01 to 0x0F	5300 Module data name, such as Digital Inputs or Analog Input	Type based on module in rack	RO	Yes	0

0x7000 to 0x70F0: Output area, process data output objects (mapped to RxPDOs)

The 5300 PLC uses what is called slot based addressing where each 16 of the Output area represents the data area of the module inserted into that rack slot of the controller. Thus, 0x7000 to 0x700F is the entry for each output section of the module inserted into the first controller slot, 0x7010 to 0x701F the second, etc. As an example, a M3-10A in the first slot would have two entries, each of 8 bits for its 16 bits of total output data, using two entries at 0x7000.1 and 0x7000.2.

```
⊟  7000:0      M3-10A +VS 16DI/16DO NPN         RO        > 2 <
   7000:01   Digital Outputs                   RW        0x00 (0)
   7000:02   Digital Outputs                   RW        0x00 (0)
```

Or an Analog M3-34B Module in slot 1 of the controller:

```
⊟  7060:0      M3-34B 4 4-20ma 4 0-10V          RO        > 8 <
   7060:01   Analog Output                     RW        0
   7060:02   Analog Output                     RW        0
   7060:03   Analog Output                     RW        0
   7060:04   Analog Output                     RW        0
   7060:05   Analog Output                     RW        0
   7060:06   Analog Output                     RW        0
   7060:07   Analog Output                     RW        0
   7060:08   Analog Output                     RW        0
```

This will return the actual data as it is read by the controller.

An Analog card input would have 16-bit entries, one for each channel. With EtherCAT the logical mapping of how objects interact are as follows:

0x1C13 <- 0x1Axx <- 0x60xx

Subindex	Description	Data Type	Access	PDO	Initial Value

0x00	Number of input entries	USINT	RO	No	0
0x01 to 0x0F	5300 Module data name, such as Digital Outputs or Analog Output	Type based on module in rack	RW	Yes	0

0xF000: Module Device Profile.

This Object provides information on what modules are installed in the 5300 backplane.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of entries	USINT	RO	No	5
0x01	Index distance	UINT	RO	No	0x0010
0x02	Maximum number of modules	UINT	RO	No	0x0010
0x03	Standard entries in Object 0x8yy0	UDINT	RO	No	0
0x04	Standard entries in Object 0x9yy0	UDINT	RO	No	0
0x05	Module PDO group of the device	UDINT	RO	No	0

0xF040: Device Address List.

This Object provides information as to where a module is plugged into the 5300 backplane. The slot number will be entered at the appropriate index, where index 1 is slot 1, index 8 is slot 8. A module plugged into backplane slot 3 would have a 3 in index 3. If a module is not available the index will contain a 0.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Number of entries	USINT	RO	No	16
0x01 to 0x10	SubIndex 001 to 016	UINT	RO	No	0x0000

0xF050: Available 5300 backplane IO modules and slots where installed.

This Object provides information on what modules are installed in the 5300 backplane.

Subindex	Description	Data Type	Access	PDO	Initial Value
0x00	Maximum number of module backplanes slots	USINT	RO	No	16
0x01	Slot 1 Module Code ID	UDINT	RO	No	0
0x02	Slot 2 Module Code ID	UDINT	RO	No	0
0x03	Slot 3 Module Code ID	UDINT	RO	No	0
0x04	Slot 4 Module Code ID	UDINT	RO	No	0
0x05	Slot 5 Module Code ID	UDINT	RO	No	0
0x06	Slot 6 Module Code ID	UDINT	RO	No	0
0x07	Slot 7 Module Code ID	UDINT	RO	No	0
0x08	Slot 8 Module Code ID	UDINT	RO	No	0
0x09	Slot 9 Module Code ID	UDINT	RO	No	0

0x0A	Slot 10 Module Code ID	UDINT	RO	No	0
0x0B	Slot 11 Module Code ID	UDINT	RO	No	0
0x0C	Slot 12 Module Code ID	UDINT	RO	No	0
0x0D	Slot 13 Module Code ID	UDINT	RO	No	0
0x0E	Slot 14 Module Code ID	UDINT	RO	No	0
0x0F	Slot 15 Module Code ID	UDINT	RO	No	0
0x10	Slot 16 Module Code ID	UDINT	RO	No	0

Module DINT codes can be translated as follows:

```
#define MODULE_M3_10A (~0x01 & 0x00ff)      // 16DI VDC SNK / 16DO VDC SNK
#define MODULE_M3_10B (~0x02 & 0x00ff)      // 16DI 5VDC SNK / 16DO VDC SNK
#define MODULE_M3_10E (~0x0F & 0x00ff)      // 16DI VDC SRC / 16DO VDC SRC - PROTOTYPE

#define MODULE_M3_11A (~0x09 & 0x00ff)      // 16DI VDC SRC / 16DO VDC SRC
#define MODULE_M3_11B (~0x0a & 0x00ff)      // 16DI 5VDC SRC / 16DO 5VDC SRC

#define MODULE_M3_12A (~0x11 & 0x00ff)      // 16 RELAY FORM-A (NO SPST)
#define MODULE_M3_12B (~0x12 & 0x00ff)      // 8 RELAY FORM-A (NO SPST)

#define MODULE_M3_13A (~0x13 & 0x00ff)      // 4 RELAY FORM-C (NO/NC SPDT)

#define MODULE_M3_14A (~0x1d & 0x00ff)      // 32DO VDC SNK
#define MODULE_M3_14B (~0x1e & 0x00ff)      // 16DO VDC SNK

#define MODULE_M3_15A (~0x1f & 0x00ff)      // 32DO VDC SRC
#define MODULE_M3_15B (~0x20 & 0x00ff)      // 16DO 5VDC SRC
#define MODULE_M3_15C (~0x21 & 0x00ff)      // 32DO 5VDC SRC

#define MODULE_M3_16A (~0x15 & 0x00ff)      // 32DI VDC SNK
#define MODULE_M3_16B (~0x16 & 0x00ff)      // 16DI VDC SNK
#define MODULE_M3_16C (~0x17 & 0x00ff)      // 32DI 5VDC SNK
#define MODULE_M3_16D (~0x18 & 0x00ff)      // 32DI 5VDC SRC
#define MODULE_M3_16E (~0x19 & 0x00ff)      // 32DI VDC SRC
#define MODULE_M3_16F (~0x1a & 0x00ff)      // 16DI VDC SRC

#define MODULE_M3_18A (~0x25 & 0x00ff)      // 8DI SPDT SWITCH / 8DO LED

#define MODULE_M3_DUALPORT_41A (~0xC0 & 0x00ff) // EtherCAT Module

 // Dualport boards
#define MODULE_M3_DUALPORT        0x7F
#define MODULE_M3_DUAL_20         0x5320
// M3-20 PCA codes
#define MODULE_M3_20A 0      // 16 +VS VDC smart sinking in, 16 +VS VDC sinking out, 8CNT 8PWM
#define MODULE_M3_20B 1             // VDC SNK Dual Stepper 16DI, 16DO
```

```
#define MODULE_M3_20C 2              // 16 +5 VDC smart sinking inputs, 16 +5 VDC sinking outputs,
8CNT 8PWM
#define MODULE_M3_20D 3             // 5VDC SNK Dual Stepper 16DI, 16DO
#define MODULE_M3_20E 0x04           // M3-20E - VDC SNK 16/16 DIO 8 PWM
#define MODULE_M3_20F 0x05          // M3-20F - 5VDC SNK 16/16 DIO 8 PWM
#define MODULE_M3_20G 0x06           // M3-20G - VDC SNK 16DI 16DO 16CNT 16PWM, same as 20A
#define MODULE_M3_20H 0x07           // M3-20H - 5VDC SNK 16DI 16DO 16CNT 16PWM, same as 20C

#define MODULE_M3_DUAL_21 0x5321
// M3-21 PCA codes
#define MODULE_M3_21A 0        // 16 +VS VDC smart sourcing inputs, 16 +VS VDC sourcing outputs
#define MODULE_M3_21B 1              // VDC Sourcing Dual Stepper 16DI, 16DO
#define MODULE_M3_21C 2               // 16 +5 VDC smart sourcing inputs, 16 +5 VDC sourcing outputs
#define MODULE_M3_21D 3             // 5VDC Sourcing Dual Stepper 16DI, 16DO
#define MODULE_M3_21E 0x04          // M3-21E - VDC SNK 16/16 DIO 8 PWM
#define MODULE_M3_21F 0x05          // M3-21F - 5VDC SNK 16/16 DIO 8 PWM
#define MODULE_M3_21G 0x06           // M3-20G - VDC SNK 16DI 16DO 16CNT 16PWM, same as 21A
#define MODULE_M3_21H 0x07           // M3-20H - 5VDC SNK 16DI 16DO 16CNT 16PWM, same as 21C

#define MODULE_M3_DUAL_30 0x5330
// M3-30 PCA codes
#define MODULE_M3_30A 0                   // 16AI +-10V / 8AO +-10V
#define MODULE_M3_30B 1                   // 8AI +-10V / 8AO +-10V
#define MODULE_M3_30C 2                   // 8AI +-20mV / 8AO +-10V
#define MODULE_M3_30D 3                   // 8AI +-100mV / 8AO +-10V
#define MODULE_M3_30E 4                   // 16AI +-20mA / 8AO +-10V

#define MODULE_M3_DUAL_31 0x5331
// M3-31 PCA codes
#define MODULE_M3_31A 1                   // 32AI +-10V
#define MODULE_M3_31B 2                   // 8AI +-10V
#define MODULE_M3_31C 3                   // 16AI +-20mV
#define MODULE_M3_31D 4                   // 16AI +-100mV/TC
//#define MODULE_M3_31E 5                 // 32AI +-20mA

#define MODULE_M3_DUAL_32 0x5332
// M3-32 PCA codes
#define MODULE_M3_32A 1                   // 32AO +-10V
#define MODULE_M3_32B 2                   // 16AO +-10V

#define MODULE_M3_DUAL_33 0x5333
// M3-33 PCA codes
#define MODULE_M3_33A 1                   // 8 AI +/- 20mv
#define MODULE_M3_33B 2                   // 4 AI +/- 20mv
#define MODULE_M3_33C 3                    // 8 channel Thermocouple
#define MODULE_M3_33D 4                   // 4 channel Thermocouple
#define MODULE_M3_33E 5                   // 8 AI +/- 10V
```

```
#define MODULE_M3_DUAL_34 0x5334
// M3-34 PCA codes
#define MODULE_M3_34A 1                    // 8 AO +/- 20ma
#define MODULE_M3_34B 2                    // 4 AO +/- 20ma  4 AO 10V

#define MODULE_M3_DUAL_35 0x5335
// M3-35 PCA codes
#define MODULE_M3_35A 0                    // 8AI +-20mV
#define MODULE_M3_35B 1                    // 8AI +-100mV/TC

#define MODULE_M3_DUAL_39 0x5339

#define MODULE_M3_DUAL_40 0x5340           // C - PLS, E - High speed counter
#define MODULE_M3_40A 0                    // Dual Axis Servo
#define MODULE_M3_40B 1                    // Dual Axis Stepper
#define MODULE_M3_40C 2                    // Dead
#define MODULE_M3_40D 3                    // 1.5 Axis Servo
#define MODULE_M3_40E 4                    // Quad Counter (Differential Ended)
#define MODULE_M3_40F 5                    // Quad Counter (Single Ended)
#define MODULE_M3_40G 6                    // 3 axis stepper

#define MODULE_M3_DUAL_41 0x5341           // EtherCAT Interface
#define MODULE_M3_DUAL_42 0x5342           // EtherCAT Interface
#define MODULE_M3_41A 10                   // EtherCAT only
#define MODULE_M3_41B 11                   // EtherCAT with 2 local servo's

#define MODULE_M3_DUAL_60 0x5360           // DeviceNet

#define MODULE_M3_DUAL_61 0x5361           // DeviceNet
#define MODULE_M3_61A 0                    // DeviceNet Master
#define MODULE_M3_61B 1                    // DeviceNet Slave
#define MODULE_M3_61C 2                    // Ethernet/IP Master
#define MODULE_M3_61D 3                    // Ethernet/IP Slave
#define MODULE_M3_61E 4                    // Profibus Master
#define MODULE_M3_61F 5                    // Profibus Slave
#define MODULE_M3_61G 6                    // CANOpen Slave

#define MODULE_M3_10A (~0x01 & 0x00ff) // 16DI VDC SNK / 16DO VDC SNK
#define MODULE_M3_10B (~0x02 & 0x00ff)     // 16DI 5VDC SNK / 16DO VDC SNK
#define MODULE_M3_10E (~0x0F & 0x00ff)     // 16DI VDC SRC / 16DO VDC SRC - PROTOTYPE

#define MODULE_M3_11A (~0x09 & 0x00ff)     // 16DI VDC SRC / 16DO VDC SRC
#define MODULE_M3_11B (~0x0a & 0x00ff)     // 16DI 5VDC SRC / 16DO 5VDC SRC
```

#define MODULE_M3_12A (~0x11 & 0x00ff)   // 16 RELAY FORM-A (NO SPST)
#define MODULE_M3_12B (~0x12 & 0x00ff)   // 8 RELAY FORM-A (NO SPST)

Entry codes will appear as below for each module in F050.1 to F050.16.

Module Description	Code ID
Empty Slot	0x00000001
M3-10A +VS 16DI/16DO NPN	(MODULE_M3_10A << 16)
M3-10B +5V 16DI/16DO NPN	MODULE_M3_10B << 16)
M3-10E VDC SRC 16DI/16DO	(MODULE_M3_10E << 16)
M3-11A +VS 16DI/16DO PNP	(MODULE_M3_11A << 16)
M3-11B +5V 16DI/16DO PNP	(MODULE_M3_11B << 16)
M3-12A 16 relay NO SPST	(MODULE_M3_12A << 16)
M3-12B 8 relay NO SPST	(MODULE_M3_12B << 16)
M3-13A 4 relay NO/NC SPDT	(MODULE_M3_13A << 16)
M3-14A 32 Sink Out	(MODULE_M3_14A << 16)
M3-14B 16 Sink Out	(MODULE_M3_14B << 16)
M3-15A 32 Src Out	(MODULE_M3_15A << 16)
M3-15B 16 Src Out	(MODULE_M3_15B << 16)
M3-15C 32 Src +5V Out	`(MODULE_M3_15C << 16)`
M3-16A 32 +VS Sink In	(MODULE_M3_16A << 16)
M3-16B 16 +VS Sink In	(MODULE_M3_16B << 16),
M3-16C 32 +5V Sink In	(MODULE_M3_16C << 16)
M3-16D 32 +5V Src In	(MODULE_M3_16D << 16)
M3-16E 32 +VS Src In	(MODULE_M3_16E << 16)
M3-16F 16 +VS Src In	(MODULE_M3_16F << 16)
M3-18A 16DI SPDT/16DO LED	(MODULE_M3_18A << 16)
M3-20A +VS 16DI/16DO NPN	(MODULE_M3_DUAL_20 << 16) | MODULE_M3_20A
M3-20B +5V 16DI/16DO NPN	(MODULE_M3_DUAL_20 << 16) | MODULE_M3_20C
"M3-21A +VS 16DI/16DO PNP	(MODULE_M3_DUAL_21 << 16) | MODULE_M3_21A
M3-21C +5V 16DI/16DO PNP	(MODULE_M3_DUAL_21 << 16) | MODULE_M3_21C
"M3-31A 16 +/-10V AIN	(MODULE_M3_DUAL_31 << 16) | MODULE_M3_31A
M3-31B 8 +/-10V AIN	(MODULE_M3_DUAL_31 << 16) | MODULE_M3_31B
M3-31C 16 4-20ma AIN	(MODULE_M3_DUAL_31 << 16) | MODULE_M3_31C
M3-31D 8 4-20ma AIN	(MODULE_M3_DUAL_31 << 16) | MODULE_M3_31D
M3-33A 8 Diff 20mv AIN	(MODULE_M3_DUAL_33 << 16) | MODULE_M3_33A

M3-33B 4 Diff 20mv AIN	(MODULE_M3_DUAL_33 << 16) | MODULE_M3_33B
M3-33C 8 Thermo 100mv AIN	(MODULE_M3_DUAL_33 << 16) | MODULE_M3_33C
M3-33D 4 Thermo 100mv AIN	(MODULE_M3_DUAL_33 << 16) | MODULE_M3_33D
M3-33E 8 Diff +/-10V AIN	(MODULE_M3_DUAL_33 << 16) | MODULE_M3_33E
M3-32A 32 +/-10V AOUT	(MODULE_M3_DUAL_32 << 16) | MODULE_M3_32A
M3-32B 16 +/-10V AOUT	(MODULE_M3_DUAL_32 << 16) | MODULE_M3_32B
M3-34A 8 4-20ma AOUT	(MODULE_M3_DUAL_34 << 16) | MODULE_M3_34A
M3-34B 4 4-20ma 4 0-10V	(MODULE_M3_DUAL_34 << 16) | MODULE_M3_34B
M3-40A 2 Axis Servo	(MODULE_M3_DUAL_40 << 16) | MODULE_M3_40A
M3-40B 2 Axis Stepper 24V	(MODULE_M3_DUAL_40 << 16) | MODULE_M3_40B
M3-40C 2 Axis Stepper 5V	(MODULE_M3_DUAL_40 << 16) | MODULE_M3_40C
M3-40D 1.5 Axis Stepper	(MODULE_M3_DUAL_40 << 16) | MODULE_M3_40D
M3-40E Diff Quad Counter	(MODULE_M3_DUAL_40 << 16) | MODULE_M3_40E
M3-40F Single End Quad Cntr	(MODULE_M3_DUAL_40 << 16) | MODULE_M3_40F
M3-40G 3 Axis Stepper	(MODULE_M3_DUAL_40 << 16) | MODULE_M3_40G
M3-41A EtherCAT Master	(MODULE_M3_DUAL_41 << 16) | MODULE_M3_41A
M3-61A DeviceNet Master	(MODULE_M3_DUAL_61 << 16) | MODULE_M3_61A
M3-61A DeviceNet Slave	(MODULE_M3_DUAL_61 << 16) | MODULE_M3_61B
M3-61C Ethernet/IP Master	(MODULE_M3_DUAL_61 << 16) | MODULE_M3_61C
M3-61D Ethernet/IP Slave	(MODULE_M3_DUAL_61 << 16) | MODULE_M3_61D

Blank

CHAPTER

7

[7] EtherCAT LED Indicators

RUN and ERR Indicators

In compliance with the EtherCAT Technology Group ETG1300 Indicator and labeling specification, whenever a 5300 EtherCAT slave option module is installed two LED's take on a different meaning from that of the normal 5300 controller. The LEDs are labeled RUN and ERR and operate as defined below.

RUN indicator green LED:

| Indicator states	Slave State	Description
Off	INITIALISATION	The device is in state INIT
Blinking	PRE-OPERATIONAL	The device is in state PRE-OPERATIONAL
Single Flash	SAFE-OPERATIONAL	The device is in state SAFE-OPERATIONAL
On	OPERATIONAL	The device is in state OPERATIONAL
Flickering	INITIALISATION or BOOTSTRAP	The device is booting and has not yet entered the INIT state, or: The device is in state BOOTSTRAP. Firmware download operation in progress

ERR indicator, red LED:

ERR State	Error Name	Description	Example
On	Application controller failure	An critical communication or application controller error has occurred	Application controller is not responding any more (PDI Watchdog Timeout detected by ESC)
n Flashes	Reserved	Reserved for future use	
Triple Flash	Reserved	Reserved for future use	
Double Flash	Process Data Watchdog Timeout/ EtherCAT Watchdog Timeout	An application watchdog timeout has occurred.	Sync Manager Watchdog timeout
Single Flash	Local Error	Slave device application has changed the EtherCAT state autonomously, due to local error (see ETG.1000 part 6 EtherCAT State Machine). Error Indicator bit is set to 1 in AL Status register.	Device changes its EtherCAT state from Op to SafeOpError due to a synchronization error.
Blinking	Invalid Configuration	General Configuration Error	State change commanded by master is impossible due to register or object settings, or invalid hardware configuration (pin sharing violation detected by ESC)
Flickering	Booting Error	Booting Error was detected. INIT state reached, but Error Indicator bit is set to 1 in AL Status register, or	Checksum error in Application controller flash memory.
Off	No error	The EtherCAT communication of the device is in working condition	

Blank

CHAPTER

8
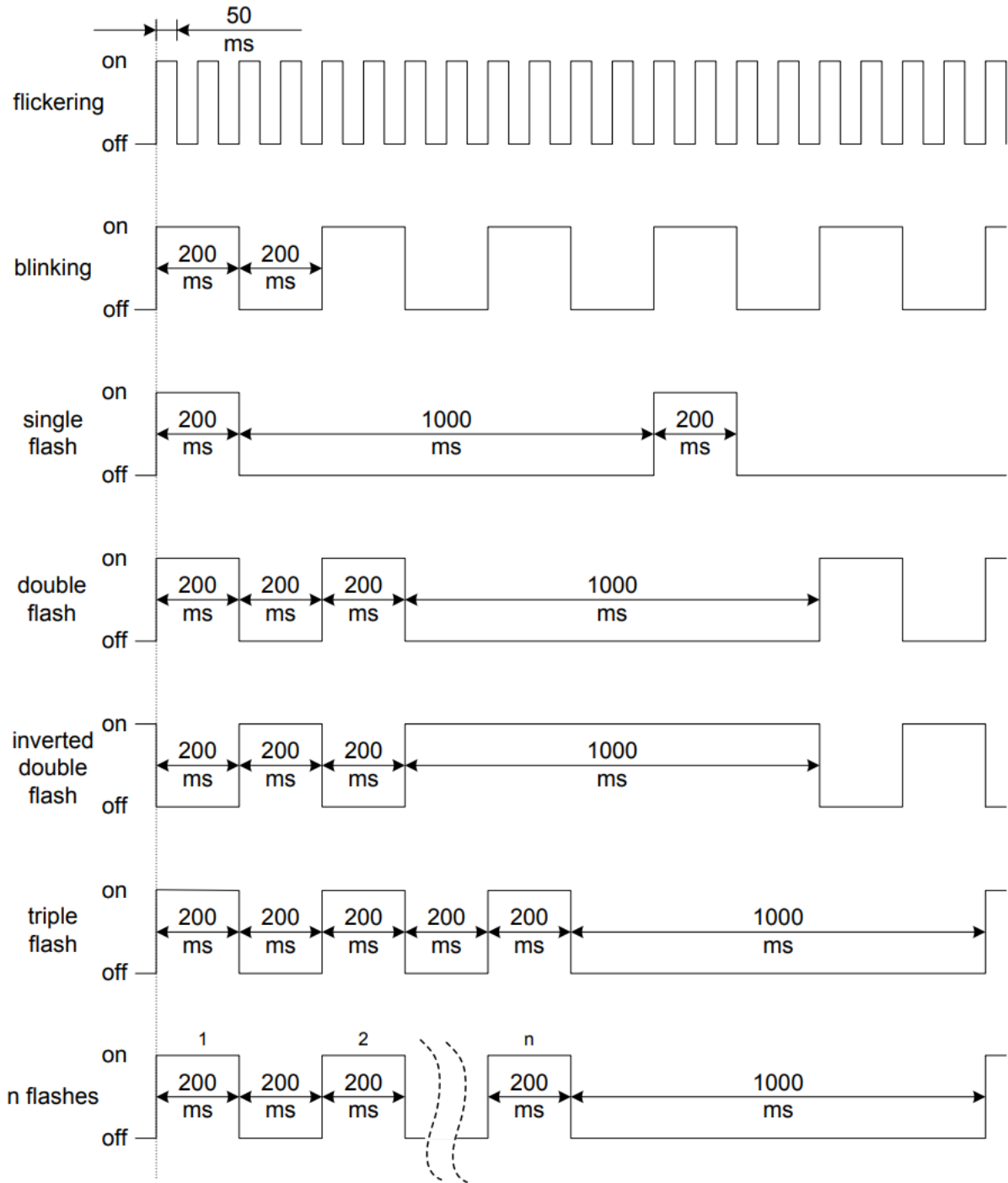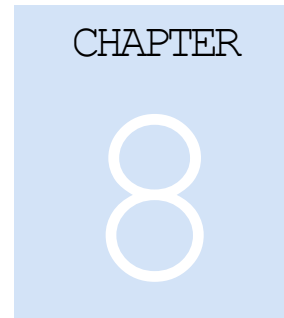
[8] Slave Conformance Testing

Conformance Testing

The 5300 PLC Controller has fully passed testing required by the EtherCAT Technology Group using the Beckhoff "EtherCAT Conformance Test Tool". This is a self-certification process whereby Control Technology Corporation maintains a yearly license for the tool and conducts its own rigorous testing to ensure compliance. CTC also has tested compatibility with its Incentive PC EtherCAT Master as well as the Beckhoff TwinCAT product.