

# WAGO SYSTEM **750**

## WAGO 750-841



## Quick Start Guide

Version 2.0.0

## TABLE OF CONTENTS

<b>ASSIGNING AN IP ADDRESS .....</b>	<b>4</b>
<b>PROGRAMMING EXAMPLE – STEP-BY-STEP LADDER DIAGRAM.....</b>	<b>10</b>
STARTING A NEW PROJECT .....	11
PLC CONFIGURATION.....	13
NETWORK 1: THE FIRST RUNG.....	18
NETWORK 2: THE SECOND RUNG .....	19
NETWORK 3: THE THIRD RUNG .....	21
NETWORK 4: THE FOURTH RUNG .....	23
CHECKING YOUR PROJECT FOR ERRORS .....	24
DOWNLOADING YOUR PROJECT.....	25
PROGRAMMING EXAMPLE IN FUNCTION BLOCK DIAGRAM.....	27
PROGRAMMING EXAMPLE IN STRUCTURED TEXT .....	28
<b>MODE SWITCH AND PROGRAMMING INTERFACE.....</b>	<b>29</b>
<b>CREATING A BOOT PROJECT.....</b>	<b>30</b>
<b>APPENDIX A - STANDARD FUNCTIONS AND FUNCTION BLOCKS .....</b>	<b>31</b>
<b>APPENDIX B - MEMORY ADDRESSING .....</b>	<b>33</b>

## **PURPOSE OF DOCUMENT**

This guide provides users with basic product information for creating applications using WAGO-IO-PRO CAA programming software with the WAGO 750-841 Programmable Fieldbus Controller (PFC). Each section of this guide gives simple how-to instructions and helpful hints that you will be able to apply to your application. The brief outline below indicates what you can expect to learn from each section of this document:

### **Assigning an IP Address**

This is a step-by-step procedure for assigning an IP address to the 750-841 PFC. Assigning an IP address is a prerequisite for communicating with the PFC through its Ethernet port.

### **Programming Example**

This programming example gives click-by-click instructions for building, downloading, and running a simple PFC application.

### **Mode Switch and Serial Programming Port**

Learn the characteristics of these important features of the WAGO PFC.

### **Creating a Boot Project**

Learn how to store your WAGO PFC program to the PFC's file memory so that it will be retained during power loss.

### **Standard Functions and Function Blocks**

Learn more about how to apply the building blocks of a WAGO-IO-PRO CAA application.

### **Memory Addressing**

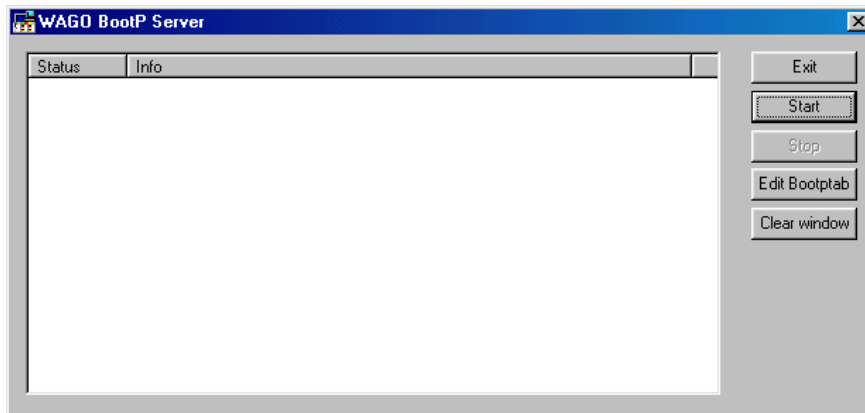
Learn the syntax used by WAGO-IO-PRO CAA to address memory locations in the PFC.

## ASSIGNING AN IP ADDRESS

This section provides a step-by-step procedure for assigning an IP address to the WAGO 750-841 PFC using WAGO's BootP Server. A copy WAGO's BootP Server can be downloaded from WAGO's website, which is located at <http://www.wago.com>.

Assigning an IP address is a prerequisite for communicating with the PFC through its Ethernet port.

Start the WAGO BootP Server. The screen below will appear.

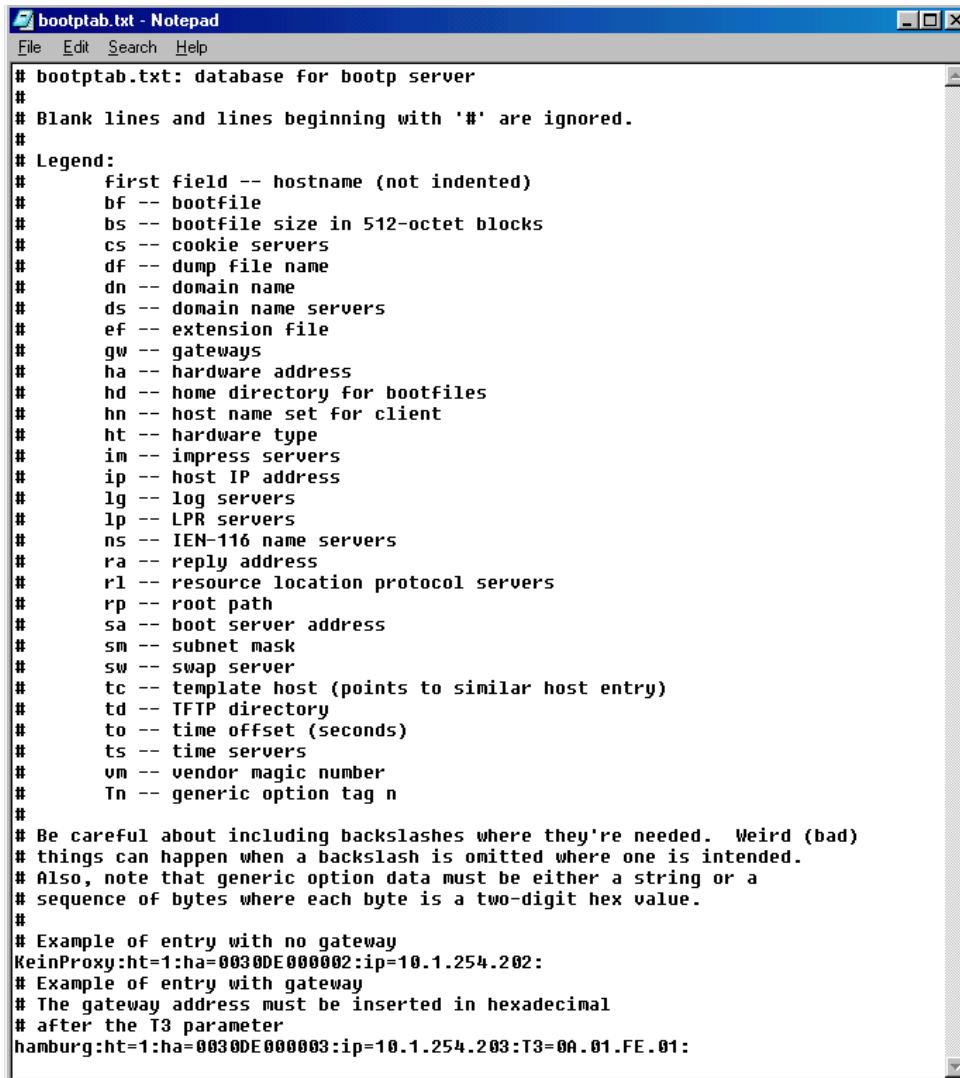


### BootP Protocol

BootP (bootstrap) protocol defines a request/response mechanism for assigning an IP address to a Bootp client (e.g., WAGO 750-841 PFC) based on its MAC-ID.

When the BootP client starts up, it broadcasts Bootp requests on the network. If a BootP server hears the request, and if the server is configured for the client, it will respond with the appropriate IP information. The client then uses this information to assign its IP address, subnet mask, and default gateway.

Click on **Edit Bootptab** command button. The Notepad editor will appear with a text file similar to the following:



```
# bootptab.txt: database for bootp server
#
# Blank lines and lines beginning with '#' are ignored.
#
# Legend:
#   First field -- hostname (not indented)
#   bf -- bootfile
#   bs -- bootfile size in 512-octet blocks
#   cs -- cookie servers
#   df -- dump file name
#   dn -- domain name
#   ds -- domain name servers
#   ef -- extension file
#   gw -- gateways
#   ha -- hardware address
#   hd -- home directory for bootfiles
#   hn -- host name set for client
#   ht -- hardware type
#   im -- impress servers
#   ip -- host IP address
#   lg -- log servers
#   lp -- LPR servers
#   ns -- IEN-116 name servers
#   ra -- reply address
#   rl -- resource location protocol servers
#   rp -- root path
#   sa -- boot server address
#   sm -- subnet mask
#   sw -- swap server
#   tc -- template host (points to similar host entry)
#   td -- TFTP directory
#   to -- time offset (seconds)
#   ts -- time servers
#   vm -- vendor magic number
#   Tn -- generic option tag n
#
# Be careful about including backslashes where they're needed. Weird (bad)
# things can happen when a backslash is omitted where one is intended.
# Also, note that generic option data must be either a string or a
# sequence of bytes where each byte is a two-digit hex value.
#
# Example of entry with no gateway
KeinProxy:ht=1:ha=0030DE000002:ip=10.1.254.202:
# Example of entry with gateway
# The gateway address must be inserted in hexadecimal
# after the T3 parameter
hamburg:ht=1:ha=0030DE000003:ip=10.1.254.203:T3=0A.01.FE.01:
```

The text file shown contains many lines, most of which begin with the '#' symbol. Any line that begins with a '#' symbol is a comment, and will not be processed. Look at the bottom line, beginning with the word **hamburg**. If a gateway address is not going to be used (typical), put a '#' symbol in front of this line. It is not needed.

Next look at the line beginning with the word **KeinProxy**. KeinProxy is a German term meaning 'Node Name'. This word can be changed to any descriptor you desire to identify your WAGO Ethernet IO device (e.g., WAGONode1).

The remaining contents of this line need to be modified to set the IP address of the WAGO Ethernet PFC.

Look on the right side of the WAGO PFC for the unique Mac ID of the device.

Example: MAC ID 0030DE006C5C

In the KeinProxy line, you will find the characters **ha=**. 'ha' is short for hardware address, also known as the MAC ID. After **ha=**, enter the MAC ID of your device.

Example: ha=0030DE006C5C

Also in the KeinProxy Line, you will find the characters **ip=**, short for IP address. Fill in the desired IP address for the WAGO PFC. This will be the IP address of the device on the network.

Example: ip=10.5.0. 2

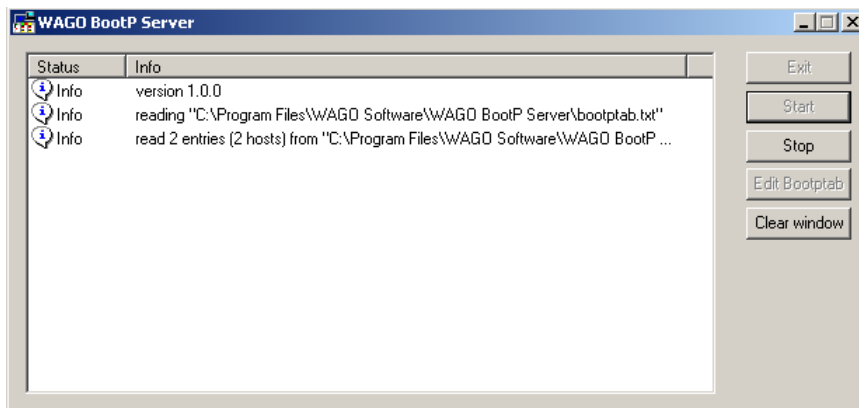
If a Gateway is to be used then fill in:

ha= ( MAC ID) : ip= (IP Address): T3= (Gateway Address)

When all this data has been entered it must be saved. At the top of the Notepad editor, Click **File..Save** to save the text file (Do not change the file name.).

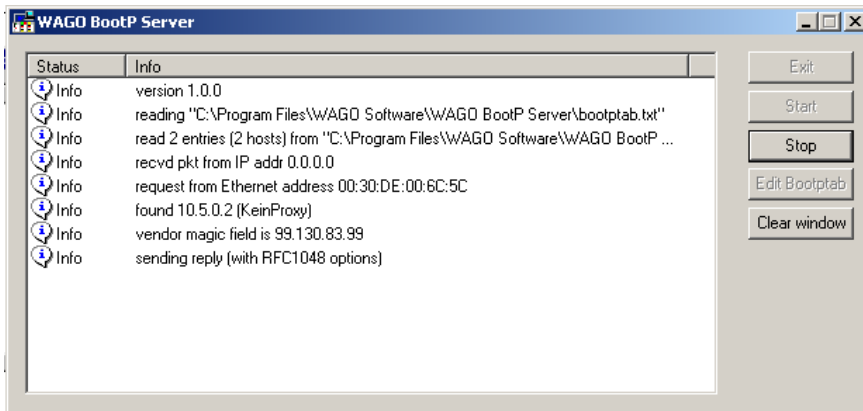
Close the Notepad editor. The WAGO BootP Server is again displayed.

Click the **Start** command button. Three or five status messages will appear, similar to the following screen capture:



Turn power off to the WAGO PFC, and wait for a couple of seconds.

Turn power back on to the WAGO PFC. You will see more status messages scroll down, similar to the following screen capture:



Verify the status information displayed:

Packets were received from an IP Address

The Ethernet Address is the same as the MAC ID of the PFC

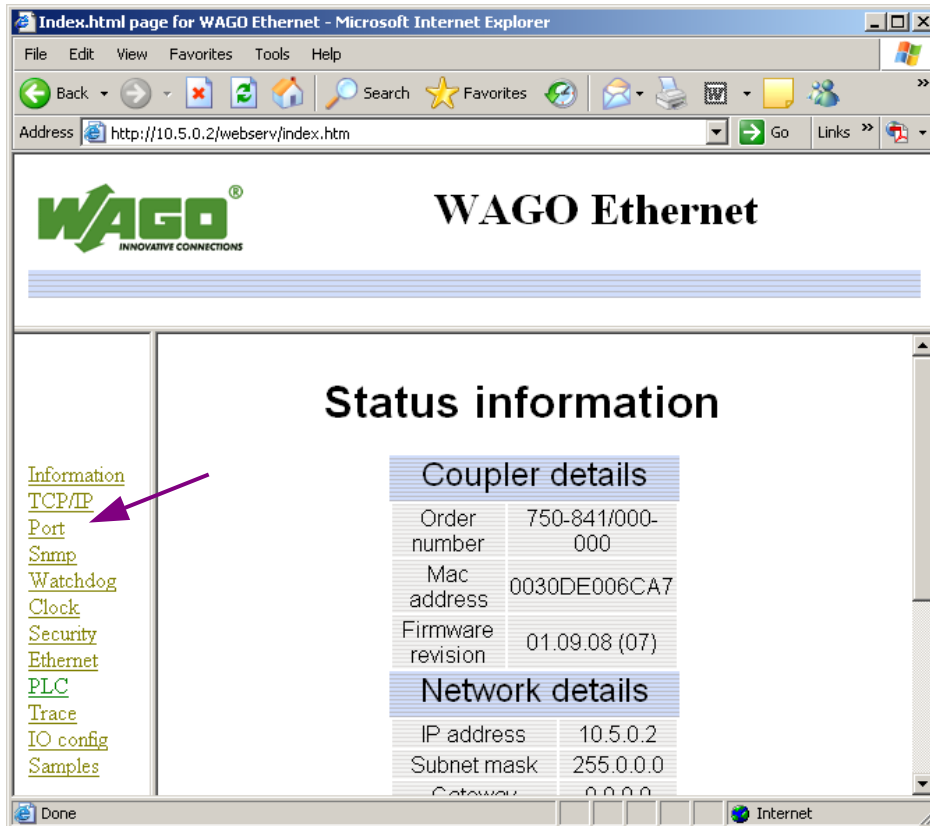
The KeinProxy, or node identifier, and IP address match what was entered in the text file using Notepad.

Verify the WAGO PFC is operating correctly. Its diagnostic LED's should be illuminated as follows:

<b>Link</b>	Green
<b>MS (Module Status)</b>	Green
<b>NS (Node Status)</b>	Green or Flashing Green
<b>TxD/RxD</b>	Flash as data is Sent/Received
<b>I/O</b>	Green

At this point the WAGO PFC contains an IP address. The following steps will store the assigned IP address to EEPROM so it's retained during power loss

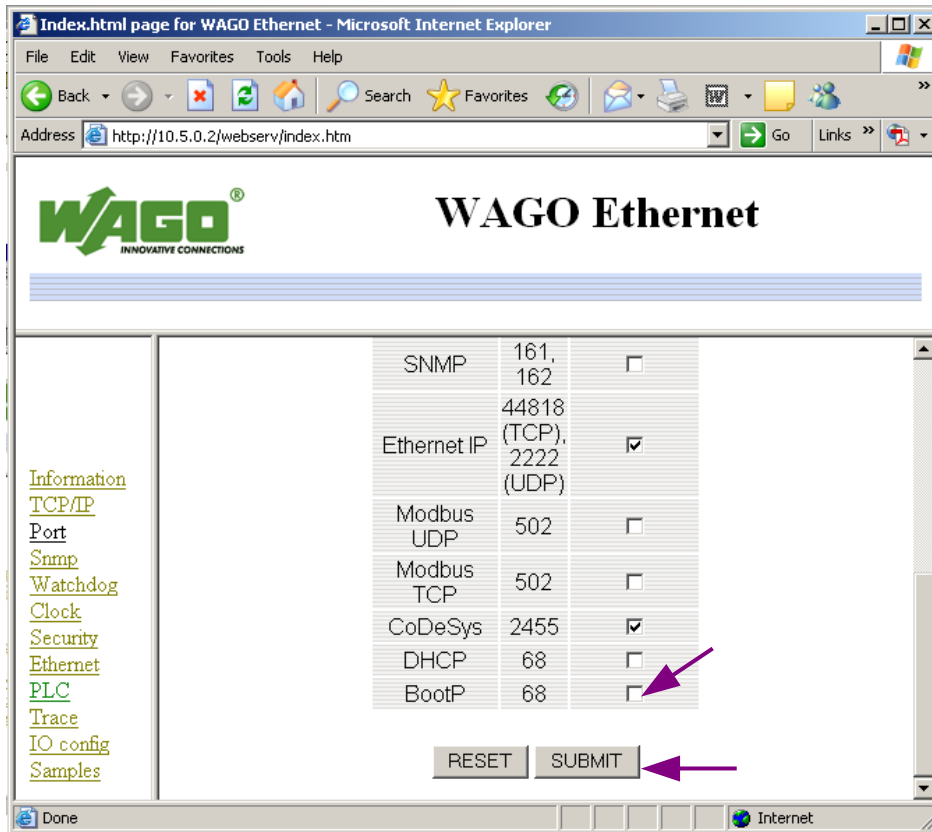
Launch Microsoft Internet Explorer. In the address box enter the IP address of your 750-841 PFC and press the **Enter** key. Internet Explorer will display a screen similar to the following:



Click the **Port** hyperlink on the left-hand side of the screen. A password dialog box will be displayed. Enter “**admin**” for the user name and “**wago**” for the password.



The following screen is displayed:



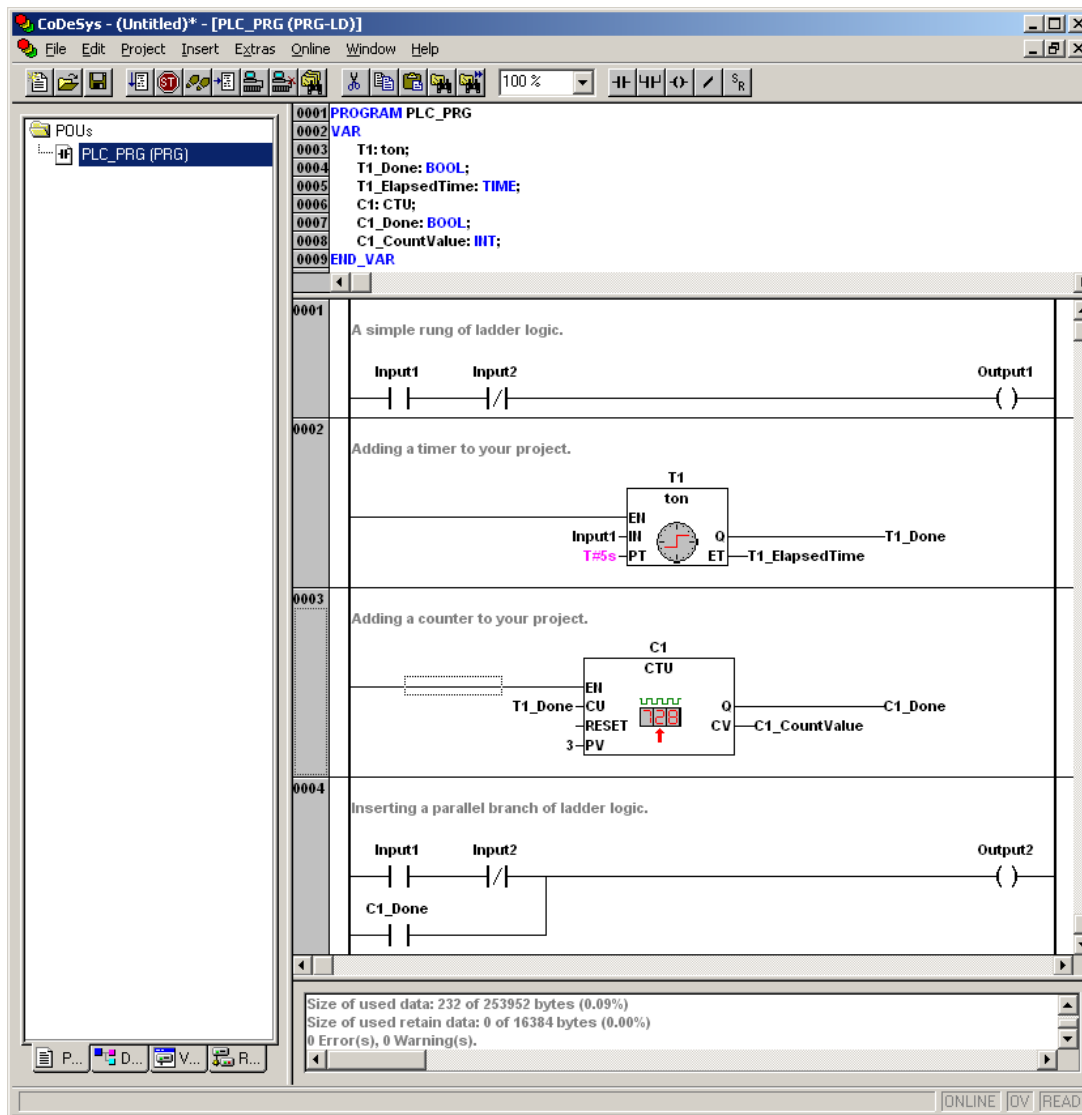
Unselect the **BootP** check-box and click the **SUBMIT** command button.

Your WAGO 750-841 PFC now contains a static IP address and is ready for communications on an Ethernet network.

## PROGRAMMING EXAMPLE – STEP-BY-STEP LADDER DIAGRAM

The following example provides step-by-step instructions for creating a WAGO-IO-PRO CAA project using the Ladder Diagram programming language. This procedure will familiarize you with many basic concepts used in WAGO-IO-PRO CAA.

### Programming Example – Ladder Diagram

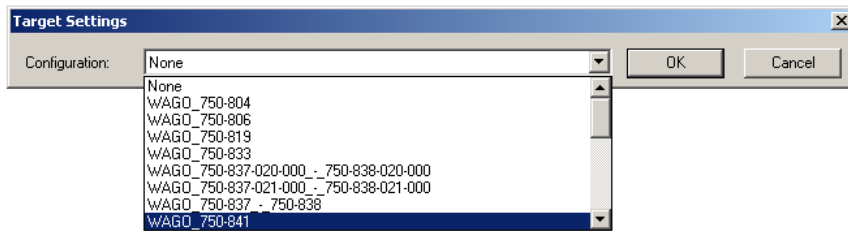


## STARTING A NEW PROJECT

If it is not started already, start WAGO-IO-PRO CAA.

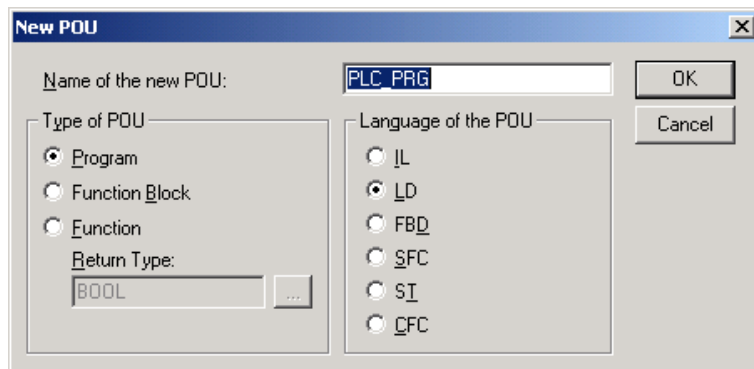
From the top menu bar, select **File..New**. You may be asked if you'd like to save your existing program. Answer accordingly.

A *Target Settings* window will appear. Select the **WAGO\_750-841** or **WAGO\_750-841\_DEMO** depending on the target support package installed on your computer.



After the target is selected, several configuration options will appear. Keep the default settings and Click **OK** to continue.

A window will appear labeled *New POU*.



Complete the New POU information as follows:

Name of the new POU: **PLC\_PRG**

Type of the POU: **Program**

Language of the POU: **LD**

Click **OK** to continue.

## ABOUT POUS

Within WAGO-IO-PRO CAA, programs, function blocks, and functions are called Program Organization Units, or POUs. Along with operators, POUs are the building blocks from which a project is created. POUs are referred to as being either *standard* or *user-defined*.

Standard POUs reside within libraries that you have included in your project. Select **Window..Library Manager** from the top menu bar to view the libraries that are included in your project. When viewing the Library Manager, you can click **Insert..Additional Library** to add libraries to your project.

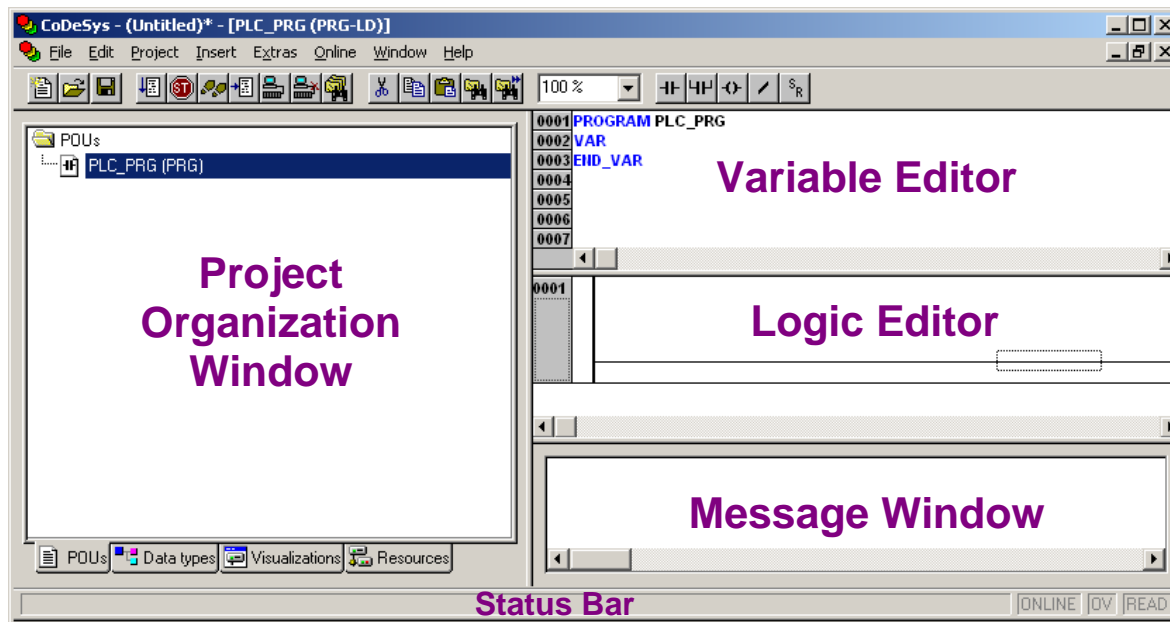
User-defined POUs are those that you have created or copied into your project from other applications.

Every project needs one POU entitled PLC\_PRG (assuming the Task Manager is not used). PLC\_PRG is the main program that can contain all the logic in a project, or can call other POUs containing additional logic. Project execution will begin with the contents of PLC\_PRG.

POUs can be written in the following IEC-61131 languages:

- Instruction List (IL)
- Ladder Diagram (LD)
- Function Block Diagram (FBD)
- Sequential Function Chart (SFC)
- Structured Text (ST)
- Continuous Function Chart (CFC)

The WAGO-IO-PRO CAA (CoDeSys) development environment is displayed. Below is a brief description of the various fields.



**Variable Editor** - Between the keywords VAR and END\_VAR all of the local variables for a Program Organization Unit (POU) are declared.

**Logic Editor** – This is where the body of your control program is written. A text or a graphic editor will be displayed based on the POU Language that is used. Ladder Diagram, Function Block Diagram, Sequential Function Chart, and Continuous Function Chart use a graphical editor. Instruction List and Structure Text use a text-based editor.

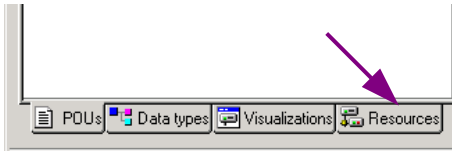
**Project Organization Window** – With the POUs tab selected, this window displays the user POUs (Programs, Function blocks, and Functions) that make up your project. These objects consist of a variable declaration part and a logic/program part.

**Message Window** – Displays messages regarding the last compile, checking, or comparing operation.

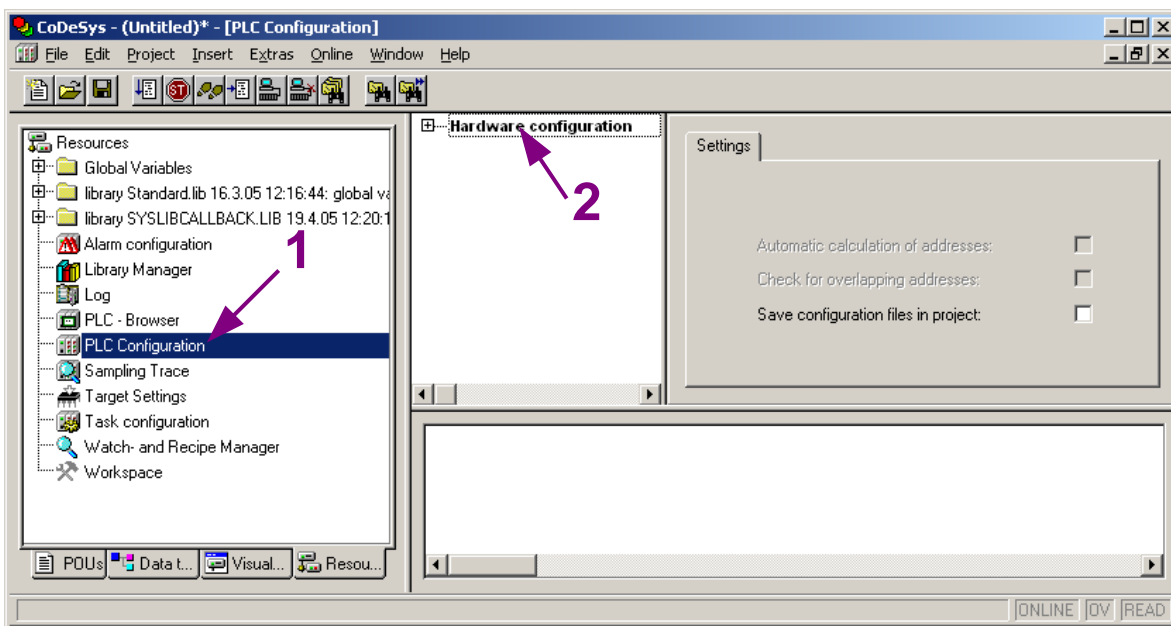
**Status Bar** – The status bar is at the bottom of the window frame and gives information about the current project and menu commands.

## PLC CONFIGURATION

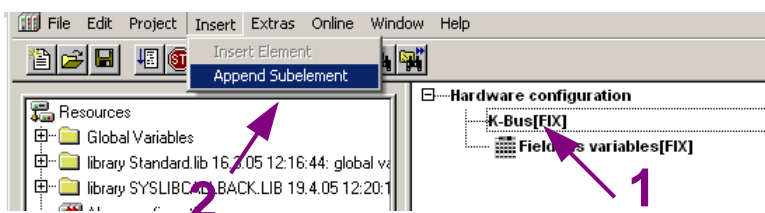
Before writing your control program, the Hardware Configuration Editor is used to map the attached I/O modules of the WAGO node to the input and output process image. Individual I/O points are assigned global variable names for use in your control program. To open the Hardware Configuration Editor, begin by clicking the **Resources** tab at the bottom of the Project Organization Window.



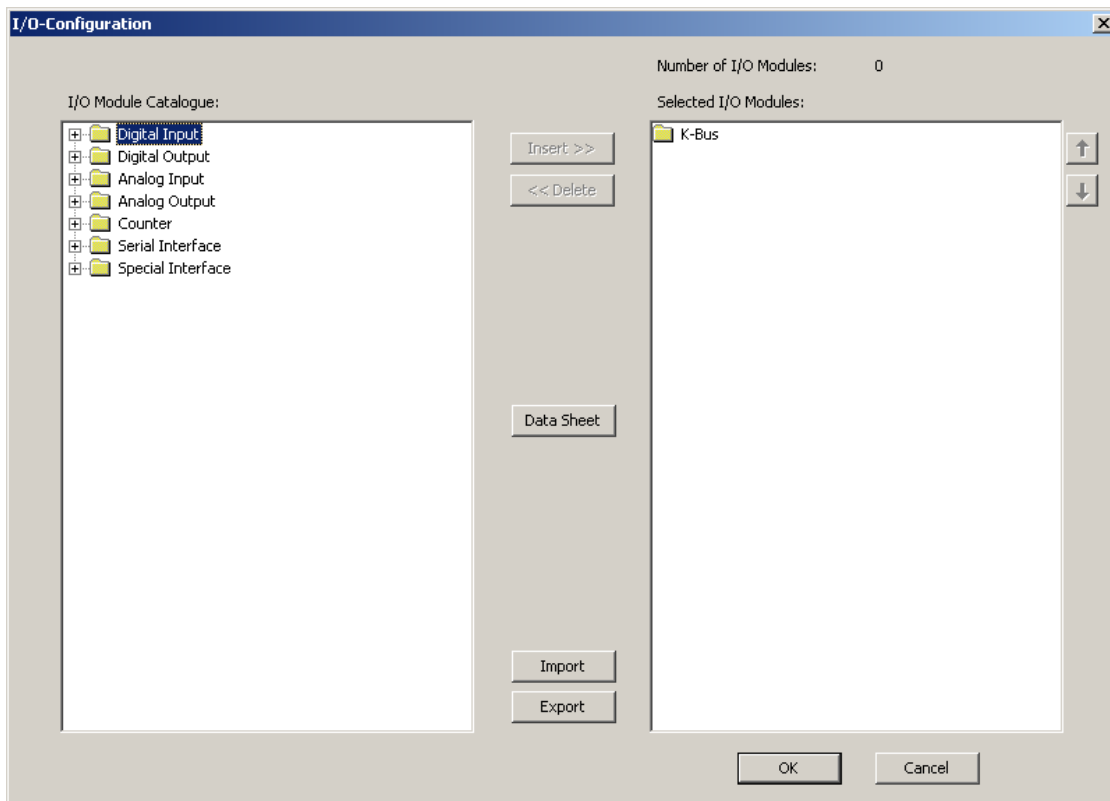
Double-click **PLC Configuration** in the Project Organization Window, then double-click **Hardware configuration** in the window to the right, as shown below.



Click on **K-Bus[FIX]** and then select **Insert...Append Subelement** from the top menu bar.



The following window appears:



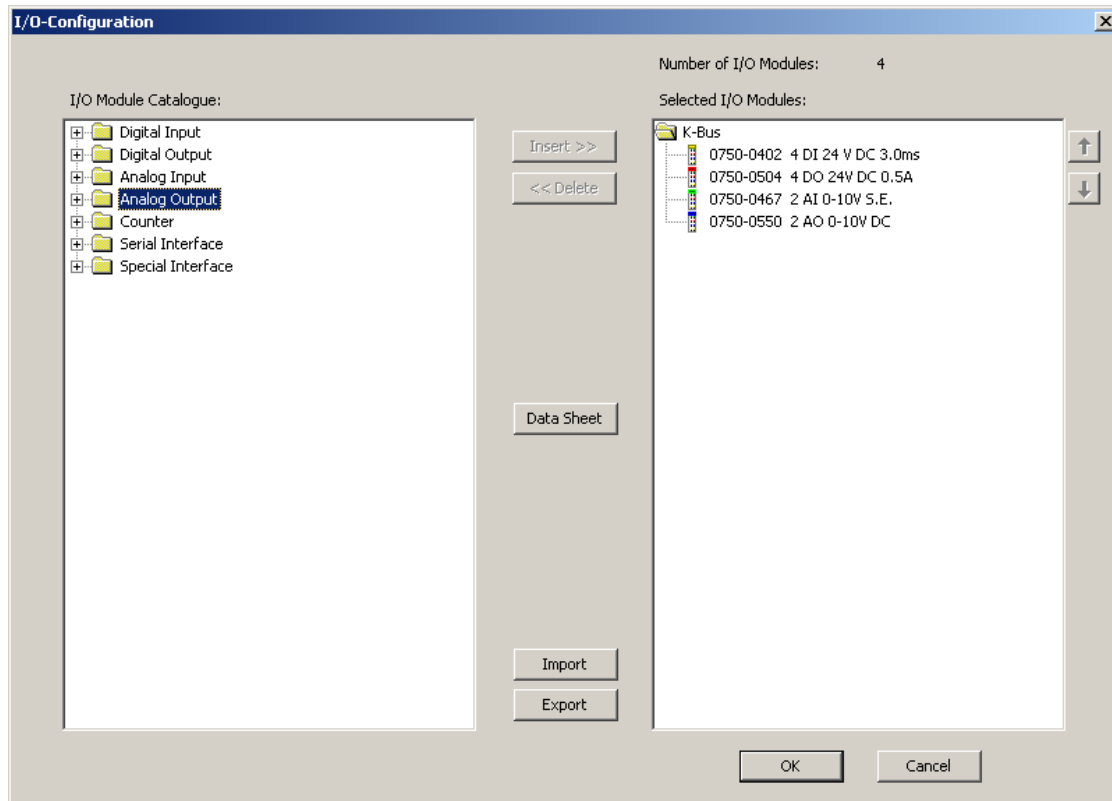
Click the “+” sign in front of the *Digital Input* entry in the *I/O Module Catalogue* window and select the **0750-0402 4 DI 24VDC 3.0ms** module. Then click the **Insert** button, the module will appear in the *Selected I/O Modules* window. Repeat this process for the following other modules:

Digital Output - 0750-0504 4 DO 24VDC 0.5A  
 Analog Input - 0750-0467 2 AI 0-10V S.E.  
 Analog Output - 0750-0550 2 AO 0-10V DC

The 750-600 End Module is not added to the I/O configuration since it does not consume or produce process data.

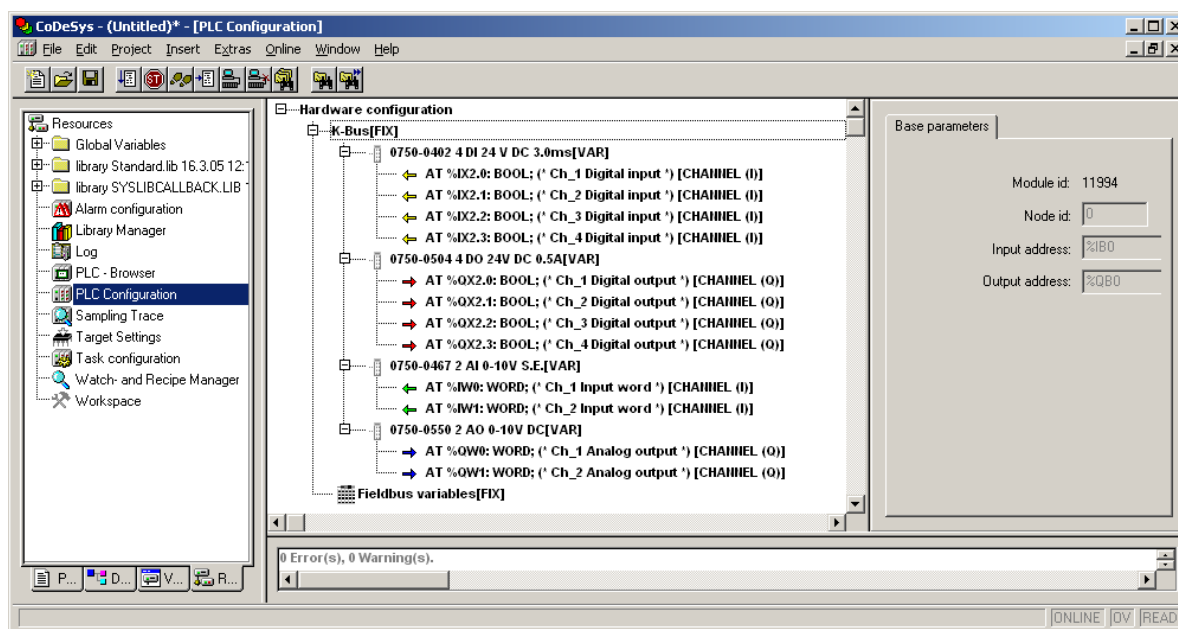
Note: This example program assumes that you have a WAGO Demo Node (750-841 Ethernet Controller, 750-402 24VDC 4-Channel Digital Input, 750-504 24VDC 4-Channel Digital Output, 750-467 2-Channel 0-10VDC Analog Input, 750-550 2-Channel 0-10VDC Analog Output, and 750-600 End Module). If your node configuration is different, choose the appropriate I/O modules for your node.

When complete, your screen should look similar to the one below. Click **OK** to continue.



Note: If you wish to delete a module from your selection, select the appropriate module in the *Selected I/O Modules* window and click the Delete button. Additionally, the arrow buttons on the right-hand side of the I/O-Configuration window allows for repositioning of selected modules.

The *Hardware configuration* window regains focus. Now click the + signs in front of K-Bus[FIX] and the added modules. The IEC61131 memory addresses (e.g., %IX2.0) are displayed for each I/O point defined in the hardware configuration.

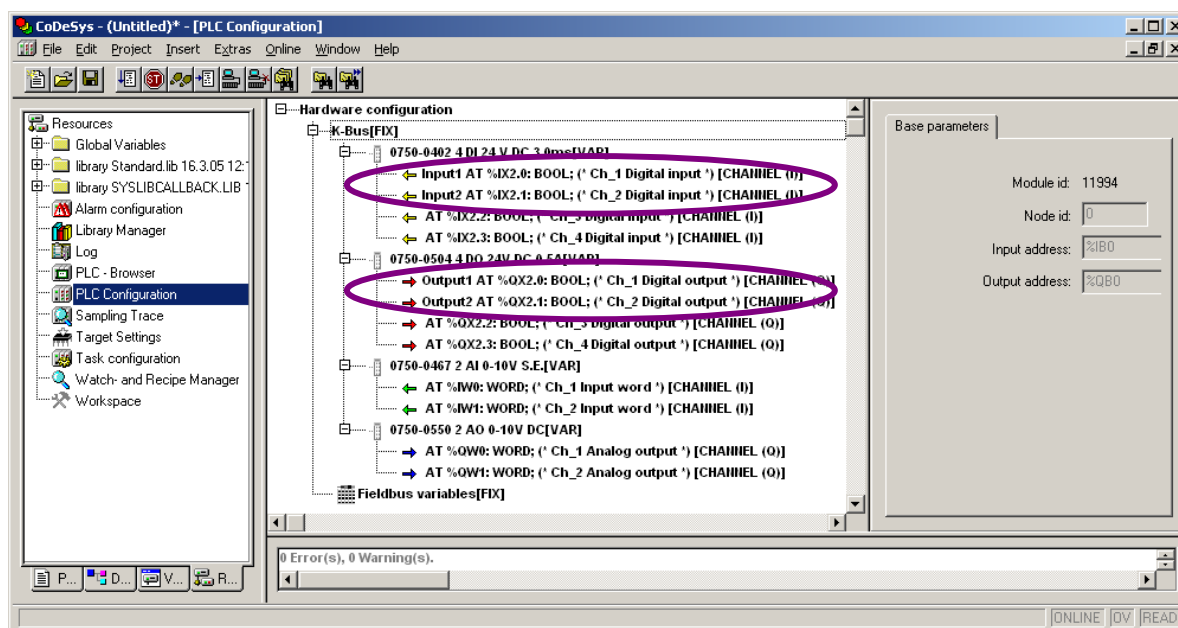


Double-clicking on the keyword **AT** will open an entry field, which allows you to enter a global variable name for each I/O point.

Enter the following variable names:

- Input1** – for the first digital input (%IX2.0)
- Input2** – for the second digital input (%IX2.1)
- Output1** – for the first digital output (%QX2.0)
- Output2** – for the second digital output (%QX2.1)

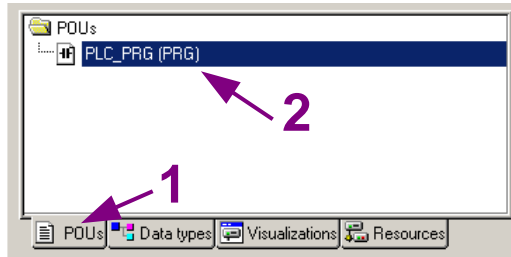
When complete, your screen will look similar to the one below.



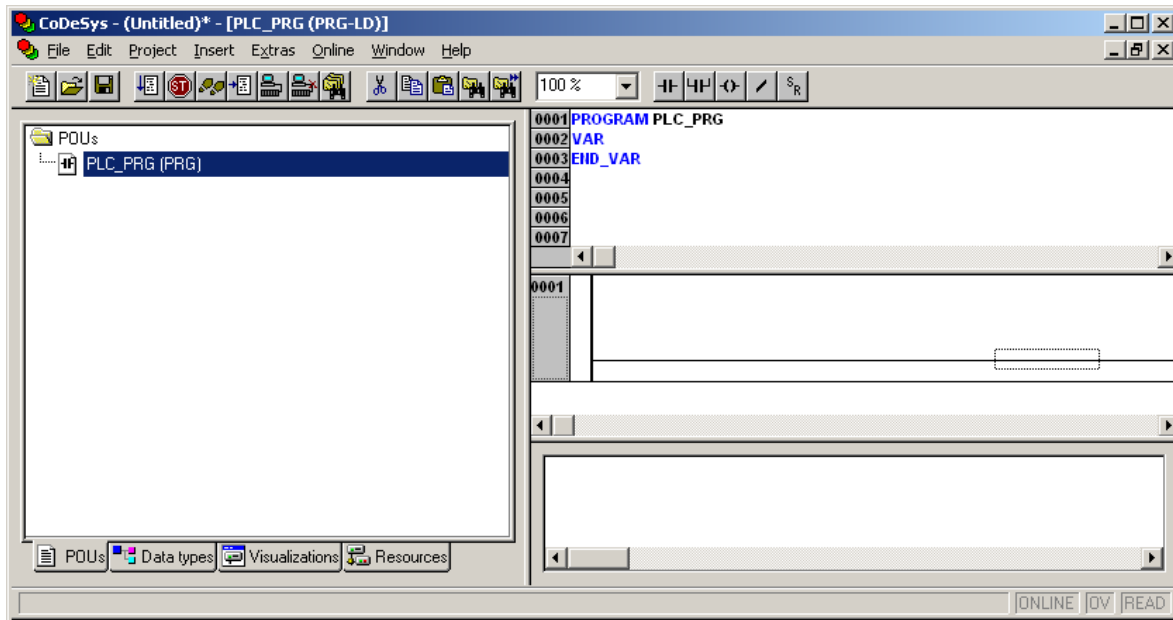


The Hardware Configuration is complete.

The next step is to write your control program in the Logic Editor. To display the Logic Editor, click the **POUs** tab at the bottom of the Project Organization Window and then double-click **PLC\_PRG**.



The Logic Editor is displayed with one empty rung.



## NETWORK 1: THE FIRST RUNG

Click on the empty rung of ladder logic to position the Logic Editor cursor.

Click on the Contact icon in the toolbar.



A normally-open contact will appear in the Logic Editor.

Highlight the '???' above the contact by clicking on it. Press **F2** to display the *Help Manager*. In the left window of the *Help Manager* select **System Variables**. Now in the right window, select the variable **Input1**, and click **OK**.

Click in the Logic Editor to the right of the contact you just entered to relocate the Logic Editor cursor.

Click on the Contact icon in the toolbar.



A second normally-open contact will appear in the Logic Editor. Highlight the '???' above the contact and press **F2** to display the *Help Manager*. Select the variable **Input2** and click **OK**.

Click on the normally-open contact you just created.

Click on the Negate icon in the toolbar.



The contact now appears as normally-closed.

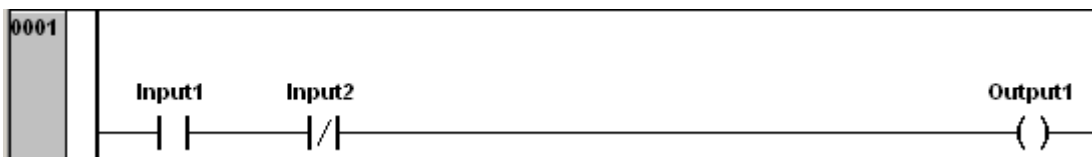
Click in the Logic Editor to the right of the contact you just entered to relocate the Logic Editor cursor.

Click on the Coil icon in the toolbar.



A coil will appear in the Logic Editor. '???' appears above the coil and is highlighted for data entry. Press **F2** to display the *Help Manager*. Select the variable **Output1** and click **OK**.

The first rung of logic is now complete, and should appear as shown below. Verify that this rung is correct before proceeding to the next rung.



### ABOUT VARIABLES

Variable names can be any sequence of up to 32 alphanumeric characters, with the following restrictions:

- Variable names cannot begin with a numeric digit (0-9).
- Variable names cannot contain any blank spaces.
- The underscore ( \_ ) character is a valid character, but a variable name cannot include consecutive underscore characters.
- Variable names cannot be the same as any keywords (e.g. FOR, AND, MOVE, and CASE are all examples of keywords, and therefore are not valid variable names)

Variable names are not case sensitive; the variable name input\_10, Input\_10, and INPUT\_10 will all be recognized as the same variable. Any variable name used in a project must be declared, either locally in the declaration part of a POU, or in a global variable list.

If you would like to change a variable declaration statement, you can do so by typing the changes directly in the Variable Editor window.

You can also change the variable declaration statement by reopening the Declare Variable window. This is done by clicking on the variable name in the project, and selecting from the top menu bar **Edit..Auto Declare...**

## NETWORK 2: THE SECOND RUNG

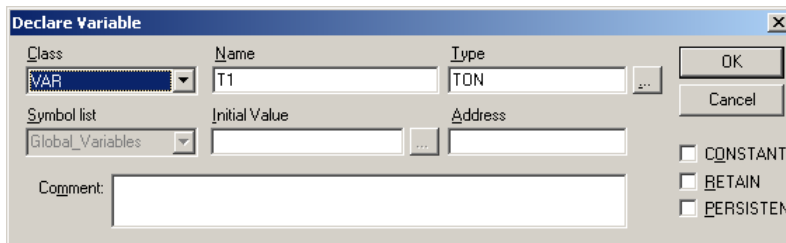
From the top menu bar, select **Insert..Network After** to create a second rung of ladder.

From the top menu bar, select **Insert..Box with EN**. The **AND** operator box will appear in your logic.

Click on the word **AND**, type **TON**, and hit **Enter**. The **TON** (On-Delay Timer) function block is now displayed.

Click on the '???' above the function block.

Type in **T1**, and hit **Enter**. T1 is the instance name of the TON function block. A window will appear labeled *Declare Variable*



The 'Declare Variable' dialog box is shown. It has a 'Class' dropdown set to 'VAR', a 'Name' field containing 'T1', and a 'Type' dropdown set to 'TON'. There are fields for 'Symbol list' (set to 'Global\_Variables'), 'Initial Value', and 'Address'. A 'Comment' text area is at the bottom. On the right, there are 'OK' and 'Cancel' buttons, and three checkboxes: 'CONSTANT', 'RETAIN', and 'PERSISTENT'.

The *Declare Variable* window will assist you in declaring new variables (or objects) in your project, and will appear whenever a variable is used that is yet to be declared.

Enter the following information in the Declare Variable window for T1:

Class **VAR** Name **T1** Type **TON**

Click **OK** when completed.

Notice that the variable declaration statement:

**T1: TON;**

now appears in the Variable Editor window. The variable is now declared.

Click on the '???' preceding the IN input of the TON function block.

Press **F2** to display the *Help Manager*. Select the variable **Input1** and click **OK**.

Click on the '???' preceding the PT input of the TON function block.

Type **T#5s** and hit **Enter**. This sets the Preset Time of the timer equal to five seconds.

Click on the '???' to the right of the Q output of the TON function block.

Type in '**T1\_Done**', and hit **Enter**.

Notice that the Auto Declare window appears. This is because the variable you entered was not already declared.

Enter the following information in the Declare Variable window:

Class **VAR** Name **T1\_Done** Type **BOOL**

Click **OK** when completed.

### More on Timers

The On-Delay Timer TON is just one of three timers available in the WAGO-IO-PRO CAA standard library (also TOF and TP).

The WAGO-IO-PRO CAA Help Menu is an excellent reference for information on timers. It gives the definition and datatype of the inputs and outputs for each timer function block. It also shows examples of how to use each of the timers in several of the IEC-61131 programming languages.

To view help on timers, select from the top menu bar **Help..Contents**, click the **Index** tab, and type in 'TON', 'TOF', or 'TP'.

Notice that the variable declaration statement:

**T1\_Done: BOOL;**

now appears in the Variable Editor window. The variable is now declared.

Click to the right of the TON output ET (**do not click on the line – click to the right of it**).

Type in '**T1\_ElapsedTime**', and hit **Enter**. This variable will contain the Elapsed Time of the TON function block T1.

Enter the following information in the Declare Variable window:

Class **VAR** Name **T1\_ElapsedTime** Type **Time**

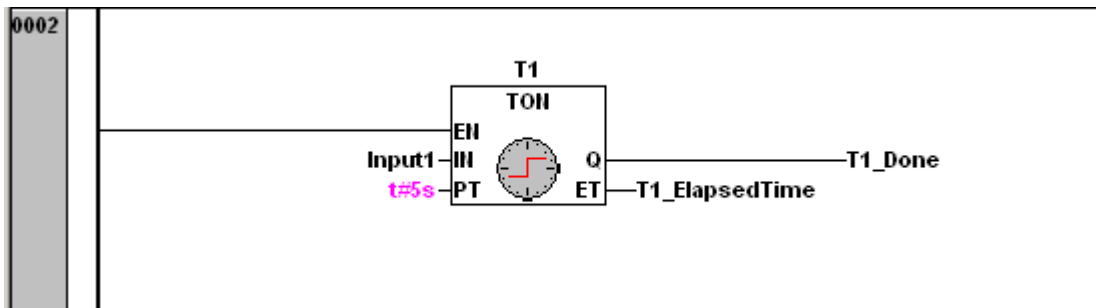
Click **OK** when completed.

Notice that the variable declaration statement:

**T1\_ElapsedTime: Time;**

now appears in the Variable Editor window. The variable is now declared.

The second rung of logic is now complete, and should appear as shown below. Verify that this rung is correct before proceeding to the next rung.



### NETWORK 3: THE THIRD RUNG

From the top menu bar, select **Insert..Network After** to create a third rung of ladder.

From the top menu bar, select **Insert..Box with EN**. The *AND* operator box will appear in your logic.

Click on the word *AND*, type **CTU**, and hit **Enter**. The *CTU* function block is now displayed.

Click on the '???' above the function block.

Type in **C1**, and hit **Enter**. C1 is the instance name of the CTU function block.

Enter the following information in the Declare Variable window:

Class **VAR** Name **C1** Type **CTU**

Click **OK** when completed.

Click on the '???' preceding the CU input of the CTU function block.

Type in the variable name **T1\_Done**, and hit **Enter**.

Notice that the Auto Declare window does not appear. This is because the variable you entered is already declared.

Click on the '???' preceding the RESET input of the CTU function block.

Hit **Delete**. This example will not make use of the reset operation of the CTU function block.

Click on the '???' preceding the PV input of the CTU function block.

Type **3** and hit **Enter**. This sets the Preset Value of the counter equal to three.

Click on the '???' to the right of the Q output of the CTU function block.

Type in '**C1\_Done**', and hit **Enter**. Enter the following information in the Declare Variable window:

Class **VAR** Name **C1\_Done** Type **BOOL**

Click **OK** when completed.

Click to the right of the CTU output CV (**do not click on the line – click to the right of it**).

Type in '**C1\_Value**', and hit enter. This variable will contain the Current Value of the CTU function block C1.

Enter the following information in the Declare Variable window:

Class **VAR** Name **C1\_Value** Type **WORD**

Click **OK** when completed.

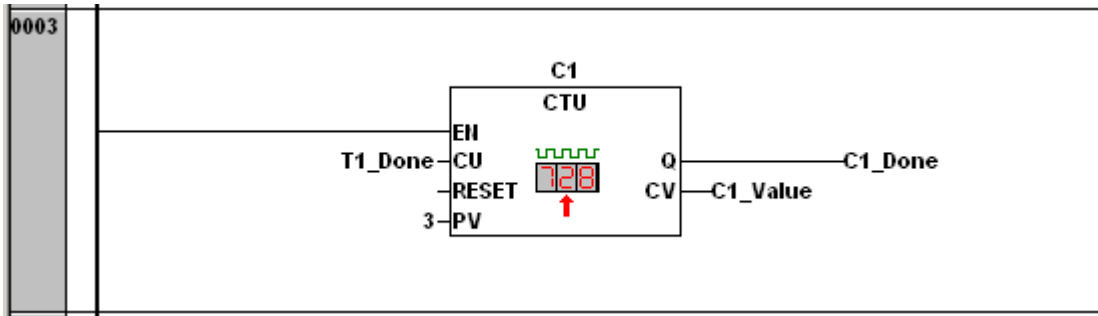
#### More on Counters

The Up Counter CTU is just one of three counters available in the WAGO-IO-PRO CAA standard library (also CTD and CTUD).

The WAGO-IO-PRO CAA Help Menu is an excellent reference for information on counters. It gives the definition and datatype of the inputs and outputs for each counter function block. It also shows examples of how to use each of the counters in several of the IEC-61131 programming languages.

To view help on counters, select from the top menu bar **Help..Contents**, click the **Index** tab, and type in 'CTU', 'CTD', or 'CTUD'.

The third rung of logic is now complete, and should appear as shown below. Verify that this rung is correct before proceeding to the next rung.



## NETWORK 4: THE FOURTH RUNG

The fourth rung has similarities to the first, so let's begin by copying and pasting the first rung of logic. Proceed as follows:

Click on the first rung. (Do not click on a contact or coil.)

From the top menu bar, select **Edit..Copy**.

Click on the third rung. From the top menu bar, select **Insert..Network After** to create a fourth rung of ladder.

From the top menu bar, select **Edit..Paste**. The contents of the first rung will be pasted in. (Note that an empty fifth rung of logic is now present. This can be deleted by clicking on the fifth rung, and hitting **Delete**.)

Change the variable name on the coil of the fourth rung by highlighting the variable name **Output1** and press **F2** to display the *Help Manager*. Select the variable **Output2** and click **OK**.

To insert a parallel branch around the normally-open and normally-closed contacts, proceed as follows: Click on the normally-open contact. Hold down the shift key and click on the normally-closed contact. Both contacts are now highlighted. Click on the Parallel Contact icon in the toolbar.



Type **C1\_Done** above the new contact, and hit **Enter**.

The fourth rung of logic is now complete, and should appear as shown below. Verify that this rung is correct before proceeding to the next section.



### Saving Your Project

It is recommended that you save your project regularly to protect against losing valuable work.

To perform simple saves, select **File..Save** or **File..Save As** from the top menu bar.

You can configure WAGO-IO-PRO CAA to automatically save your project at regular intervals. To enable this feature, select **Project..Options...** from the top menu bar. Select the category **Load & Save**, check the **Auto Save** option, and alter the **Auto Save Interval** as desired.

You can also configure WAGO-IO-PRO CAA to automatically save your project each time it is compiled (i.e. each time it is checked for errors). To enable this feature, select **Project..Options...** from the top menu bar. Select the category **Load & Save**, and check the **Auto save before compile** option.

## CHECKING YOUR PROJECT FOR ERRORS

At any time during the development of a project, you can check for errors by selecting **Project..Build** from the top menu bar.

When you check your project for errors, diagnostic messages will appear below the Logic Editor window. The last diagnostic message to appear will indicate how many errors you have in your project. Error messages include the line number of where the error exists, as well as a description of the error.

You will need to correct all errors before being permitted to download and run your project.

### Debug Tip

If you receive multiple error messages when checking your project, it is recommended that you scroll up to the first error message to appear, and start your debug process there. Multiple error messages are often the result of a single error, and the first error message to appear will give you the best indication of where that error exists.



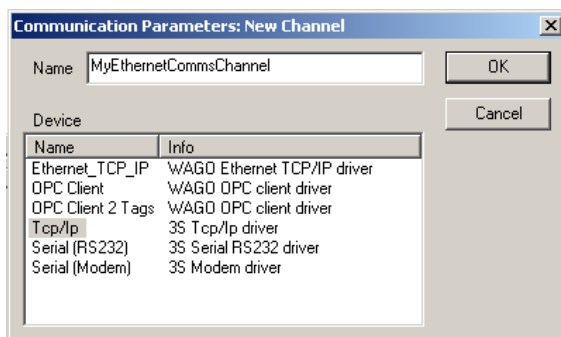
## DOWNLOADING YOUR PROJECT

If you have not configured a communications channel for communicating with your PFC, you will need to do so. Communication channel parameters vary depending on the PFC and communications port being used.

The following example shows how to setup a communications channel between a computer with an Ethernet adapter and a 750-841 WAGO Ethernet PFC.

Select **Online..Communication Parameters...** to configure the communications channel to your PFC.

Click **New....** The following window will appear:



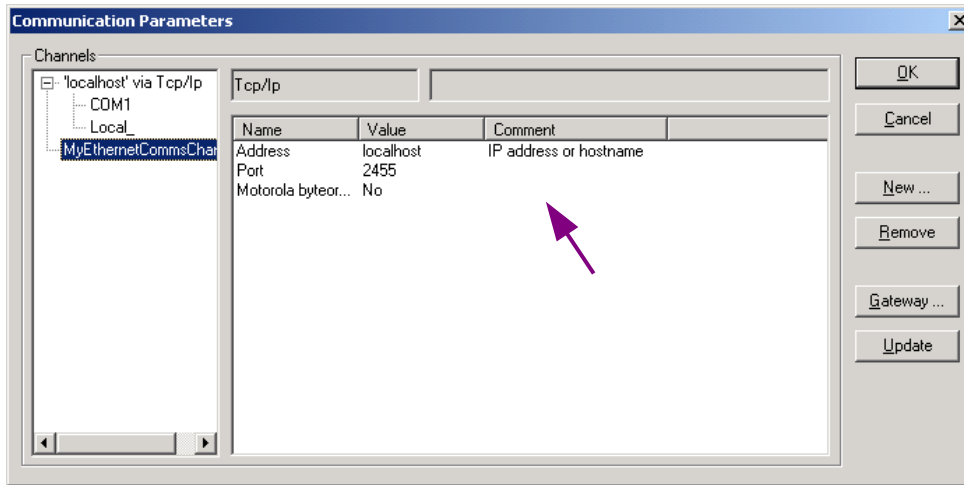
In the *Name* field, type in a unique name for your communications channel.

In the selection list, select:

**Tcp/Ip (3S Tcp/Ip driver)**  
as the communications driver.

Click **OK**.

A window similar to the one below will appear:

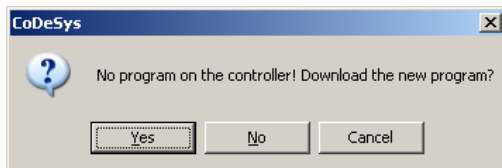


Double-click on the word **localhost**. An entry field will appear. Enter in the IP address of your WAGO Ethernet PFC, and hit **Enter**.

Click **OK**. (Be sure to hit Enter before clicking OK.)

You should now be ready to Login to the PFC and download your project.

Click **Online..Login**. The window below will appear, prompting you to download your new program.

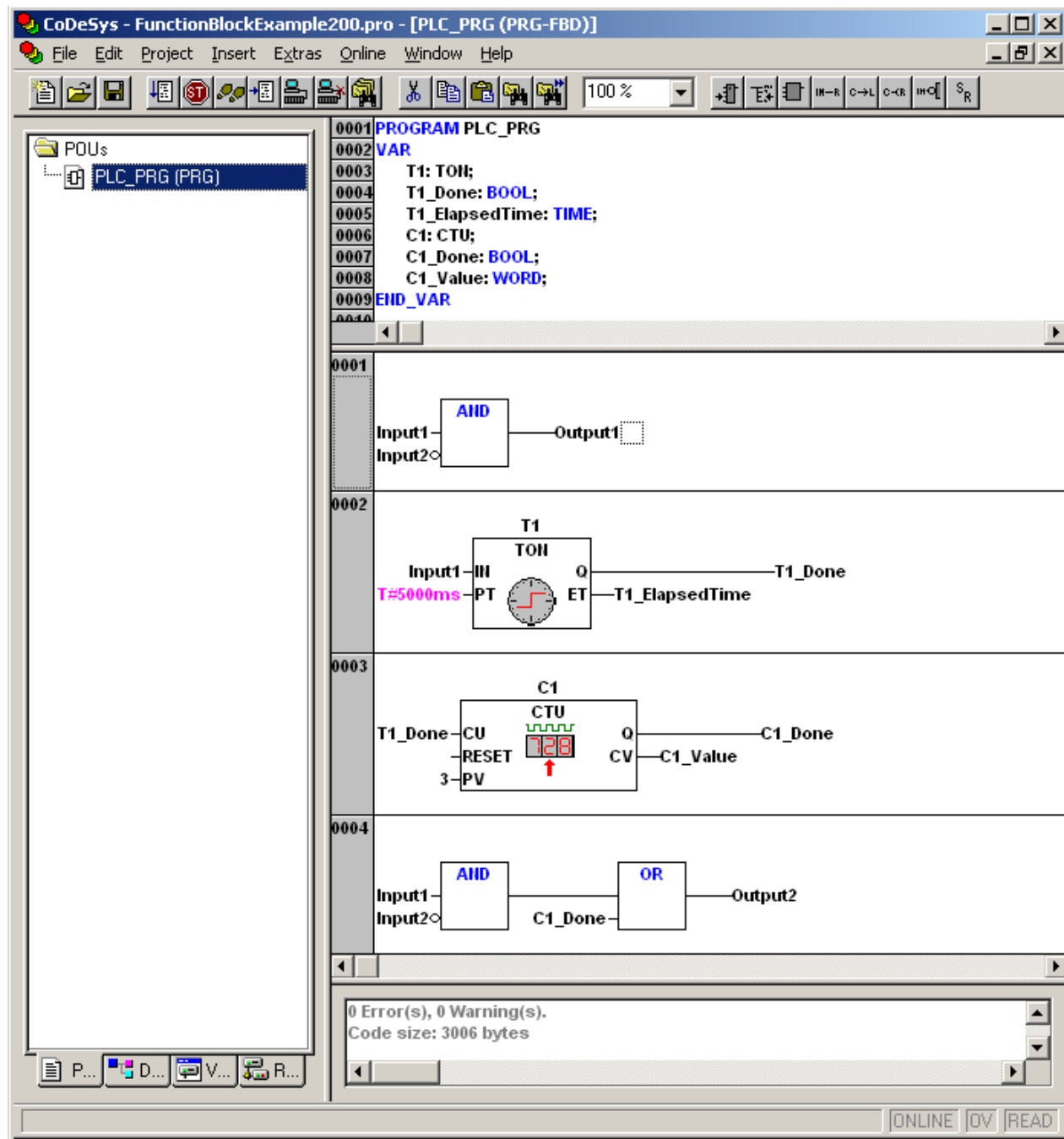


Click **Yes** to download your program.

To place your PFC in run Mode, select **Online..Run** from the top menu bar..

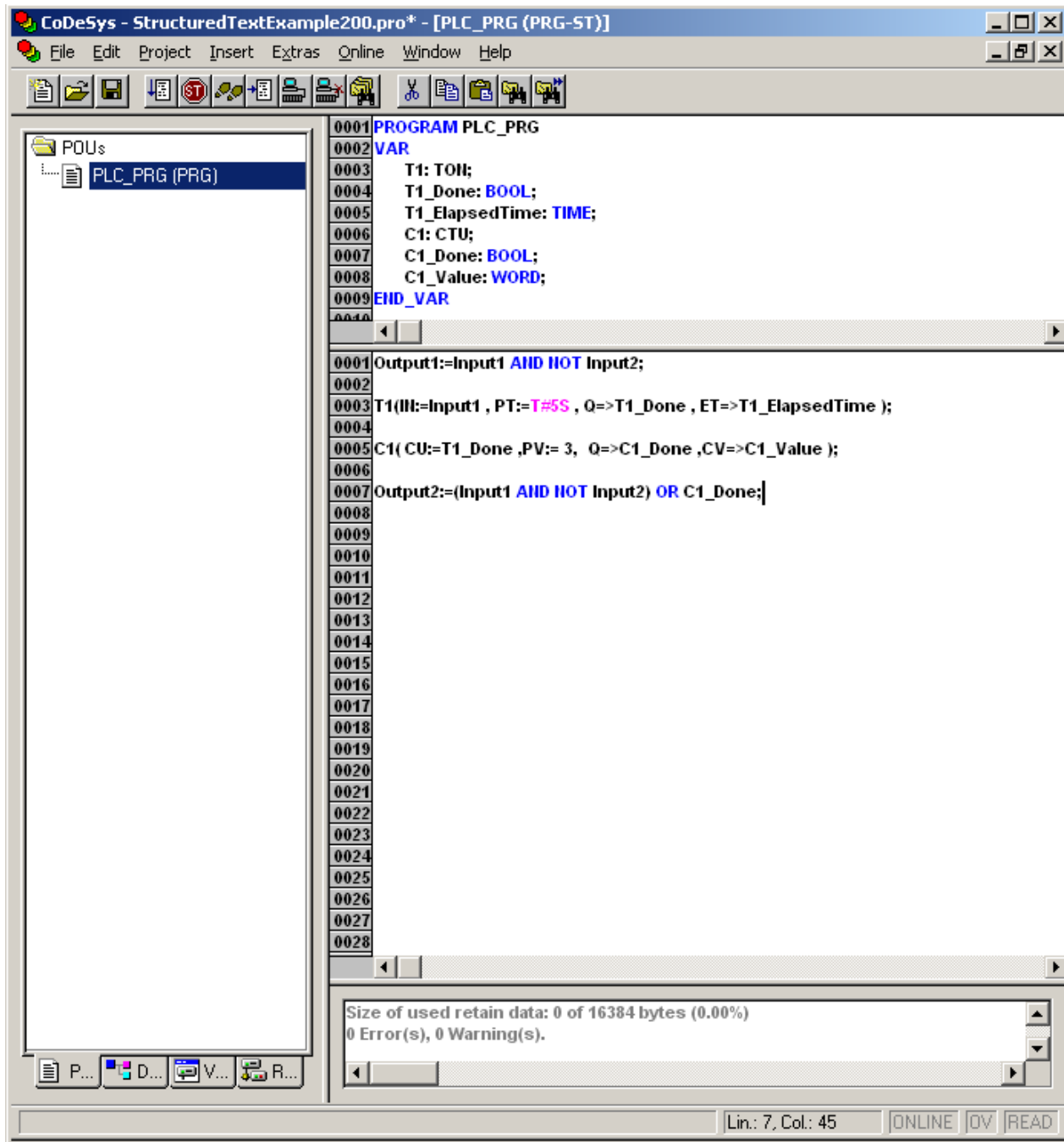
## PROGRAMMING EXAMPLE IN FUNCTION BLOCK DIAGRAM

Below is an example of how the project in the Programming Example might look if it were written in Function Block Diagram:



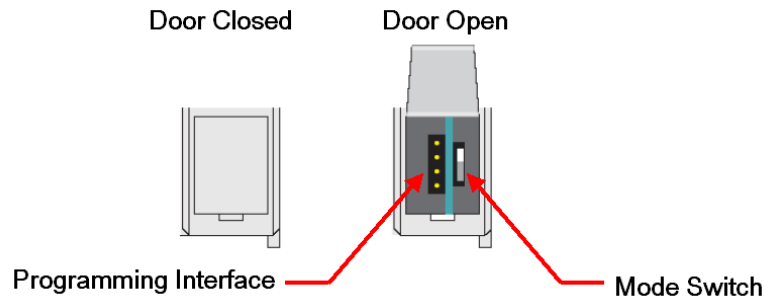
### PROGRAMMING EXAMPLE IN STRUCTURED TEXT

Below is an example of how the project in the Programming Example might look if it were written in Structured Text:



## MODE SWITCH AND PROGRAMMING INTERFACE

On the front of every WAGO PFC is a small door that allows access to the PFC's Mode Switch and Programming Interface.



The **Programming Interface** provides a serial connection to the PFC for programming and configuration (e.g., WAGO-IO-PRO CAA, WAGO-IO-CHECK, and WAGO-IO\_UPDATE software). A WAGO 750-920 communications cable is required to connect the Programming Interface 4-pin header to a PC's 9-pin RS232 port. The default settings for this interface are **19200 Baud, E, 8, 1**.

The **Mode Switch** is a slide/push switch with 3 slide positions. It provides the following functionality:

Switch Position	Function
<b>Top Position</b>	Switching from the middle to the top position will initiate PFC program execution. PFC's Boot Project will begin execution upon PFC power cycle if switch is in top position.
<b>Middle Position</b>	Switching from the top to the middle position halts the PFC program execution.
<b>Bottom Position</b>	Allows downloading of new firmware to the PFC. (This function can only be performed at the factory.)
<b>Pushed</b>	Pushing down on the operating mode switch while in any of the above positions will perform a hardware reset on the PFC. All outputs will be reset. All variables will be set to 0 (FALSE for boolean) or their initialized value.

## CREATING A BOOT PROJECT

A WAGO-IO-PRO CAA Project that has been successfully compiled (checked) can be downloaded to the PFC in such a way that the controller can load it automatically when restarted. This operation is called “Creating a Boot Project” and requires a licensed version of WAGO-IO-PRO CAA (demo versions of WAGO-IO-PRO do not allow this functionality).

To create a boot project, first login to the PFC. This is done by selecting **Online..Login** from the top menu bar. If your WAGO-IO-PRO CAA offline program does not match the current program in the PFC’s program memory, a dialog box will ask you to confirm the download, select **YES**.

Next, select **Online..Create boot project** from the main menu bar. A dialog box will appear in the middle of the WAGO-IO-PRO CAA work area as the program is copied to file memory in the PFC. When complete, power can be disconnected from the PFC and the program is retained in file memory. The program will remain in file memory until a new Boot Project is created or until the menu item **Online..Reset (Original)** is selected.

## APPENDIX A - STANDARD FUNCTIONS AND FUNCTION BLOCKS

WAGO-I/O-PRO CAA supports all the IEC 61131 Standard Functions and Function Blocks. Standard functions in WAGO-I/O-PRO CAA that are implicitly recognized by the programming environment (don't require the Standard library) are also called Operators.

To enter a standard function or function block in a WAGO-I/O-PRO CAA project, select **Insert** from the main menu bar and click on the appropriate item from the drop-down menu. For Structured Text POU's, select **Operator**; for Ladder Diagram POU's, select **Box with EN**; and for Function Block POU's, select **Box**. Function Block and Ladder Diagram POU's by default display the **AND** function. A different function or function block is selected by one of two means:

- 1) Click on the word **AND**, and hit the **F2** key. A selection box will appear listing all the available functions (operators) and function blocks. Select the desired item and click **OK**.
- 2) Click on the word **AND**, and type the name of the desired function (operator) and function block. The list of available options is shown below.

Consult the WAGO-I/O-PRO CAA Help Topics to learn more about functions, function blocks, and operators.

### IEC 61131 Standard Functions (Operators)

<b>ABS</b>	<b>EXPT</b>	<b>MIN</b>	<b>SHL</b>
<b>ACOS</b>	<b>GE</b>	<b>MOD</b>	<b>SHR</b>
<b>ADD</b>	<b>GT</b>	<b>MOVE</b>	<b>SIN</b>
<b>ADR</b>	<b>INDEXOF</b>	<b>MUL</b>	<b>SIN</b>
<b>AND</b>	<b>INI</b>	<b>MUX</b>	<b>SIZEOF</b>
<b>ASIN</b>	<b>LE</b>	<b>NE</b>	<b>SQRT</b>
<b>ATAN</b>	<b>LIMIT</b>	<b>NOT</b>	<b>SUB</b>
<b>COS</b>	<b>LN</b>	<b>OR</b>	<b>TAN</b>
<b>DIV</b>	<b>LOG</b>	<b>ROL</b>	<b>TRUNC</b>
<b>EQ</b>	<b>LT</b>	<b>ROR</b>	<b>XOR</b>
<b>EXP</b>	<b>MAX</b>	<b>SEL</b>	

### IEC 61131 Standard Function Blocks

<b>SR</b>	<b>CTU</b>	<b>TP</b>
<b>RS</b>	<b>CTD</b>	<b>TON</b>
<b>R_TRIG</b>	<b>CTUD</b>	<b>TOF</b>
<b>F_TRIG</b>		<b>RTC</b>

IEC 61131 does not permit implicit conversion from a “larger” variable type to a “smaller” type (for example, from INT to BYTE or from DINT to WORD). Special type conversions are required in order to accomplish this. WAGO-I/O-PRO CAA provides conversion Operators for converting from any elementary type to any other elementary type.

### IEC61131 Conversion Operators

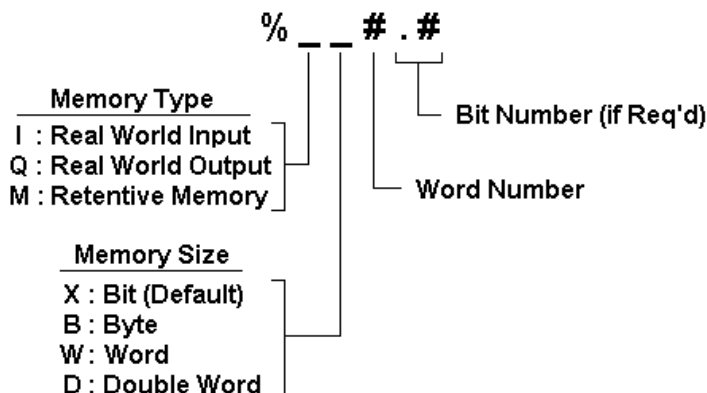
BOOL_TO_BYTE	DT_TO_BOOL	SINT_TO_BOOL	UDINT_TO_BOOL
BOOL_TO_DATE	DT_TO_BYTE	SINT_TO_BYTE	UDINT_TO_BYTE
BOOL_TO_DINT	DT_TO_DATE	SINT_TO_DATE	UDINT_TO_DATE
BOOL_TO_DT	DT_TO_DINT	SINT_TO_DINT	UDINT_TO_DINT
BOOL_TO_DWORD	DT_TO_DWORD	SINT_TO_DT	UDINT_TO_DT
BOOL_TO_INT	DT_TO_INT	SINT_TO_DWORD	UDINT_TO_DWORD
BOOL_TO_REAL	DT_TO_REAL	SINT_TO_INT	UDINT_TO_INT
BOOL_TO_SINT	DT_TO_SINT	SINT_TO_REAL	UDINT_TO_REAL
BOOL_TO_STRING	DT_TO_STRING	SINT_TO_STRING	UDINT_TO_SINT
BOOL_TO_TIME	DT_TO_TIME	SINT_TO_TIME	UDINT_TO_STRING
BOOL_TO_TOD	DT_TO_TOD	SINT_TO_TOD	UDINT_TO_TIME
BOOL_TO_UDINT	DT_TO_UDINT	SINT_TO_UDINT	UDINT_TO_TOD
BOOL_TO_UINT	DT_TO_UINT	SINT_TO_UINT	UDINT_TO_UINT
BOOL_TO_USINT	DT_TO_USINT	SINT_TO_USINT	UDINT_TO_USINT
BOOL_TO_WORD	DT_TO_WORD	SINT_TO_WORD	UDINT_TO_WORD
BYTE_TO_BOOL	DWORD_TO_BOOL	STRING_TO_BOOL	UINT_TO_BOOL
BYTE_TO_DATE	DWORD_TO_BYTE	STRING_TO_BYTE	UINT_TO_BYTE
BYTE_TO_DINT	DWORD_TO_DATE	STRING_TO_DATE	UINT_TO_DATE
BYTE_TO_DT	DWORD_TO_DINT	STRING_TO_DINT	UINT_TO_DINT
BYTE_TO_DWORD	DWORD_TO_DT	STRING_TO_DT	UINT_TO_DT
BYTE_TO_INT	DWORD_TO_INT	STRING_TO_DWORD	UINT_TO_DWORD
BYTE_TO_REAL	DWORD_TO_REAL	STRING_TO_INT	UINT_TO_INT
BYTE_TO_SINT	DWORD_TO_SINT	STRING_TO_REAL	UINT_TO_REAL
BYTE_TO_STRING	DWORD_TO_STRING	STRING_TO_SINT	UINT_TO_SINT
BYTE_TO_TIME	DWORD_TO_TIME	STRING_TO_TIME	UINT_TO_STRING
BYTE_TO_TOD	DWORD_TO_TOD	STRING_TO_TOD	UINT_TO_TIME
BYTE_TO_UDINT	DWORD_TO_UDINT	STRING_TO_UDINT	UINT_TO_TOD
BYTE_TO_UINT	DWORD_TO_UINT	STRING_TO_UINT	UINT_TO_UDINT
BYTE_TO_USINT	DWORD_TO_USINT	STRING_TO_USINT	UINT_TO_USINT
BYTE_TO_WORD	DWORD_TO_WORD	STRING_TO_WORD	UINT_TO_WORD
DATE_TO_BOOL	INT_TO_BOOL	TIME_TO_BOOL	USINT_TO_BOOL
DATE_TO_BYTE	INT_TO_BYTE	TIME_TO_BYTE	USINT_TO_BYTE
DATE_TO_DINT	INT_TO_DATE	TIME_TO_DATE	USINT_TO_DATE
DATE_TO_DT	INT_TO_DINT	TIME_TO_DINT	USINT_TO_DINT
DATE_TO_DWORD	INT_TO_DT	TIME_TO_DT	USINT_TO_DT
DATE_TO_INT	INT_TO_DWORD	TIME_TO_DWORD	USINT_TO_DWORD
DATE_TO_REAL	INT_TO_REAL	TIME_TO_INT	USINT_TO_INT
DATE_TO_SINT	INT_TO_SINT	TIME_TO_REAL	USINT_TO_REAL
DATE_TO_STRING	INT_TO_STRING	TIME_TO_SINT	USINT_TO_SINT
DATE_TO_TIME	INT_TO_TIME	TIME_TO_STRING	USINT_TO_STRING
DATE_TO_TOD	INT_TO_TOD	TIME_TO_TOD	USINT_TO_TIME
DATE_TO_UDINT	INT_TO_UDINT	TIME_TO_UDINT	USINT_TO_TOD
DATE_TO_UINT	INT_TO_UINT	TIME_TO_UINT	USINT_TO_UDINT
DATE_TO_USINT	INT_TO_USINT	TIME_TO_USINT	USINT_TO_UINT
DATE_TO_WORD	INT_TO_WORD	TIME_TO_WORD	USINT_TO_WORD
DINT_TO_BOOL	REAL_TO_BOOL	TOD_TO_BOOL	WORD_TO_BOOL
DINT_TO_BYTE	REAL_TO_BYTE	TOD_TO_BYTE	WORD_TO_BYTE
DINT_TO_DATE	REAL_TO_DATE	TOD_TO_DATE	WORD_TO_DATE
DINT_TO_DT	REAL_TO_DINT	TOD_TO_DINT	WORD_TO_DINT
DINT_TO_DWORD	REAL_TO_DT	TOD_TO_DT	WORD_TO_DT
DINT_TO_INT	REAL_TO_DWORD	TOD_TO_DWORD	WORD_TO_DWORD
DINT_TO_REAL	REAL_TO_INT	TOD_TO_INT	WORD_TO_INT
DINT_TO_SINT	REAL_TO_SINT	TOD_TO_REAL	WORD_TO_REAL
DINT_TO_STRING	REAL_TO_STRING	TOD_TO_SINT	WORD_TO_SINT
DINT_TO_TIME	REAL_TO_TIME	TOD_TO_STRING	WORD_TO_STRING
DINT_TO_TOD	REAL_TO_TOD	TOD_TO_TIME	WORD_TO_TIME
DINT_TO_UDINT	REAL_TO_UDINT	TOD_TO_UDINT	WORD_TO_TOD
DINT_TO_UINT	REAL_TO_UINT	TOD_TO_UINT	WORD_TO_UDINT
DINT_TO_USINT	REAL_TO_USINT	TOD_TO_USINT	WORD_TO_UINT
DINT_TO_WORD	REAL_TO_WORD	TOD_TO_WORD	WORD_TO_USINT



## APPENDIX B - MEMORY ADDRESSING

Addressable memory locations within the WAGO 750-841 PFC include inputs and outputs defined by the process image table, and storage locations in the PFC's 24Kbytes of retentive memory. The PFC's addressable memory is accessed in WAGO-IO-PRO CAA programming software using the following syntax:

### Syntax for Addressable Memory Locations in the PFC



### Examples of PFC memory location addresses:

%IX2.0	Input Bit 0 of Word 2
%I2.0	Input Bit 0 of Word 2 (Same as %IX2.0)
%IB7	Input Byte 7
%IW20	Input Word 20
%ID4	Input Double Word 4
%QX0.0	Output Bit 0 of Word 0
%Q0.0	Output Bit 0 of Word 0 (Same as %QX0.0)
%QB12	Output Byte 12
%QW15	Output Word 15
%QD2	Output Double Word 2
%MX5.0	Retentive Memory Bit 0 of Word 5
%M5.0	Retentive Memory Bit 0 of Word 5 (Same as %MX5.0)
%MB100	Retentive Memory Byte 100
%MW0	Retentive Memory Word 0
%MD48	Retentive Memory Double Word 48

Variables used within WAGO-IO-PRO CAA can be mapped to addressable memory locations using this syntax. Real-world I/O points are defined in the Hardware Configuration Editor, which automatically generates the syntax. Variables defined in the Variable Editor are assigned an address location by using the keyword **AT** with the address location (e.g., **VariableName AT %MW0: WORD;**).

### PFC Addressable Memory Map

	PFC Word Address	PFC Byte Address	PFC Bit Address	MODBUS/TCP Address	Memory Map Usage	Network/PFC Accessibility
Input Process Image	%IW0	%IB0, %IB1	%IX0.0 to %IX0.15	4x00001	Real World Analog Inputs & Real World Digital Inputs	Network Access is Read Only  PFC Access is Read Only
	%IW1	%IB2, %IB3	%IX1.0 to %IX1.15	4x00002		
	%IW2	%IB4, %IB5	%IX2.0 to %IX2.15	4x00003		
	thru	Thru	thru	thru		
	...	...	...	...		
	%IW255	%IB510, %IB511	%IX255.0 to %IX255.15	4x00256		
	%IW256	%IB512, %IB513	%IX256.0 to %IX256.15	4x00769	PFC Network Variable Memory  (Network to PFC)	Network Access is Read/Write  PFC Access is Read Only
	%IW257	%IB514, %IB515	%IX257.0 to %IX257.15	4x00800		
	%IW258	%IB516, %IB517	%IX258.0 to %IX258.15	4x00801		
	thru	Thru	thru	thru		
	...	...	...	...		
	%IW511	%IB1023, %IB1024	%IX511.0 to %IX511.15	4x01024		
Output Process Image	%QW0	%QB0, %QB1	%QX0.0 to %QX0.15	4x00513	Real World Analog Outputs & Real World Digital Outputs	Network Access is Read/Write  PFC Access is Read/Write
	%QW1	%QB2, %QB3	%QX1.0 to %QX1.15	4x00514		
	%QW2	%QB4, %QB5	%QX2.0 to %QX2.15	4x00515		
	thru	Thru	thru	thru		
	...	...	...	...		
	%QW255	%QB510, %QB511	%QX255.0 to %QX255.15	4x00768	PFC Network Variable Memory  (PFC to Network)	Network Access is Read Only  PFC Access is Read/Write
	%QW256	%QB512, %QB513	%QX256.0 to %QX256.15	4x00257		
	%QW257	%QB514, %QB515	%QX257.0 to %QX257.15	4x00258		
	%QW258	%QB516, %QB517	%QX258.0 to %QX258.15	4x00259		
	thru	Thru	thru	thru		
	...	...	...	...		
	%QW511	%QB1023, %QB1024	%QX511.0 to %QX511.15	4x00512		
Retentive Memory	%MW0	%MB0, %MB1	%MX0.0 to %MX0.15	4x12289	PFC Variable Retain Memory	Network Access is Read/Write  PFC Access is Read/Write
	%MW1	%MB2, %MB3	%MX1.0 to %MX1.15	4x12290		
	%MW2	%MB4, %MB5	%MX2.0 to %MX2.15	4x12291		
	thru	Thru	thru	thru		
	...	...	...	...		
	%MW12287	%MB24574, %MB24575	%MX12287.0 to %MX12287.15	4x24576		

\*Reference the 750-841 Ethernet TCP/IP Manual for additional PFC memory usage.